aws

Controls Reference Guide

AWS Control Tower



AWS Control Tower: Controls Reference Guide

Table of Contents

Introduction **1**
About controls **2**
About control relationships 2
Controls that have non-deployable Regions 4
Control behavior and guidance 4
Considerations for controls and OUs 6
Exception to controls for the management account 6
Exception to controls for the Security OU 7
Considerations for controls and accounts 9
Common controls 9
Control objectives 11
Legacy control objectives 30
Frameworks supported 31
View control details 32
Access control metadata in the console 32
Access control metadata programmatically 35
View enabled controls 35
Resource identifiers for APIs and controls **36**
Understand control ARNs 36
Find global ARNs 37
Migrate from regional ARNs to global ARNs 37
Find identifiers for OUs 37
Legacy control identifiers 38
Non-API controls 40
Control API examples **42**
EnableControl 42
DisableControl 43
GetControlOperation 43
GetEnabledControl 44
ListControlOperations 45
ListEnabledControls 49
ListTagsForResource 51
ResetEnabledControl 52
TagResource 52

UntagResource ... 52
UpdateEnabledControl ... 53
View parameters ... 54
Enable controls with CloudFormation ... 56
Create the stack through CloudFormation ... 56
Create the stack through CloudFormation and the AWS CLI ... 57
Configure controls with CloudFormation ... 57
Enable configurable controls with CloudFormation templates ... 58
Control Catalog by category ... 60
Mandatory controls ... 61
Changes in Landing Zone 4.0 controls ... 61
Mandatory controls for the Security OU ... 66
Mandatory controls for all OUs ... 73
Proactive controls ... 89
Update hooks ... 90
Amazon API Gateway controls ... 92
AWS Certificate Manager controls ... 139
AWS AppSync controls ... 146
Amazon Athena controls ... 176
Amazon CloudFront controls ... 185
AWS CloudTrail controls ... 291
Amazon CloudWatch controls ... 324
AWS CodeBuild controls ... 347
AWS Database Migration Service (AWS DMS) controls ... 405
Amazon DocumentDB controls ... 417
Amazon DynamoDB controls ... 428
DynamoDB Accelerator controls ... 440
AWS Elastic Beanstalk controls ... 463
Amazon Elastic Compute Cloud (Amazon EC2) controls ... 507
Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling controls ... 679
Amazon ElastiCache controls ... 768
Amazon Elastic Container Registry controls ... 831
Amazon Elastic Container Service controls ... 846
Amazon Elastic File System controls ... 936
Amazon Elastic Kubernetes Service (EKS) controls ... 957
Elastic Load Balancing controls ... 977

Amazon Elastic Map Reduce (Amazon EMR) controls 1114
AWS Glue controls 1145
Amazon GuardDuty controls 1152
AWS Identity and Access Management (IAM) controls 1158
AWS Key Management Service (AWS KMS) controls 1211
Amazon Kinesis controls 1234
AWS Lambda controls 1241
Amazon MQ controls 1283
Amazon Managed Streaming for Apache Kafka (Amazon MSK) controls 1297
Amazon Neptune controls 1319
AWS Network Firewall controls 1343
Amazon OpenSearch controls 1379
Amazon Relational Database Service (Amazon RDS) controls 1516
Amazon Redshift controls 1757
Amazon Simple Storage Service (Amazon S3) controls 1812
Amazon SageMaker AI controls 1884
Amazon Simple Queue Service (Amazon SQS) controls 1905
AWS Step Functions controls 1919
AWS WAF regional controls 1939
AWS WAF controls 1950
AWS WAFV2 controls 1961
Preventive controls 1974
About RCP controls 1975
Declarative policies 1993
Controls for AWS Backup 2000
Digital sovereignty group 2009
Detective controls 2066
More about detective controls 2066
The Security Hub CSPM standard 2067
AWS Config controls 2070
Parameterized controls 2072
Optional controls 2077
Strongly recommended controls 2078
Strongly recommended preventive 2079
Strongly recommended detective 2082
Elective controls 2094

Elective detective controls 2094
Elective preventive controls 2098
AWS assistant search for controls 2113
Tags for enabled controls 2115
Controls and compliance 2118
How can administrators review compliance? 2119
Compliance status in the console 2121
Drift prevention and notification 2123
Publishers and subscribers for SNS topics 2124
Compliance notifications by SNS and email 2125
The AWS Config SNS topic policy 2126
Control availability tables by Region 2128
Control metadata tables 2129
All global identifiers 2130
Document history 2131
AWS Glossary 2132

Introduction: AWS Control Tower Controls Reference Guide

The *AWS Control Tower Controls Reference Guide* accompanies the *AWS Control Tower User Guide*. It gives information about the controls that are specific to the AWS Control Tower environment.

The *Controls Reference Guide* contains

- A detailed page about each control that can be enabled through AWS Control Tower.
- Extensive reference tables showing the regional availability and the metadata for the controls.
- It explains how to view and enable controls from the AWS Control Tower console and with AWS Control Tower APIs.
- It includes examples of using the AWS CLI and AWS CloudFormation to work with controls.
- The *Controls and compliance* section explains how AWS Control Tower defines compliance and how an administrator can review compliance within the AWS Control Tower environment.

For more information about the AWS Control Tower application programming interface (API), see The AWS Control Tower API Reference.

Additional control APIs are available in the AWS Control Catalog namespace. The `GetControl` and `ListControls` APIs can help you find information about AWS Control Tower controls.

For information about availability of specific controls across AWS Regions, see Control limitations, and AWS Region availability for AWS Control Tower controls.

About controls in AWS Control Tower

A control is a high-level rule that provides ongoing governance for your overall AWS environment. It's expressed in plain language. AWS Control Tower implements *preventive*, *detective*, and *proactive* controls that help you govern your resources and monitor compliance across groups of AWS accounts.

A control applies to an entire organizational unit (OU), and every AWS account within the OU is affected by the control. Therefore, when users perform work in any AWS account in your landing zone, they're always subject to the controls that are governing their account's OU.

> **Note**
>
> We are transitioning our terminology to align better with industry usage and with other AWS services. During this time, you may see the previous term, *guardrail*, as well as the new term, *control*, in our documentation, console, blogs, and videos. These terms are synonymous for our purposes.

The purpose of controls

Controls assist you to express your policy intentions. For example, if you enable the detective control **Detect Whether Public Read Access to Amazon S3 Buckets is Allowed** on an OU, you can determine whether an entity (such as a user) would be permitted to have read access over the internet to any Amazon S3 buckets, for any accounts under that OU.

About control relationships

Certain controls stand in specified relationships to each other. These relationships are defined as follows:

- **Alternative**: One control can replace or substitute the other. Example: An AWS Config rule and a Security Hub CSPM control using the same underlying Config rule.
- **Complementary**: The related controls work together to strengthen governance, each covering different aspects of security and compliance and enhancing the effectiveness of the other. Example: A Security Hub CSPM control and a proactive control that both check that an Amazon S3 bucket should have **Block public access** settings configured.

- **Mutually Exclusive**: Controls cannot be enabled together on the same target or else either control fails in achieving the desired outcome. Example: Two proactive controls that enforce two incompatible features.

You can discover the control relationships in the AWS Control Tower console, or by calling the `ListControlMappings` API in Control Catalog. Here are some examples.

Find all related controls, request:

```
{
    "Filter": {
        "ControlArns": ["arn:aws:controlcatalog:::control/CONTROL_A_ARN"],
        "MappingTypes": ["RELATED_CONTROL"]
    }
}
```

Find all related controls, response:

```
{
    "ControlMappings": [
        {
            "ControlArn": "arn:aws:controlcatalog:::control/CONTROL_A_ARN",
            "MappingType": "RELATED_CONTROL",
            "Mapping": {
                "RelatedControl": {
                    "ControlArn": "arn:aws:controlcatalog:::control/CONTROL_B_ARN",
                    "RelationType": "ALTERNATIVE"
                }
            }
        }, {
            "ControlArn": "arn:aws:controlcatalog:::control/CONTROL_A_ARN",
            "MappingType": "RELATED_CONTROL"
            "Mapping": {
                "RelatedControl": {
                    "ControlArn": "arn:aws:controlcatalog:::control/CONTROL_C_ARN",
                    "RelationType": "COMPLEMENTARY"
                }
            }
        }
        ...
    ],
    "NextToken": "..."
```

```
}
```

For more information, see ListControlMappings.

Controls that have non-deployable Regions

This section lists controls that are not activated when deployed in certain Regions, due to lack of underlying dependencies. This section presents summary information about these non-deployable Regions, for quick reference. You can find the most updated information about the deployable Regions for any control by calling the `ListControls` and `GetControl` APIs. You also can view the deployable Regions in the AWS Control Tower console.

When you activate a control on an OU that's governed by AWS Control Tower, the control's effective area is the intersection of your governed Regions with the control's deployable Regions, with a few minor caveats related to occasional states of mixed governance.

For example, a control can be enabled on an OU that operates in governed Regions X, Y and Z. But after it is enabled, the same control is deployed only on Regions X and Z, because the control itself does not support Region Y.

It's important to monitor the relationships among controls that you deploy and Regions where you operate workloads, so that you don't experience gaps in protection of your AWS resources.

How to check your protected Regions

- In the AWS Control Tower console, you can view the enabled controls and Regions in the **Enabled controls** section.
- If you call the `GetEnabledControl` API, the **targetRegions** parameter will show only those Regions where you can deploy the control effectively not the non-deployable Regions..

Control behavior and guidance

Controls are categorized according to their *behavior* and their *guidance*.

The *behavior* of each control is one of preventive, detective, or proactive. Control *guidance* refers to the recommended practice for how to apply each control to your OUs. The guidance of a control is independent of whether its behavior is preventive, detective, or proactive.

Control behavior

- **Preventive** – A preventive control ensures that your accounts maintain compliance, because it disallows actions that lead to policy violations. The status of a preventive control is either **enforced** or **not enabled**. Preventive controls are supported in all AWS Regions.
- **Detective** – A detective control detects noncompliance of resources within your accounts, such as policy violations, and provides alerts through the dashboard. The status of a detective control is either **clear**, **in violation**, or **not enabled**. Detective controls apply only in those AWS Regions supported by AWS Control Tower.
- **Proactive** – A proactive control scans your resources before they are provisioned, and makes sure that the resources are compliant with that control. Resources that are not compliant will not be provisioned. Proactive controls are implemented by means of AWS CloudFormation hooks, and they apply to resources that would be provisioned by AWS CloudFormation. The status of a proactive control is PASS, FAIL, or SKIP. For more information about AWS CloudFormation hooks, see Characteristics of hooks in the AWS CloudFormation documentation.

Implementation of control behavior

- The preventive controls are implemented using Service Control Policies (SCPs), Resource Control Policies (RCPs), and declarative policies, which are part of AWS Organizations.
- The detective controls are implemented using AWS Config rules.
- The proactive controls are implemented using AWS CloudFormation hooks.
- Certain mandatory controls are implemented by means of a single SCP that performs multiple actions, rather than as unique SCPs. Therefore, the same SCP is shown in the control reference, under each mandatory control to which that SCP applies.
- The integrated, detective Security Hub controls are implemented using AWS Config rules, similarly to all Security Hub controls. These controls are owned by the **Service-Managed Standard: AWS Control Tower**, which is part of Security Hub.
- The integrated AWS Config controls that are available in the control catalog are owned by AWS Config and implemented as Config rules, exactly as any other AWS Config controls.

Control guidance

AWS Control Tower provides three categories of guidance: *mandatory*, *strongly recommended*, and *elective* controls.

- Mandatory controls are enforced in your landing zone depending on what you may have enabled in your environment. These controls protect AWS Control Tower-deployed resources.
- Strongly recommended controls are designed to enforce some common best practices for well-architected, multi-account environments. These controls apply at the OU level, for all accounts in that OU.
- Elective controls enable you to track or lock down actions that are commonly restricted in an AWS enterprise environment. These controls apply at the OU level, for all accounts in that OU.

Starting with AWS Control Tower Landing Zone version 4.0, mandatory controls are no longer applied by default.

Considerations for controls and OUs

When working with controls and OUs, consider the following properties:

Controls, landing zones, and OUs

- Mandatory controls are no longer enabled by default. Optional controls are applied at the discretion of administrators.
- Controls can now be enabled on any OU within a customer's AWS Organization once they enable AWS Control Tower.
- Regarding nested OUs, preventive controls enabled on any OUs higher in the tree will apply all OUs in the tree.
- Detective controls can be applied to an OU that has either the ConfigBaseline enabled or the AWSControlTowerBaseline.
- Hook controls can now be deployed into any OU. The hook will deploy the AWSServiceRoleForControlTower Service Linked Role (SLR), into the account and activate the opt-in regions.

For more information about how controls are applied to nested OUs, in AWS Control Tower, see Nested Ous and controls.

Exception to controls for the management account

The root user and any administrators in the management account can perform work that controls would otherwise deny. This exception is intentional. It prevents the management account from

entering into an unusable state. All actions taken within the management account continue to be tracked in the logs contained within the log archive account, for purposes of accountability and auditing.

Exception to controls for the Security OU

For customers on LZ v4.0:

There is no longer a Security OU managed by AWS Control Tower so restrictions below do not apply.

For existing customers on LZ v3.3 and below:

AWS Control Tower deploys and manages resources in the Security OU, which are required so that AWS Control Tower can function properly. You can deploy certain preventive controls (SCP-based) and detective controls (based on AWS Config rules) to this OU. Most controls cannot be enabled for this OU.

Controls that cannot be deployed to the Security OU

- You cannot deploy proactive controls to the Security OU.
- You cannot deploy Security Hub controls to the Security OU.
- You cannot deploy RCP-based controls to the Security OU.
- You cannot deploy declarative policies to the Security OU.
- Certain SCP-based controls cannot be deployed to the Security OU.

Controls that are deployable to the Security OU

- All controls implemented by AWS Config rules
- AWS-GR_AUDIT_BUCKET_DELETION_PROHIBITED (Mandatory)
- AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED
- AWS-GR_AUDIT_BUCKET_LOGGING_ENABLED
- AWS-GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED (Mandatory)
- AWS-GR_AUDIT_BUCKET_RETENTION_POLICY
- AWS-GR_CLOUDTRAIL_CHANGE_PROHIBITED
- AWS-GR_CLOUDTRAIL_CLOUDWATCH_LOGS_ENABLED

- AWS-GR_CLOUDTRAIL_ENABLED
- AWS-GR_CLOUDTRAIL_VALIDATION_ENABLED
- AWS-GR_CLOUDWATCH_EVENTS_CHANGE_PROHIBITED
- AWS-GR_CONFIG_AGGREGATION_AUTHORIZATION_POLICY
- AWS-GR_CONFIG_AGGREGATION_CHANGE_PROHIBITED
- AWS-GR_CONFIG_CHANGE_PROHIBITED
- AWS-GR_CONFIG_ENABLED
- AWS-GR_CONFIG_RULE_CHANGE_PROHIBITED
- AWS-GR_CT_AUDIT_BUCKET_ENCRYPTION_CHANGES_PROHIBITED (Mandatory)
- AWS-GR_CT_AUDIT_BUCKET_LIFECYCLE_CONFIGURATION_CHANGES_PROHIBITED (Mandatory)
- AWS-GR_CT_AUDIT_BUCKET_LOGGING_CONFIGURATION_CHANGES_PROHIBITED (Mandatory)
- AWS-GR_CT_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED
- AWS-GR_DISALLOW_CROSS_REGION_NETWORKING
- AWS-GR_DISALLOW_VPC_INTERNET_ACCESS
- AWS-GR_DISALLOW_VPN_CONNECTIONS
- AWS-GR_IAM_ROLE_CHANGE_PROHIBITED
- AWS-GR_LAMBDA_CHANGE_PROHIBITED
- AWS-GR_LOG_GROUP_POLICY
- AWS-GR_REGION_DENY
- AWS-GR_RESTRICT_ROOT_USER
- AWS-GR_RESTRICT_ROOT_USER_ACCESS_KEYS
- AWS-GR_RESTRICT_S3_CROSS_REGION_REPLICATION
- AWS-GR_RESTRICT_S3_DELETE_WITHOUT_MFA
- AWS-GR_SNS_CHANGE_PROHIBITED
- AWS-GR_SNS_SUBSCRIPTION_CHANGE_PROHIBITED
- CT.BACKUP.PV.1
- CT.BACKUP.PV.2
- CT.BACKUP.PV.3
- CT.CLOUDFORMATION.PR.1

- CT.IAM.PV.1
- CT.S3.PV.1
- CT.S3.PV.7
- CT.S3.PV.8
- CT.SNS.PV.1

Considerations for controls and accounts

When working with controls and accounts, consider the following properties:

Controls and accounts

- Accounts created through the Account Factory in AWS Control Tower inherit the controls of the parent OU, and the associated resources are created.
- When you enable optional controls, AWS Control Tower creates and manages certain additional AWS resources in your accounts. Do not modify or delete resources created by AWS Control Tower . Doing so could result in the controls entering an unknown state. For more information, see The AWS Control Tower controls library.

About common controls

This page provides an overview and a partial list of the common controls available for the Control Catalog. You can get a complete list of common controls by calling the `ListCommonControls` API.

Common controls in the ontology

- In the *Control Catalog ontology*, a *common control* conceptually expresses a single constraint or outcome that controls can help you to accomplish.
- In the hierarchy of the ontology, each common control has a single *control objective*.

A common control is more granular than a control objective, but the common control is not limited to any particular implementation. In fact, a common control can be implemented in several different ways, by individual controls, that you can view and enable. Hierarchically, a common control is the parent of certain individual controls that you see in the Control Catalog.

Because common controls are free of implementation, they do not have a defined **Behavior** or **Guidance**. A common control may be implemented by various preventive, detective, or proactive controls, or a combination of these. Common controls cannot actually be enabled, only the implementations can be enabled.

Common controls and industry frameworks

Each common control can be mapped to several industry frameworks, because that common control can help you meet specific framework requirements. In the Control Catalog ontology, an industry framework is represented by the term *Standard control*, because the standard control represents a specific requirement of an industry standard.

> **Note**
>
> In the AWS Control Tower console, the field called **Frameworks** on the **Control detail** page shows the frameworks that are related to the control.

View common controls

You can view each common control on a page in the console, and see a list of specific controls that implement the common control's functionality.

In the AWS Control Tower console, you can view the **Common control** field on the **Control details** page of a control. Each implemented control has a parent common control.

You can find the common control that is related to any specific control programmatically, by calling the `GetControl` or `ListControls` API.

Examples of common controls

- Log aggregation
- Secure development environment
- Network topology design and review
- Asset retirement and disposition
- Data backup procedures
- Evidence preservation procedures and chain of custody
- Security metrics
- Guest and limited access wireless netwok management

- Asset labeling
- Security cameras
- Vendor incident management
- Data error checking and correction
- Secure encryption protocols
- Log protection and integrity
- Collaboration and communication
- External vulnerability scanning
- Patch testing and approval
- Access request and approval workflows
- Rollback and recovery procedures
- Security testing

Control catalog: control objectives

This document gives details about the control objectives from the AWS Control Tower Control Catalog.

Control Catalog: control objectives

Number	Objective	Explanation
1	Asset inventory management	This control objective focuses on maintaining an accurate and up-to-date inventory of assets, including hardware, software, and data, to protect organization investments from harm or loss.
2	Asset classification	This control objective focuses on classifying assets based on their value, sensitivity, and criticality to the organization to manage investment risk

Number	Objective	Explanation
		and unauthorized access to assets and information.
3	Asset maintenance	This control objective focuses on maintaining the availability and integrity of assets, including performan ce management, regular maintenance, and repairs to protect and extract the maximum value of the organization's IT investments.
4	Asset lifecycle management	This control objective focuses on managing assets throughout their entire lifecycle, including acquisiti on, deployment, use, and retirement. This helps manage risks associated with asset costs by ensuring optimum asset productivity, performance, efficiency, and profitability.
5	Asset loss prevention, response, and recovery	This control objective focuses on preventing asset loss, and responding to and recoverin g lost, stolen, or damaged assets to contribute to the organization's profitability by reducing losses.

Number	Objective	Explanation
6	Business continuity	This control objective focuses on developing and maintaini ng plans, procedures, and protocols that support an organization's ability to recover critical business functions in the event of a disruption, including backup and recovery and business impact analysis.
7	Disaster recovery	This control objective focuses on the steps and technolog ies necessary to recover critical information resources in the event of a natural disaster, security event and/ or incident, and/or system outage and ensure critical business functions can continue.
8	Crisis and emergency management	This control objective focuses on the development and maintenance of plans and procedures to mitigate the effects of and recover from a crisis or emergency, and ensure that critical business functions can continue.

Number	Objective	Explanation
9	Data classification and handling	This control objective focuses on the classification of data based on its sensitivity and implementation of appropria te controls for handling and protecting data based on its classification. This includes data handling procedures, access controls, and data loss prevention (DLP) solutions to minimize the risk of data loss, corruption, or compromise.
10	Data integrity	This control objective focuses on data integrity, such as data validation, checksum verificat ion, and digital signing, to ensure data is reliable and traceable to origin.
11	Data retention and disposal	This control objective focuses on the implementation of policies for retaining and disposing of data in a secure and compliant manner. This includes securely deleting data from storage devices, wiping data from devices before disposal, and establish ing retention periods for different types of data.

Number	Objective	Explanation
12	Data backup and recovery	This control objective focuses on the implementation of backup and recovery procedures to ensure that data can be restored in the event of a data loss incident. This includes regular backups, offsite storage, and testing of backup and recovery procedures.
13	Data encryption	This control objective focuses on the use of encryption to protect data both in transit and at rest minimize the risk of data loss, corruption, or compromise. This includes using encryption for email, file transfer, and storage encryption.
14	Cryptographic key management	This control objective focuses on processes for managing cryptographic keys throughout their lifecycle, from creation to destruction, to minimize the risk of data loss, corruption, or compromise.

Number	Objective	Explanation
15	Data anonymization, tokenization, masking, and redaction	This control objective focuses on data anonymization, tokenization, masking, and redaction to protect sensitive data to minimize the risk of unauthorized access to data, and data loss, corruption, or compromise. This includes truncating and replacing sensitive informati on with realistic data, but fictitious data or tokens to represent sensitive data for other purposes, such as data analytics.
16	Identity management	This control objective focuses on the management of digital identities, including the creation, maintenance, verification, and retirement of user accounts, enforceme nt of authentication and authorization policies, and use of federated identities to reduce the risk of unauthori zed access to resources and data.
17	Authentication and access control	This control objective focuses on user and system authentic ation, password managemen t policies, privileged account management, and access controls restrictions.

Number	Objective	Explanation
18	Identity governance and administration (IGA)	This control objective focuses on the policies, procedures, and technologies used to manage user identities and access entitlements throughout their lifecycle.
19	Incident response planning	This control objective focuses on developing and maintaining incident response plans, assimilating an incident response team, defining roles and responsibilities, and conducting incident response training and exercises. This enables the organization to act quickly and respond efficiently in the event of a threat.
20	Incident containment and mitigation	This control objective focuses on containing and limiting the impact of security incidents, and mitigating the root cause of an incident to prevent further damage to the organization and assets.

Number	Objective	Explanation
21	Incident investigation and response	This control objective focuses on investigating security incidents, including preservin g and analyzing evidence, conducting interviews, performing root cause analysis, and implement ing remediation measures. This enables organizations to take corrective actions to prevent recurrence and demonstrates the organizat ion's commitment to protect the well-being of people.
22	Incident reporting and communication	This control objective focuses on reporting potential security incidents to appropriate security personnel and communica ting security incidents to relevant stakeholders to improve risk managemen t, take necessary response actions, and track incident progress to help determine business impacts.
23	Incident metrics and continuous improvement	This control objective focuses on measuring and tracking incident management performance, such as incident response times, incident resolution rates, and root cause analysis outcomes

Number	Objective	Explanation
24	Privacy Laws	This control objective focuses on regulations and laws to protect individuals' personal data from unauthorized access, misuse, or disclosur e. The intent is to ensure that entities collecting or processing such data do so responsibly, transpare ntly, and with the individua l's knowledge and consent, thereby safeguarding an individual's right to privacy and trust in the digital ecosystem.
25	Log generation and integrity	This control objective focuses on log generation, including security logs and audit trails, to record activity and reduce fraud, errors, and unauthori zed use within the organizat ion's compute environment. This includes ensuring log integrity and confidentiality remain intact from the point of generation.
26	Log retention	This control objective focuses on retaining and archiving log data for a specified period of time to meet regulatory requirements and support incident response and forensic investigations.

Number	Objective	Explanation
27	Log aggregation and analysis	This control objective focuses on real-time monitoring of logs and events. This involves aggregating, normalizing, correlating, analyzing, and reviewing log data, including security events, audit trails, and user activity, from multiple sources to identify security events or patterns that may indicate an ongoing or emerging threat.
28	Alerting and Notification	This control objective focuses on generating alerts and notifications to relevant parties of potential security incidents or risks based on log monitoring and analysis results.
29	Log monitoring and event management tools	This control objective focuses on selecting, implement ing, and maintaining log monitoring and event management tools and technologies to support real-time log analysis, retention, and protection. This supports timely response to anomalous activity and security events reducing the risk of undetecte d or unknown threats to the organization.

Number	Objective	Explanation
30	Network architecture and secure configuration	This control objective focuses on network architecture, design, and secure configura tions network devices or services, including routing switching, and firewall solutions to prevent malicious and unnecessary content from entering the environme nt.
31	Network monitoring	This control objective focuses on monitoring and analyzing network traffic, including intrusion detection and prevention and network flow analysis to identify and respond to potential threat quickly before they escalate into a serious security incident.
32	Wireless network security	This control objective focuses on management of wireless network access, including secure configuration and encryption.

Number	Objective	Explanation
33	Network filtering	This control objective focuses on filtering and inspecting network traffic for unwanted or malicious content to reduce the risk of unauthorized data exfiltration, spread of malware, and impact of compromised systems to other resources.
34	Physical security management	This control objective focuses on the organization's ability to ensure the physical security of corporate facilities, data centers, and other locations to prevent or reduce threats to people, information, and assets that may cause damage or loss. This includes surveillance and physical access management processes and systems.
35	Environmental protection controls	This control objective focuses on securing the physical environment of assets, such as temperature and humidity controls, fire suppression systems, water detection mechanisms, and redundant power supply, to prevent or reduce threats to people, information, and assets that may cause damage or loss.

Number	Objective	Explanation
36	Vulnerability scanning and remediation	This control objective focuses on scanning systems and applications to identify and remediate known vulnerabilities to reduce the risk of attack and exploitations.
37	IT risk assessment and management	This control objective focuses on identifying, assessing, prioritizing, reporting, and responding to risks, and implementing risk mitigation strategies to reduce risks to acceptable levels based on defined risk tolerance.
38	Vulnerability assessments and prioritization	This control objective focuses on processes for determining the severity of risks, threats, and vulnerabilities and prioritizing them for remediation based on the level of risk they pose to reduce chances of attack and exploitations.
39	Continuous vulnerability monitoring	This control objective focuses on continuously monitoring and actively searching for vulnerabilities and threats to detect vulnerabilities in real-time enabling faster response and containment of a security incident to reduce risk of escalation.

Number	Objective	Explanation
40	Threat intelligence	This control objective focuses on gathering and analyzing information on current and emerging threats to proactively identify vulnerabilities and potential attack vectors, and reduce the risk of security incidents.
41	Vulnerability reporting and metrics	This control objective focuses on determining and reporting on key performance indicators (KPIs) and metrics to measure the effectiveness of threat and vulnerability management processes and communicate risk to stakeholders.
42	Offensive security	This control objective focuses on actively identifying and exploiting vulnerabilities in computer systems and networks to assess their security posture and develop strategies to improve their overall security to reduce the risk of exploitation and attack.

Number	Objective	Explanation
43	Malware protection	This control objective focuses on preventing, detecting, and remediating malware infections on systems and networks to reduce the risk of attack and exploitation.
44	Development lifecycle processes	This control objective focuses on the software developme nt lifecycle processes, such as requirements gathering, design, coding, deploymen t, and maintencence of code to reduce the risk of introducing vulnerabilities or insecure code into production environments.
45	Code reviews and testing	This control objective focuses on testing and reviewing code to ensure that it meets the requirements and is securely developed to reduce the risk of introducing vulnerabi lities or insecure code in to production environments.

Number	Objective	Explanation
46	Secure configuration management	This control objective focuses on maintaining secure configurations of systems and software, and managing configuration drift, to reduce the risk of performance issues, inconsistencies, errors and compliance issues that can lead to compromise and unintended data exposure.
47	Patch management	This control objective focuses on maintaining system and software security and functionality by ensuring that patches and updates are applied in a timely and effective manner to reduce the risk of cyberattacks and compromise.
48	Change management	This control objective focuses on managing changes to the software, including testing, approval, and implement ation to minimize disruptio n, reduce costly back-out activities, and provide clear communication of changes to stakeholders.

Number	Objective	Explanation
49	DevOps	This control objective focuses on integrating developme nt and operations teams to ensure the rapid and secure deployments, accelerated innovation, and reduced failure rates and recovery times.
50	Compliance management	This control objective focuses on establishing and enforcing policies, procedures, and controls that adhere to applicable regulatory security and compliance requirements and industry best practices that ensure operational accountability with organizat ional obligations that govern the business.

Number	Objective	Explanation
51	Security governance & program management	This control objective focuses on establishing the set of practices, policies, and procedures that guide an organization's approach to information security to ensure alignment with the organization's business goals and objectives to protect the organization's assets from unauthorized access, use, disclosure, modification, or destruction. This involves establishment of a consisten t and structured information security program and strategy that supports employee awareness and acknowled gement of security policies outlining employee responsib ilities for complying with organizational governance.

Number	Objective	Explanation
52	Security awareness	This control objective focuses on security awareness for individuals within an organization by providing training, knowledge, understanding, and behaviora l awareness related to security risks and best practices, with the goal of preventing security incidents and improving overall security posture.
53	Vendor selection	This control objective focuses on the organization establish ing risk-based evaluatio n criteria, due diligence procedures, and contractual clauses for third party vendor, supplier, and service provider selection to identify and prepare for vendor risks, and avoid disruption to business performance.

Number	Objective	Explanation
54	Vendor management	This control objective focuses on assessing and managing third party risks, monitoring and reviewing third party activities, and managing third party security incidents to identify vendor risks and reduce potential for business disruption or negative impact on business performance. Third-parties include vendors, suppliers, and service providers.

Note

This data was generated from Control Catalog using the command: `aws controlcatalog list-objectives`

Legacy control objectives

Warning

This page will be removed in a future release.

These control objectives were the original objectives for AWS Control Tower controls. As AWS Control Tower has expanded to include more industry frameworks, we have expanded the list of objectives. This list is available as historical guidance, to help you make the transition to newer controls and API implementations.

Sometimes controls must be applied in a group so that the control objective is enforced. Information about related controls is viewable in the AWS Control Tower console, on the **Control details** page.

Legacy control objectives

For more information about controls, see *AWS Control Tower Controls Reference Guide*. To retrieve the most up-to-date list of new control objectives, call the `ListObjectives` API from the *controlcatalog* namespace of AWS Control Tower.

- **CO.1** Establish logging and monitoring
- **CO.2** Encrypt data at rest
- **CO.3** Encrypt data in transit
- **CO.4** Protect data integrity
- **CO.5** Enforce least privilege
- **CO.6** Limit network access
- **CO.7** Optimize costs
- **CO.8** Improve resiliency
- **CO.9** Improve availability
- **CO.10** Protect configurations
- **CO.11** Prepare for incident response
- **CO.12** Manage vulnerabilities
- **CO.13** Manage secrets
- **CO.14** Prepare for disaster recovery
- **CO.15** Use strong authentication

Frameworks supported

The controls available in Control Catalog support several industry frameworks.

- ACSC-Essential-Eight-Nov-2022
- ACSC-ISM-02-Mar-2023
- AWS-WAF-v10
- CCCS-Medium-Cloud-Control-May-2019

- CIS-AWS-Benchmark-v1.2
- CIS-AWS-Benchmark-v1.3
- CIS-AWS-Benchmark-v1.4
- CIS-v7.1
- CIS-v8.0
- FedRAMP-r4
- ISO-IEC-27001:2013-Annex-A
- NIST-CSF-v1.1
- NIST-SP-800-171-r2
- NIST-SP-800-53-r5
- PCI-DSS-v3.2.1
- PCI-DSS-v4.0
- SSAE-18-SOC-2-Oct-2023

For more information, see the Control Catalog Ontology overview.

View control details

You can view control details in the AWS Control Tower console or retrieve them programmatically using Control Catalog APIs.

Access control metadata in the console

To view details about an individual control in the AWS Control Tower console, select the name of the control from the table on the **Controls** page. On the console page for the control, you may see metadata items such as **Categories**: Common controls, Frameworks, Services, and Groups. Select each item to get more information about the control. You can find additional information in the tabs: **About**, **OUs enabled**, and **Accounts**.

For each control, the global **API controlIdentifier** is available in the console, along with the framework and objective.

In each **Control details** page of the console, you can find the following details for each control:

- **Name** – The name of the control.

- **Common control (formerly, Control objective)** – The pre-defined objective that an implemented control helps you enforce. See Control catalog: control objectives and About common controls.
- **Service** – The AWS service to which this control applies.
- **Control owner** – The AWS service that owns and maintains this control.
- **Behavior** – A control's behavior is set to preventive, detective, or proactive.
- **Implementation** – The underlying implementation method for this control, such as SCP, AWS Config managed rule, or CloudFormation hook.
- **GovernedResources** – The AWS resources that are monitored or affected by this control. This field can show an AWS service name, or an CloudFormation ARN. It can be blank if there's no CloudFormation ARN to represent the resource, or if the control governs resources across several AWS services (for example, the Region Deny control).
- **Framework** – The industry-standard compliance framework that this control helps to enforce, for example, **NIST 800-53 Rev 5**.
- **Control ID (Alias)** – A unique identifier assigned to each control. This identifier is part of a classification system for the controls.
- **API controlIdentifier** – This identifier is needed when calling the AWS Control Tower APIs.
- **Group** – A label for a group of controls with similar purpose, such as helping you create Digital Sovereignty.
- **Guidance** – The guidance is either mandatory, strongly recommended, or elective.
- **Severity** – The relative risk associated with any violation of this control.
- **Release date** –The date the control became available.
- **Deployable Regions** – Regions in which the control is available to be deployed.

> **Note**
>
> The control **State** and status information is available in the console only. It is not available from the public API. To view the status of a control, navigate to the **Control details** page in the AWS Control Tower console.

Open the tabs

- In the **About** tab, you can view the relationship of the control with other controls. We provide recommendations about how certain controls can work together with other controls to provide best security for your AWS environment.
- The **OUs enabled** tab shows a list of OUs on which the control is actively enabled.

The status of the Region deny control is shown as a separate entry.

Other information may appear on the **Control details** page, including these items:

- **Description** – A brief description of the control and its function.
- **Remediation message** – Suggestions for what to change if your CloudFormation hook control returns a FAIL status.
- **Remediation samples** – Examples showing configurations that can return a PASS or FAIL result for your CloudFormation hook control.
- **Usage considerations** – Additional information about how to apply this control or about the resources it can affect.
- The **Gherkin** artifact – The Gherkin is a readable specification for the CloudFormation hook controls, showing requirements for tests that cause PASS, FAIL, or SKIP results to be returned.

To view a control artifact

Each control is implemented by one or more artifacts. These artifacts can include a baseline CloudFormation template, a service control policy (SCP) to prevent account-level configuration changes or activity that may create configuration drift, and AWS Config Rules to detect account-level policy violations.

To view a control's artifact, select the **Artifact** tab to view the **Service control policy (SCP)**, **AWS Config rule**, or **CloudFormation policy template** on the **Control details** page.

> **Note**
>
> The four mandatory controls with `"Sid": "GRCLOUDTRAILENABLED"` are identical by design. The sample code is correct.

Access control metadata programmatically

Control metadata is available through Control Catalog APIs. For information about control identifiers, see Resource identifiers for APIs and controls.

To list all controls, use the ListControls API. For example:

```
aws controlcatalog list-controls --max-result 2 --region us-east-1
```

To retrieve detailed metadata for a specific control, use the GetControl API. For example:

```
aws controlcatalog get-control --control-arn
 arn:aws:controlcatalog:::control/4b0nsxnd47747up54ytdqesxi --region us-east-1
```

For more information, see the Control Catalog API Reference.

View enabled controls

To view your enabled controls in the AWS Control Tower console, navigate to the **Enabled Controls** page by selecting it from the left navigation pane.

You also can view your enabled controls programmatically, by calling the `ListEnabledControls` API.

Resource identifiers for APIs and controls

Each control in AWS Control Tower has unique Amazon Resource Names (ARNs) for use with the AWS Control Tower and Control Catalog APIs. You can call an API using a global ARN or a regional ARN.

- We recommend that you use the global ARNs for all use cases.
- The regional ARNs have been available longer but are considered as legacy ARNs.

Understand control ARNs

AWS Control Tower supports two types of control ARNs.

Global ARNs (recommended)

A global ARN is available for all controls that are part of Control Catalog. Global ARNs use the following format.

```
arn:{PARTITION}:controlcatalog:::control/{GLOBAL_CONTROL_ID}
```

For example, `arn:aws:controlcatalog:::control/k4izcjxhukijhajp6ks5mjxk`. Global ARNs are independent of any AWS Region. We recommend that you use global ARNs for all use cases.

Regional ARNs (legacy)

Older Control Catalog controls also have regional ARNs. A regional ARN is a unique identifier for each Region in which AWS Control Tower operates. Regional ARNs use the following format.

```
arn:{PARTITION}:controltower:{REGION}::control/{REGIONAL_CONTROL_ID}
```

For example, `arn:aws:controltower:us-east-1::control/YEHYWYAUIQHZ`. Regional ARNs for the same control can be different in different Regions.

Benefits of using global ARNs

Global ARNs provide several advantages over Regional ARNs.

- **Region independence** – The same ARN works across all AWS Regions within the same partition.

- **Simplified management** – You don't need to maintain Region-specific identifiers. This simplifies multi-Region management and deployments.
- **Future-proof** – New controls are assigned global ARNs only.

Find global ARNs

You can retrieve global ARNs for Control Catalog controls in the following ways.

AWS Control Tower console

To view the global ARNs and other details about Control Catalog controls in the console, navigate to the **Control details** page in the AWS Control Tower console. You can find the identifier in the **API identifier** field.

ListControls API

You can use the ListControls API to retrieve all controls with their global ARNs.

Migrate from regional ARNs to global ARNs

Regional ARNs are no longer displayed in the console or the documentation, in favor of global ARNs. If you're using existing regional ARNs in your automation, you can continue to use them with the AWS Control Tower and Control Catalog APIs. However, we recommend that you migrate to global ARNs.

Retrieve global ARNs by using regional control ARNs

You can use regional ARNs with the GetControl API to retrieve control metadata and global ARNs. For example:

```
aws controlcatalog get-control --control-arn arn:aws:controltower:us-east-1::control/
YEHYWYAUIQHZ --region us-east-1
```

The response includes the corresponding global ARN, but not the Regional ARN.

Find identifiers for OUs

For more information about how to find the resource identifier for an OU and its resources, see Resource types defined by AWS Organizations.

To learn more about how to get information from an OU, see the AWS Organizations API Reference.

Identifiers for legacy controls

The following section contains the Regional `API controlIdentifier` designations of the legacy **Strongly recommended** and **Elective**, *preventive* and *detective*, controls that are owned by AWS Control Tower, including the elective **Data residency** controls. This information is presented as a reference. Although we recommend that you call APIs using the global identifiers, some controls may have been activated with Regional identifiers and still can be tracked by them.

> **Note**
>
> Mandatory controls cannot be deactivated by the control APIs.

Each item in the list that follows serves as a link, which provides more information about these individual (legacy) controls that are owned by AWS Control Tower, as given in The AWS Control Tower Control Catalog.

Designations for legacy Elective controls

- arn:aws:controltower:REGION::control/AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED
- arn:aws:controltower:REGION::control/AWS-GR_AUDIT_BUCKET_LOGGING_ENABLED
- arn:aws:controltower:REGION::control/AWS-GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED
- arn:aws:controltower:REGION::control/AWS-GR_AUDIT_BUCKET_RETENTION_POLICY
- arn:aws:controltower:REGION::control/AWS-GR_IAM_USER_MFA_ENABLED
- arn:aws:controltower:REGION::control/AWS-GR_MFA_ENABLED_FOR_IAM_CONSOLE_ACCESS
- arn:aws:controltower:REGION::control/AWS-GR_RESTRICT_S3_CROSS_REGION_REPLICATION
- arn:aws:controltower:REGION::control/AWS-GR_RESTRICT_S3_DELETE_WITHOUT_MFA
- arn:aws:controltower:REGION::control/AWS-GR_S3_VERSIONING_ENABLED

Designations for legacy Data residency controls (elective)

- arn:aws:controltower:REGION::control/AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED

- arn:aws:controltower:REGION::control/AWS-GR_AUTOSCALING_LAUNCH_CONFIG_PUBLIC_IP_DISABLED
- arn:aws:controltower:REGION::control/AWS-GR_DISALLOW_CROSS_REGION_NETWORKING
- arn:aws:controltower:REGION::control/AWS-GR_DISALLOW_VPC_INTERNET_ACCESS
- arn:aws:controltower:REGION::control/AWS-GR_DISALLOW_VPN_CONNECTIONS
- arn:aws:controltower:REGION::control/AWS-GR_DMS_REPLICATION_NOT_PUBLIC
- arn:aws:controltower:REGION::control/AWS-GR_EBS_SNAPSHOT_PUBLIC_RESTORABLE_CHECK
- arn:aws:controltower:REGION::control/AWS-GR_EC2_INSTANCE_NO_PUBLIC_IP
- arn:aws:controltower:REGION::control/AWS-GR_EKS_ENDPOINT_NO_PUBLIC_ACCESS
- arn:aws:controltower:REGION::control/AWS-GR_ELASTICSEARCH_IN_VPC_ONLY
- arn:aws:controltower:REGION::control/AWS-GR_EMR_MASTER_NO_PUBLIC_IP
- arn:aws:controltower:REGION::control/AWS-GR_LAMBDA_FUNCTION_PUBLIC_ACCESS_PROHIBITED
- arn:aws:controltower:REGION::control/AWS-GR_NO_UNRESTRICTED_ROUTE_TO_IGW
- arn:aws:controltower:REGION::control/AWS-GR_REDSHIFT_CLUSTER_PUBLIC_ACCESS_CHECK
- arn:aws:controltower:REGION::control/AWS-GR_S3_ACCOUNT_LEVEL_PUBLIC_ACCESS_BLOCKS_PERIODIC
- arn:aws:controltower:REGION::control/AWS-GR_SAGEMAKER_NOTEBOOK_NO_DIRECT_INTERNET_ACCESS
- arn:aws:controltower:REGION::control/AWS-GR_SSM_DOCUMENT_NOT_PUBLIC

Designations for legacy Strongly recommended controls

- arn:aws:controltower:REGION::control/AWS-GR_ENCRYPTED_VOLUMES
- arn:aws:controltower:REGION::control/AWS-GR_EBS_OPTIMIZED_INSTANCE
- arn:aws:controltower:REGION::control/AWS-GR_EC2_VOLUME_INUSE_CHECK
- arn:aws:controltower:REGION::control/AWS-GR_RDS_INSTANCE_PUBLIC_ACCESS_CHECK
- arn:aws:controltower:REGION::control/AWS-GR_RDS_SNAPSHOTS_PUBLIC_PROHIBITED
- arn:aws:controltower:REGION::control/AWS-GR_RDS_STORAGE_ENCRYPTED
- arn:aws:controltower:REGION::control/AWS-GR_RESTRICTED_COMMON_PORTS
- arn:aws:controltower:REGION::control/AWS-GR_RESTRICTED_SSH
- arn:aws:controltower:REGION::control/AWS-GR_RESTRICT_ROOT_USER

- arn:aws:controltower:REGION::control/AWS-GR_RESTRICT_ROOT_USER_ACCESS_KEYS
- arn:aws:controltower:REGION::control/AWS-GR_ROOT_ACCOUNT_MFA_ENABLED
- arn:aws:controltower:REGION::control/AWS-GR_S3_BUCKET_PUBLIC_READ_PROHIBITED
- arn:aws:controltower:REGION::control/AWS-GR_S3_BUCKET_PUBLIC_WRITE_PROHIBITED
- arn:aws:controltower:REGION::control/AWS-GR_DETECT_CLOUDTRAIL_ENABLED_ON_MEMBER_ACCOUNTS

Controls that cannot be changed with the AWS Control Tower APIs

The following controls cannot be activated or deactivated by means of the AWS Control Tower APIs. Except for the landing zone Region deny control, all of these are mandatory controls. In general, mandatory controls cannot be deactivated. The landing zone Region deny control must be changed in the console.

- AWS-GR_REGION_DENY (Landing zone Region deny control)
- AWS-GR_AUDIT_BUCKET_DELETION_PROHIBITED
- AWS-GR_AUDIT_BUCKET_PUBLIC_READ_PROHIBITED
- AWS-GR_AUDIT_BUCKET_PUBLIC_WRITE_PROHIBITED
- AWS-GR_CLOUDTRAIL_CHANGE_PROHIBITED
- AWS-GR_CLOUDTRAIL_CLOUDWATCH_LOGS_ENABLED
- AWS-GR_CLOUDTRAIL_ENABLED
- AWS-GR_CLOUDTRAIL_VALIDATION_ENABLED
- AWS-GR_CLOUDWATCH_EVENTS_CHANGE_PROHIBITED
- AWS-GR_CONFIG_AGGREGATION_AUTHORIZATION_POLICY
- AWS-GR_CONFIG_AGGREGATION_CHANGE_PROHIBITED
- AWS-GR_CONFIG_CHANGE_PROHIBITED
- AWS-GR_CONFIG_ENABLED
- AWS-GR_CONFIG_RULE_CHANGE_PROHIBITED
- AWS-GR_CT_AUDIT_BUCKET_ENCRYPTION_CHANGES_PROHIBITED
- AWS-GR_CT_AUDIT_BUCKET_LIFECYCLE_CONFIGURATION_CHANGES_PROHIBITED
- AWS-GR_CT_AUDIT_BUCKET_LOGGING_CONFIGURATION_CHANGES_PROHIBITED

- AWS-GR_CT_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED
- AWS-GR_IAM_ROLE_CHANGE_PROHIBITED
- AWS-GR_LAMBDA_CHANGE_PROHIBITED
- AWS-GR_LOG_GROUP_POLICY
- AWS-GR_SNS_CHANGE_PROHIBITED
- AWS-GR_SNS_SUBSCRIPTION_CHANGE_PROHIBITED
- AWS-GR_ENSURE_CLOUDTRAIL_ENABLED_ON_SHARED_ACCOUNTS
- CT.S3.PV.7
- CT.S3.PV.8
- CT.SNS.PV.1

Control API examples

Each control in AWS Control Tower has a unique identifier for use with the control APIs. The identifier for each control is shown in the **API controlIdentifier** field, on the **Control details** page in the AWS Control Tower console. This identifier is distinct from the **ControlID** field, which is a classification system for controls.

> **Note**
>
> When you invoke `EnableControl` on an account or OU, the `operationIdentifier` value is returned by means of `ListEnabledControls` or `GetEnabledControl` even if the enable operation fails. In the AWS Control Tower console, you can determine whether the `EnableControl` operation was successful, by verifying that the control is enabled on the account or OU. Programatically, you can track the status of the `EnableControl` operation with the `GetControlOperation` API command, by passing it the value of `operationIdentifier` as shown in an example that follows.

EnableControl

For more information about this API operation, see EnableControl.

Example input for EnableControl:

This example shows how to specify the control you wish to enable, and activate that control for the target OU that you identify.

```
{
   controlIdentifier: "arn:aws:controltower:us-west-2::control/AWS-
GR_AUTOSCALING_LAUNCH_CONFIG_PUBLIC_IP_DISABLED",
   targetIdentifier: "arn:aws:organizations::123456789123:ou/o-kg8aXXXXXX/ou-prlj-
a5kXXXXX"
}
```

Example output for EnableControl:

As an example of how to use this output parameter, you can pass the **operationIdentifier** parameter as an input to the **GetControlOperation** API, to track the status of your **EnableControl** task.

```
{
    "operationIdentifier":"e2bXXXXX-6cab-XXXX-bde7-XX0c6fXXXXXX"
}
```

Example CLI command:

```
aws controltower enable-control \
--control-identifier arn:aws:controltower:us-west-2::control/AWS-
GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED \
--target-identifier arn:aws:organizations::123456789123:ou/o-qnilXXXXXX/ou-vwxu-
qqlXXXXX \
--region us-west-2
```

DisableControl

For more information about this API operation, see DisableControl.

Example input for DisableControl:

```
{
    controlIdentifier: "arn:aws:controltower:us-west-2::control/AWS-
GR_AUTOSCALING_LAUNCH_CONFIG_PUBLIC_IP_DISABLED",
    targetIdentifier: "arn:aws:organizations::123456789123:ou/o-kg8aXXXXXX/ou-prlj-
a5kXXXXX"
}
```

Example output for DisableControl:

```
{
    "operationIdentifier":"e2bXXXXX-8xai-XXXX-bde7-XX0c6fXXXXXX"
}
```

GetControlOperation

For more information about this API operation, see GetControlOperation.

Example input for GetControlOperation:

When you give an **operationIdentifier** as input, you receive a status message as output.

```
{
    operationIdentifier: "e2bXXXXX-6cab-XXXX-bde7-XX0c6fXXXXXX"
}
```

Example output for GetControlOperation:

```
{
    "ControlOperationStatus":{
        "OperationType": "ENABLE_CONTROL",
        "StartTime": "2022-02-02T20:52:08.034Z",
        "Status": "IN_PROGRESS"
    }
}
```

Example output for GetControlOperation:

```
{
    "ControlOperationStatus": {
        "EndTime": "2022-04-28T19:36:31Z",
        "OperationType": "DISABLE_CONTROL",
        "StartTime": "2022-04-28T19:35:00Z",
        "Status": "SUCCEEDED"
    }
}
```

```
{
    "ControlOperationStatus": {
        "EndTime": "2022-04-28T19:36:31Z",
        "OperationType": "DISABLE_CONTROL",
        "StartTime": "2022-04-28T19:35:00Z",
        "Status": "FAILED",
        "StatusMessage": "AWS Control Tower cannot add the SCP because the IAM
 user or role does not have permission to perform the requested operation in AWS
 Organizations. To continue, update your access permissions for AWS Organizations. For
 more information, see Access Management in the IAM User Guide."
    }
}
```

GetEnabledControl

For more information about this API operation, see GetEnabledControl.

Example for GetEnabledControl

```
aws controltower get-enabled-control --enabled-control-identifier
 arn:aws:controltower:us-east-1:123456789012:enabledcontrol/49DVF3KP34ANNC57{
    "enabledControlDetails": {
        "arn": "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/49DVF3KP34ANNC57",
        "controlIdentifier": "arn:aws:controltower:us-east-1::control/AWS-
GR_EBS_OPTIMIZED_INSTANCE",
        "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-ct7amci1en/ou-
slfp-nay7ybhu",
        "targetRegions": [
            {
                "name": "eu-north-1"
            },
            {
                "name": "eu-west-2"
            }
        ],
        "statusSummary": {
            "status": "SUCCEEDED",
            "lastOperationIdentifier": "12e51344-a73a-439a-8477-fb3cd7f8b410"
        },
        "driftStatusSummary": {
            "driftStatus": "NOT_CHECKING"
        }
    }
}
```

ListControlOperations

For more information about this API operation, see ListControlOperations.

Example input and output for ListControlOperations:

```
aws controltower list-control-operations --max-items 13
```

```
{
    "controlOperations": [
        {
            "startTime": "2024-02-19T19:22:08+00:00",
            "operationType": "ENABLE_CONTROL",
```

```
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "f9f43b45-db27-44df-89d8-f9129e3632XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
SKIBWKYUQAAC",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/RWZFSHV2BBRU6JSE"
        },
        {
            "startTime": "2024-02-19T19:21:09+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress."
            "operationIdentifier": "171ee0b1-e926-486e-9775-005bd244ccXX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
PDKYAANJEWJE",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-fl6miXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/XCNJARWZFSHV6JSE"
        },
        {
            "startTime": "2024-02-19T19:20:08+00:00",
            "operationType": "DISABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "6345643b-3bd1-44dc-b1e5-9e9ae5df41XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
XAZHJTQBXMLM",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-quinbXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/NCHASYHFBJNEBFU"
        },
        {
            "startTime": "2024-02-19T19:19:12+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "a6d26135-7234-409c-ace8-7a0020996bXX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
ELALMJSUVZGW",
```

```
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/ANDPATBCJALUFNXY"
        },
        {
            "startTime": "2024-02-19T19:18:08+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "8fdd8688-e6ef-4def-9b7f-820b2393f1XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
LUZSILPCBBOK",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/RBFOAMNVHYAZBCTG"
        },
        {
            "startTime": "2024-02-19T19:17:20+00:00",
            "operationType": "DISABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "d01e0afc-722b-4762-bb93-afece269feXX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/DVIYOYZQQBH",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/MXBDYAQPLCGATSDI"
        },
        {
            "startTime": "2024-02-19T19:16:08+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "a2e7cc66-a937-4b76-96e5-87e6e5c4a7XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
EVWTVVCSKQMU",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-quinbXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/PMANDUTVCBHDKLXY"
        },
        {
```

```
            "startTime": "2024-02-19T19:15:21+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "53718bd4-1445-417a-94fa-53c5cc0cf3XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_EBS_OPTIMIZED_INSTANCE",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-fl6miXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/MCBAYDFGXBVAQTSP"
        },
        {
            "startTime": "2024-02-19T19:14:08+00:00",
            "operationType": "DISABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "33b1b0d7-508c-4159-a18c-48dd692f99XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_EBS_SNAPSHOT_PUBLIC_RESTORABLE_CHECK",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/KDGCVAZNJHFKDNCO"
        },
        {
            "startTime": "2024-02-19T19:13:09+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "IN_PROGRESS",
            "statusMessage": "Operation is in progress.",
            "operationIdentifier": "2f9e3c21-fc5f-4606-b64a-a5e63814f0XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_EC2_INSTANCE_NO_PUBLIC_IP",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-quinbXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/DSRTFJNCMLDIFTAY"
        },
        {
            "startTime": "2024-02-19T19:12:38+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "SUCCEEDED",
            "statusMessage": "Operation was successful.",
            "operationIdentifier": "d808bd59-96b0-42cc-b2fd-b5f4069adcXX",
```

```
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_EC2_VOLUME_INUSE_CHECK",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/VBYGOPDHNAGDTRWP"
        },
        {
            "startTime": "2024-02-19T19:11:08+00:00",
            "operationType": "DISABLE_CONTROL",
            "status": "SUCCEEDED",
            "statusMessage": "Operation was successful.",
            "operationIdentifier": "5decdb1e-79f7-4fe5-b1a6-b4736f1780XX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_NO_UNRESTRICTED_ROUTE_TO_IGW",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-fl6miXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/ZLPEBCHAYDGHJEYC"
        },
        {
            "startTime": "2024-02-19T19:10:11+00:00",
            "operationType": "ENABLE_CONTROL",
            "status": "FAILED",
            "statusMessage": "AWS Control Tower cannot add the SCP because the IAM
 user or role does not have permission to perform the requested operation in AWS
 Organizations. To continue, update your access permissions for AWS Organizations. For
 more information, see Access Management in the IAM User Guide."
            "operationIdentifier": "a2629bd6-8777-40b2-9998-5d89dd37dcXX",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
TKCJNPHIDFHI",
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-yy67i3pfv2/
ou-slt4-8abknXXX",
            "enabledControlIdentifier": "arn:aws:controltower:us-
west-2:123456789012:enabledcontrol/NCBDYANCODHAZBCP"
        }
    ],
    "NextToken": "eyJuZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ=="
}
```


ListEnabledControls

For more information about this API operation, see ListEnabledControls.

Example input for ListEnabledControls:

This example shows how to specify the target OU as input, so you can receive a list of controls as output.

```
{
    targetIdentifier: "arn:aws:organizations::123456789123:ou/o-kg8aXXXXXX/ou-prlj-
a5kXXXXX",
    nextToken: "bde7-XX0c6fXXXXXX",
    maxResults: 2
}
```

Example output for ListEnabledControls:

```
{
    "enabledControls": [
        {
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_AUTOSCALING_LAUNCH_CONFIG_PUBLIC_IP_DISABLED"
        },
        {
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/AWS-
GR_RESTRICT_ROOT_USER"
        }
    ],
    "nextToken": "e2bXXXXX-6cab-XXXX"
}
```

This example shows a larger set of returned values for ListEnabledControls.

```
aws controltower list-enabled-controls --target-identifier
 arn:aws:organizations::072569612342:ou/o-yy67i3pfv2/ou-slt4-fl6mi3bd --max-items 3
{
    "enabledControls": [
        {
            "arn": "arn:aws:controltower:us-west-2::enabledcontrol/
SOME_ENABLED_CONTROL",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
SOME_CONTROL",
            "targetIdentifier": "arn:aws:organizations::072569612342:ou/o-yy67i3pfv2/
ou-slt4-fl6mi3bd",
            "statusSummary": {
                "status": "SUCCEEDED",
```

```
                "lastOperationIdentifier": "12e51344-a73a-439a-8477-fb3cd7f8b410"
            },
            "driftStatusSummary": {
                "driftStatus": "NOT_CHECKING"
            }
        },
        {
            "arn": "arn:aws:controltower:us-west-2::enabledcontrol/
OTHER_ENABLED_CONTROL",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
OTHER_CONTROL",
            "targetIdentifier": "arn:aws:organizations::072569612342:ou/o-yy67i3pfv2/
ou-slt4-fl6mi3bd",
            "statusSummary": {
                "status": "FAILED",
                "lastOperationIdentifier": "12e51344-a73a-439a-8477-fb3cd7f8b410"
            },
            "driftStatusSummary": {
                "driftStatus": "UNKNOWN"
            }
        },
        {
            "arn": "arn:aws:controltower:us-west-2::enabledcontrol/
ANOTHER_ENABLED_CONTROL",
            "controlIdentifier": "arn:aws:controltower:us-west-2::control/
ANOTHER_CONTROL",
            "targetIdentifier": "arn:aws:organizations::072569612342:ou/o-yy67i3pfv2/
ou-slt4-fl6mi3bd",
            "statusSummary": {
                "status": "SUCCEEDED",
                "lastOperationIdentifier": "12e51344-a73a-439a-8477-fb3cd7f8b410"
            },
            "driftStatusSummary": {
                "driftStatus": "IN_SYNC"
            }
        }
    ],
    "nextToken": "eyJuZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```


ListTagsForResource

For more information about this API operation, see ListTagsForResource.

Example for ListTagsForResource

```
aws controltower list-tags-for-resource --resource-arn "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/49DVF3KP34ANNC57"
{
  "TestTagKey": "TestTagValue"
}
```

ResetEnabledControl

For more information about this API operation, see ResetEnabledControl.

Example for ResetEnabledControl

```
aws controltower reset-enabled-control \
    –enabled-control-identifier arn:aws:controltower:us-
east-1:01234567890:enabledcontrol/EXAMPLE_NAME
```

TagResource

For more information about this API operation, see TagResource.

Example for TagResource

```
aws controltower tag-resource --resource-arn "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/49DVF3KP34ANNC57"} --tags "TestTagKey=TestTagValue"
{
}
```

UntagResource

For more information about this API operation, see UntagResource.

Example for UntagResource

```
aws controltower untag-resource --resource-arn "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/49DVF3KP34ANNC57" --tag-keys "TestTagKey"
{
```

```
}
```

UpdateEnabledControl

For more information about this API operation, see UpdateEnabledControl.

Change the parameters of a control:

```
aws controltower update-enabled-control \
    --enabled-control-identifier arn:aws:controltower:us-
east-1:01234567890:enabledcontrol/EXAMPLE_NAME \
    --parameters '[{"key":"AllowedRegions","value":["us-east-1","us-west-1","us-
west-2","us-east-2"]},{"key":"ExemptedPrincipalArns","value":["arn:aws:iam::*:role/
ReadOnly","arn:aws:sts::*:assumed-role/ReadOnly/*"]},{"key":"ExemptedActions","value":
["logs:DescribeLogGroups","logs:StartQuery","logs:GetQueryResults","cloudwatch:Get*","cloudwatc
```

Here's a more readable version of parameters input:

```
[
    {
        "key": "AllowedRegions",
        "value":
        [
            "us-east-1",
            "us-west-1",
            "us-west-2",
            "us-east-2"
        ]
    },
    {
        "key": "ExemptedPrincipalArns",
        "value":
        [
            "arn:aws:iam::*:role/ReadOnly",
            "arn:aws:sts::*:assumed-role/ReadOnly/*"
        ]
    },
    {
        "key": "ExemptedActions",
        "value":
        [
            "logs:DescribeLogGroups",
```

```
            "logs:StartQuery",
            "logs:GetQueryResults",
            "cloudwatch:Get*",
            "cloudwatch:Describe*"
        ]
    }
]
```

View parameters

You can view the existing parameters for a control with the `GetEnabledControl` API call.

Example input:

```
aws controltower get-enabled-control --enabled-control-identifier
 arn:aws:controltower:us-east-1:01234567890:enabledcontrol/EXAMPLE_NAME \
```

Example output:

```
{
    "enabledControlDetails": {
        "arn": "arn:aws:controltower:us-east-1:01234567890:enabledcontrol/
EXAMPLE_NAME",
        "controlIdentifier": "arn:aws:controltower:us-east-1::control/EXAMPLE_NAME",
        "targetIdentifier": "arn:aws:organizations::01234567890:ou/o-EXAMPLE/ou-zzxx-
zzx0zzz2",
        ...
        ...
        ...
        "parameters": [
            {
                "key": "ExemptedPrincipalArns",
                "value": [
                    "arn:aws:iam::*:role/ReadOnly"
                ]
            },
            {
                "key": "AllowedRegions",
                "value": [
                    "us-east-1",
                    "us-west-1"
                ]
```

```
            },
            {
                "key": "ExemptedActions",
                "value": [
                    "logs:DescribeLogGroups",
                    "logs:StartQuery",
                    "logs:GetQueryResults"
                ]
            }
        ]
    }
}
```

For examples of how to work with the AWS Control Tower baseline APIs, see Examples for baseline API usage.

Enable controls with CloudFormation

You can enable controls with CloudFormation, either through the CloudFormation console, or through the AWS CLI. This section gives an example of each type.

Each control in AWS Control Tower has a unique identifier for use with the control APIs. The identifier for each control is shown in the **API controlIdentifier** field, on the **Control details** page in the AWS Control Tower console. This identifier is distinct from the **ControlID** field, which is a classification system for controls.

For more information about control identifiers, see Resource identifiers for APIs and controls.

Create the stack through CloudFormation

You can use CloudFormation to help you enable AWS Control Tower controls. Here's an example template.

```
Resources:
    TestControl:
        Type: AWS::ControlTower::EnabledControl
        Properties:
            ControlIdentifier: arn:aws:controltower:us-west-2::control/AWS-
GR_RESTRICT_ROOT_USER
            TargetIdentifier: arn:aws:organizations::123456789012:ou/o-ybfpt9XXXl/ou-
XXXc-nlqXXXXX
```

To create your stack through the CloudFormation console, edit the template to contain the control and target of your choice, then save the template with the file name `template.yaml`. Follow the CloudFormation wizard. When the wizard asks for a template file, enter the file you saved as `template.yaml`. For more information, see Creating a stack on the Amazon CloudFormation console.

> **Note**
>
> The limit for `EnableControl` and `DisableControl` updates in AWS Control Tower is 100 concurrent operations.

Create the stack through CloudFormation and the AWS CLI

Here's an example of creating the stack with the CLI.

```
aws cloudformation create-stack --region us-west-2 --stack-name testControlTower --
template-body "$(cat << TEMPLATE
Resources:
  TestControl:
    Type: AWS::ControlTower::EnabledControl
    Properties:
      ControlIdentifier: arn:aws:controltower:us-west-2::control/AWS-
GR_RESTRICT_ROOT_USER
      TargetIdentifier: arn:aws:organizations::123456789012:ou/o-ybfpt9XXXl/ou-XXXc-
nlqXXXXX
TEMPLATE)"
```

You can also save the example template as a `template.yaml` file, then upload your file to an Amazon S3 bucket. Later, you can provide the URL for the bucket with the `--template-url` flag.

When you enter your template into the wizard or through the CLI, if the stack is created, it means that the control was enabled.

View the progress of your stack through the AWS CLI:

```
aws cloudformation describe-stack-events --region us-west-2 --stack-name
 testControlTower
```

or

```
aws cloudformation describe-stacks --region us-west-2 --stack-name testControlTower
```

Delete the stack through the AWS CLI:

```
aws cloudformation delete-stack --region us-west-2 --stack-name testControlTower
```

Configure controls with CloudFormation

The following examples show how to configure controls through CloudFormation templates. These examples happen to show **Value** as a list, but it can be any of several types.

Enable configurable controls with CloudFormation templates

Enable a control with parameters through CloudFormation:

```
aws cloudformation create-stack \
    --stack-name ExampleStack \
    --template-body file://ExampleStack.yml \
    --region us-east-1
```

Example templates in YAML and JSON:

```
Resources:
  MyExampleControl:
    Properties:
      ControlIdentifier: arn:aws:controltower:us-east-1::control/EXAMPLE_NAME
      TargetIdentifier: arn:aws:organizations::01234567890:ou/o-EXAMPLE/ou-zzxx-
zzx0zzz2
      Parameters:
      - Key: AllowedRegions
        Value:
        - us-east-1
        - us-west-1
      - Key: ExemptedPrincipalArns
        Value:
        - arn:aws:iam::*:role/ReadOnly
      - Key: ExemptedActions
        Value:
        - logs:DescribeLogGroups
        - logs:StartQuery
        - logs:GetQueryResults
    Type: AWS::ControlTower::EnabledControl
```

```
{
  "Resources": {
    "MyExampleControl": {
      "Type": "AWS::ControlTower::EnabledControl",
      "Properties": {
        "TargetIdentifier": "arn:aws:organizations::01234567890:ou/o-EXAMPLE/ou-zzxx-
zzx0zzz2",
        "ControlIdentifier": "arn:aws:controltower:us-east-1::control/EXAMPLE_NAME",
```

```
        "Parameters": [
          {
            "Key": "AllowedRegions",
            "Value": [
              "us-east-1",
              "us-west-1"
            ]
          },
          {
            "Key": "ExemptedPrincipalArns",
            "Value": [
              "arn:aws:iam::*:role/ReadOnly"
            ]
          },
          {
            "Key": "ExemptedActions",
            "Value": [
              "logs:DescribeLogGroups",
              "logs:StartQuery",
              "logs:GetQueryResults"
            ]
          }
        ]
      }
    }
  }
}
```

The AWS Control Tower Control Catalog

The following sections include an individual reference entry for each of the controls available in AWS Control Tower. The controls are grouped into sections according to common characteristics. Each control reference entry includes the details, artifacts, additional information, and considerations to keep in mind when enabling a specific control on a OU in your landing zone.

> **Note**
>
> The Control Catalog was formerly called the Controls Library. We have renamed it for consistency.

How to view controls

- To retrieve information about individual controls programmatically, call the `GetControl` API from the *controlcatalog* namespace of AWS Control Tower.
- To retrieve a list of available controls programmatically, call the `ListControls` API from the *controlcatalog* namespace of AWS Control Tower.
- Additional detail about each control is available in the AWS Control Tower console, on the **Control details** pages and Control Catalog APIs. For more information, see View control details.
- To understand control ARNs, see Resource identifiers for APIs and controls.

Topics

- Mandatory controls
- Proactive controls
- Preventive controls
- Detective controls
- Controls with parameters
- Optional controls
- Strongly recommended controls
- Elective controls
- Search for controls with AWS assistant

> **Note**
>
> The four mandatory controls with `"Sid": "GRCLOUDTRAILENABLED"` are identical by design. The sample code is correct.

Mandatory controls

Mandatory controls are owned by AWS Control Tower, and they apply to every OU on your landing zone, with some exceptions for the **Security OU**. These controls are applied to your OUs by default when you set up your landing zone, and they can't be deactivated, because they protect AWS Control Tower resources.

Following, you'll find a reference for each of the mandatory controls available in AWS Control Tower today. Please note that with AWS Control Tower Landing Zone 4.0, there have been several changes to the mandatory controls.

Topics

- Changes in Landing Zone 4.0 controls
- Mandatory controls for the Security OU
- Mandatory controls for all OUs

Changes in Landing Zone 4.0 controls

- AWS Control Tower will no longer deploy the below controls from landing zone 4.0 because these protect the account trails which were deployed in landing zone 2.9 and below. These controls would still be applied for landing zone versions below 4.0 and would be deleted once customers upgrade to versions 4.0 and above.
  - Disallow Configuration Changes to CloudTrailDisallow Configuration Changes to CloudTrail
  - Integrate CloudTrail Events with Amazon CloudWatch LogsIntegrate CloudTrail Events with Amazon CloudWatch Logs
  - Enable CloudTrail in All Available RegionsEnable CloudTrail in All Available Regions
  - Enable Integrity Validation for CloudTrail Log FileEnable Integrity Validation for CloudTrail Log File
- The following detective mandatory control will be removed:

  - Detect whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabledDetect whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabled
- The following controls will be removed if the customer has AWS Config integration enabled and are upgrading to landing zone 4.0 and above (Note: Customers on landing zone versions below 4.0 have AWS Config integration enabled by default). These controls are related to legacy AWS Config aggregators and are no longer required for Service-Linked Config Aggregator. Read more on the Service-linked Config aggregator here.
  - Disallow Changes to Tags Created by AWS Control Tower for AWS Config Resources
  - Disallow Deletion of AWS Config Aggregation Authorizations Created by AWS Control Tower
- The AWS CloudTrail integration tied to the manifest `centralizedLogging` configuration has two new controls starting landing zone 4.0
  - Disallow changes to Amazon SNS subscriptions and topics managed by AWS Control Tower

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTSNSPV1",
            "Effect": "Deny",
            "NotAction": [
                "sns:ConfirmSubscription",
                "sns:Get*",
                "sns:List*",
                "sns:Publish",
                "sns:PutDataProtectionPolicy",
                "sns:Subscribe",
                "sns:TagResource",
                "sns:Unsubscribe",
                "sns:UntagResource"
            ],
            "Resource": "arn:*:sns:*:*:aws-controltower-
CentralizedLoggingNotifications*",
            "Condition": {
                 "ArnNotLike": {
                        "aws:PrincipalARN": [
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
```

```
            }
        }
    ]
}
```

- Disallow modifications to Amazon S3 buckets managed by AWS Control Tower

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV8",
            "Effect": "Deny",
            "NotAction": [
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:Get*",
                "s3:List*",
                "s3:PutBucketTagging",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": [
                "arn:*:s3:::aws-controltower-access-logs-*",
                "arn:*:s3:::aws-controltower-cloudtrail-*",
                "arn:*:s3:::aws-controltower-logs-*"
            ],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
```

```
        }
    ]
}
```

- The AWS Config integration has a new control starting landing zone 4.0
  - Disallow modifications to AWS Config recorder Amazon S3 buckets managed by AWS Control Tower

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "CTS3PV7",
             "Effect": "Deny",
             "NotAction": [
                 "s3:DeleteObject",
                 "s3:DeleteObjectTagging",
                        "s3:DeleteObjectVersion",
                        "s3:DeleteObjectVersionTagging",
                        "s3:Get*",
                        "s3:List*",
                        "s3:PutBucketTagging",
                        "s3:PutObject",
                        "s3:PutObjectAcl",
                        "s3:PutObjectLegalHold",
                        "s3:PutObjectRetention",
                        "s3:PutObjectTagging",
                        "s3:PutObjectVersionAcl",
                        "s3:PutObjectVersionTagging",
                        "s3:RestoreObject"
             ],
             "Resource": "arn:*:s3:::aws-controltower-config-*",
             "Condition": {
                 "ArnNotLike": {
                     "aws:PrincipalARN": [
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
```

```
}
```

- On landing zone 4.0, AWS Control Tower will disable the following controls as they are replaced with a single unified preventive control for AWS Config integration. The security governance boundary remains the same, but with reduced SCP space.
  - Disallow Changes to Encryption Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
  - Disallow Changes to Logging Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
  - Disallow Changes to Bucket Policy for AWS Control Tower Created Amazon S3 Buckets in Log Archive
  - Disallow Changes to Lifecycle Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
  - Disallow Deletion of Log Archive
  - Detect Public Write Access Setting for Log Archive
  - Detect Public Read Access Setting for Log Archive
- AWS Control Tower is updating the following controls for all landing zone versions, this change will take place when customers update/reset their existing setup.
  - Disallow Changes to Amazon SNS Set Up by AWS Control Tower

    The change is to specify three explicit SNS topic ARNs in the Resource section instead of using a wildcard pattern

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "GRSNSTOPICPOLICY",
             "Effect": "Deny",
             "Action": [
                 "sns:AddPermission",
                 "sns:CreateTopic",
                 "sns:DeleteTopic",
                 "sns:RemovePermission",
                 "sns:SetTopicAttributes"
             ],
```

```
            "Resource": [
                "arn:*:sns:*:*:aws-controltower-AggregateSecurityNotifications*",
                "arn:*:sns:*:*:aws-controltower-AllConfigNotifications*",
                "arn:*:sns:*:*:aws-controltower-SecurityNotifications*"
            ],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": "arn:*:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Mandatory controls for the Security OU

When you set up your landing zone, the following eight controls are applied only to the Security OU.

Topics

- Disallow Changes to Encryption Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
- Disallow Changes to Logging Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
- Disallow Changes to Bucket Policy for AWS Control Tower Created Amazon S3 Buckets in Log Archive
- Disallow Changes to Lifecycle Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive
- Disallow Deletion of Log Archive
- Detect Public Read Access Setting for Log Archive
- Detect Public Write Access Setting for Log Archive
- Detect whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabled

Disallow Changes to Encryption Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive

This control prevents changes to encryption for the Amazon S3 buckets that AWS Control Tower creates in the log archive account. This is a preventive control with mandatory guidance. By default, this control is enabled on the **Security OU**. It cannot be enabled on additional OUs.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCTAUDITBUCKETENCRYPTIONCHANGESPROHIBITED",
            "Effect": "Deny",
            "Action": [
                "s3:PutEncryptionConfiguration"
            ],
            "Resource": ["arn:aws:s3:::aws-controltower*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Changes to Logging Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive

This control prevents changes to logging configuration for the Amazon S3 buckets that AWS Control Tower creates in the log archive account. This is a preventive control with mandatory guidance. By default, this control is enabled on the **Security OU**. It cannot be enabled on additional OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCTAUDITBUCKETLOGGINGCONFIGURATIONCHANGESPROHIBITED",
            "Effect": "Deny",
            "Action": [
                "s3:PutBucketLogging"
            ],
            "Resource": ["arn:aws:s3:::aws-controltower*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Changes to Bucket Policy for AWS Control Tower Created Amazon S3 Buckets in Log Archive

This control prevents changes to bucket policy for the Amazon S3 buckets that AWS Control Tower creates in the log archive account. This is a preventive control with mandatory guidance. By default, this control is enabled on the **Security OU**. It cannot be enabled on additional OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "GRCTAUDITBUCKETPOLICYCHANGESPROHIBITED",
            "Effect": "Deny",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:DeleteBucketPolicy"
            ],
            "Resource": ["arn:aws:s3:::aws-controltower*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Changes to Lifecycle Configuration for AWS Control Tower Created Amazon S3 Buckets in Log Archive

This control prevents lifecycle configuration changes for the Amazon S3 buckets that AWS Control Tower creates in the log archive account. This is a preventive control with mandatory guidance. By default, this control is enabled on the **Security OU**. It cannot be enabled on additional OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCTAUDITBUCKETLIFECYCLECONFIGURATIONCHANGESPROHIBITED",
            "Effect": "Deny",
            "Action": [
                "s3:PutLifecycleConfiguration"
            ],
```

```
            "Resource": ["arn:aws:s3:::aws-controltower*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Deletion of Log Archive

This control prevents deletion of Amazon S3 buckets created by AWS Control Tower in the log archive account. This is a preventive control with mandatory guidance. By default, this control is enabled on the **Security OU**.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRAUDITBUCKETDELETIONPROHIBITED",
      "Effect": "Deny",
      "Action": [
        "s3:DeleteBucket"
        ],
      "Resource": [
        "arn:aws:s3:::aws-controltower*"
        ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/AWSControlTowerExecution"
          }
      }
    }
  ]
```

```
}
```

Detect Public Read Access Setting for Log Archive

This control detects whether public read access is enabled to the Amazon S3 buckets in the log archive shared account. This control does not change the status of the account. This is a detective control with mandatory guidance. By default, this control is enabled on the **Security OU**.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check that your S3 buckets do not allow
 public access
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForS3PublicRead:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks that your S3 buckets do not allow public read access. If an
 S3 bucket policy or bucket ACL allows public read access, the bucket is noncompliant.
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_PUBLIC_READ_PROHIBITED
      Scope:
        ComplianceResourceTypes:
          - AWS::S3::Bucket
```

Detect Public Write Access Setting for Log Archive

This control detects whether public write access is enabled to the Amazon S3 buckets in the log archive shared account. This control does not change the status of the account. This is a detective control with mandatory guidance. By default, this control is enabled on the **Security OU**.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Configure AWS Config rules to check that your S3 buckets do not allow
 public access
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForS3PublicWrite:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks that your S3 buckets do not allow public write access. If an
 S3 bucket policy or bucket ACL allows public write access, the bucket is noncompliant.
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_PUBLIC_WRITE_PROHIBITED
      Scope:
        ComplianceResourceTypes:
          - AWS::S3::Bucket
```

Detect whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabled

This control detects whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabled. The rule is NON_COMPLIANT if either CloudTrail or CloudTrail Lake is not enabled in a shared account. This is a detective control with mandatory guidance. By default, this control is enabled on the **Security OU**.

The artifact for this control is the following AWS Config rule.

```
    AWSTemplateFormatVersion: 2010-09-09
 Description: Configure AWS Config rules to detect whether an account has AWS
CloudTrail or CloudTrail Lake enabled.

 Parameters:
   ConfigRuleName:
     Type: 'String'
     Description: 'Name for the Config rule'

 Resources:
   CheckForCloudtrailEnabled:
```

```
      Type: AWS::Config::ConfigRule
      Properties:
        ConfigRuleName: !Sub ${ConfigRuleName}
        Description: Detects whether an account has AWS CloudTrail or CloudTrail Lake
enabled. The rule is NON_COMPLIANT if either CloudTrail or CloudTrail Lake is not
enabled in an account.
        Source:
          Owner: AWS
          SourceIdentifier: CLOUD_TRAIL_ENABLED
```

Mandatory controls for all OUs

The following 15 mandatory controls are enabled by default on all OUs, when you set up your landing zone.

> **Note**
>
> The four mandatory controls with `"Sid": "GRCLOUDTRAILENABLED"` are identical by design. The sample code is correct.

Topics

- Disallow Changes to Amazon CloudWatch Logs Log Groups set up by AWS Control Tower
- Disallow Deletion of AWS Config Aggregation Authorizations Created by AWS Control Tower
- Disallow Configuration Changes to CloudTrail
- Integrate CloudTrail Events with Amazon CloudWatch Logs
- Enable CloudTrail in All Available Regions
- Enable Integrity Validation for CloudTrail Log File
- Disallow Changes to Amazon CloudWatch Set Up by AWS Control Tower
- Disallow Changes to Tags Created by AWS Control Tower for AWS Config Resources
- Disallow Configuration Changes to AWS Config
- Enable AWS Config in All Available Regions
- Disallow Changes to AWS Config Rules Set Up by AWS Control Tower
- Disallow Changes to AWS IAM Roles Set Up by AWS Control Tower and AWS CloudFormation
- Disallow Changes to AWS Lambda Functions Set Up by AWS Control Tower

- Disallow Changes to Amazon SNS Set Up by AWS Control Tower
- Disallow Changes to Amazon SNS Subscriptions Set Up by AWS Control Tower

Disallow Changes to Amazon CloudWatch Logs Log Groups set up by AWS Control Tower

This control prevents changes to the retention policy for Amazon CloudWatch Logs log groups that AWS Control Tower created in the log archive account when you set up your landing zone. It also prevents modifying the log retention policy in customer accounts. This is a preventive control with mandatory guidance. By default, this control is enabled on all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRLOGGROUPPOLICY",
            "Effect": "Deny",
            "Action": [
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:*aws-controltower*"
            ],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

Disallow Deletion of AWS Config Aggregation Authorizations Created by AWS Control Tower

This control prevents deletion of AWS Config aggregation authorizations that AWS Control Tower created in the audit account when you set up your landing zone. This is a preventive control with mandatory guidance. By default, this control is enabled on all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRCONFIGAGGREGATIONAUTHORIZATIONPOLICY",
      "Effect": "Deny",
      "Action": [
        "config:DeleteAggregationAuthorization"
      ],
      "Resource": [
        "arn:aws:config:*:*:aggregation-authorization*"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalArn": "arn:aws:iam::*:role/AWSControlTowerExecution"
        },
        "StringLike": {
          "aws:ResourceTag/aws-control-tower": "managed-by-control-tower"
        }
      }
    }
  ]
}
```

Disallow Configuration Changes to CloudTrail

This control prevents configuration changes to CloudTrail in your landing zone. This is a preventive control with mandatory guidance. By default, this control is enabled on all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCLOUDTRAILENABLED",
            "Effect": "Deny",
            "Action": [
                "cloudtrail:DeleteTrail",
                "cloudtrail:PutEventSelectors",
                "cloudtrail:StopLogging",
                "cloudtrail:UpdateTrail"
            ],
            "Resource": ["arn:aws:cloudtrail:*:*:trail/aws-controltower-*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Integrate CloudTrail Events with Amazon CloudWatch Logs

This control performs real-time analysis of activity data by sending CloudTrail events to CloudWatch Logs log files. This is a preventive control with mandatory guidance. By default, this control is enabled on all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCLOUDTRAILENABLED",
```

```
            "Effect": "Deny",
            "Action": [
                "cloudtrail:DeleteTrail",
                "cloudtrail:PutEventSelectors",
                "cloudtrail:StopLogging",
                "cloudtrail:UpdateTrail"
            ],
            "Resource": ["arn:aws:cloudtrail:*:*:trail/aws-controltower-*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Enable CloudTrail in All Available Regions

This control enables CloudTrail in all available AWS Regions. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCLOUDTRAILENABLED",
            "Effect": "Deny",
            "Action": [
                "cloudtrail:DeleteTrail",
                "cloudtrail:PutEventSelectors",
                "cloudtrail:StopLogging",
                "cloudtrail:UpdateTrail"
            ],
            "Resource": ["arn:aws:cloudtrail:*:*:trail/aws-controltower-*"],
            "Condition": {
                "ArnNotLike": {
```

```
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Enable Integrity Validation for CloudTrail Log File

This control enables integrity validation for the CloudTrail log file in all accounts and OUs. It protects the integrity of account activity logs using CloudTrail log file validation, which creates a digitally signed digest file that contains a hash of each log that CloudTrail writes to Amazon S3. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCLOUDTRAILENABLED",
            "Effect": "Deny",
            "Action": [
                "cloudtrail:DeleteTrail",
                "cloudtrail:PutEventSelectors",
                "cloudtrail:StopLogging",
                "cloudtrail:UpdateTrail"
            ],
            "Resource": ["arn:aws:cloudtrail:*:*:trail/aws-controltower-*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Changes to Amazon CloudWatch Set Up by AWS Control Tower

This control disallows changes to Amazon CloudWatch; as it was configured by AWS Control Tower when you set up your landing zone. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRCLOUDWATCHEVENTPOLICY",
      "Effect": "Deny",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "events:DisableRule",
        "events:DeleteRule"
      ],
      "Resource": [
        "arn:aws:events:*:*:rule/aws-controltower-*"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN": "arn:aws:iam::*:role/AWSControlTowerExecution"
        }
      }
    }
  ]
}
```

Disallow Changes to Tags Created by AWS Control Tower for AWS Config Resources

This control prevents changes to the tags that AWS Control Tower created when you set up your landing zone, for AWS Config resources that collect configuration and compliance data. It denies

any `TagResource` and `UntagResource` operation for aggregation authorizations tagged by AWS Control Tower. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRCONFIGRULETAGSPOLICY",
      "Effect": "Deny",
      "Action": [
        "config:TagResource",
        "config:UntagResource"
      ],
      "Resource": ["*"],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN": "arn:aws:iam::*:role/AWSControlTowerExecution"
        },
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "aws-control-tower"
        }
      }
    }
  ]
}
```

Disallow Configuration Changes to AWS Config

This control prevents configuration changes to AWS Config. It ensures that AWS Config records resource configurations in a consistent manner by disallowing AWS Config settings changes. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRCONFIGENABLED",
            "Effect": "Deny",
            "Action": [
                "config:DeleteConfigurationRecorder",
                "config:DeleteDeliveryChannel",
                "config:DeleteRetentionConfiguration",
                "config:PutConfigurationRecorder",
                "config:PutDeliveryChannel",
                "config:PutRetentionConfiguration",
                "config:StopConfigurationRecorder"
            ],
            "Resource": ["*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Enable AWS Config in All Available Regions

This control enables AWS Config in all available AWS Regions. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "GRCONFIGENABLED",
            "Effect": "Deny",
            "Action": [
                "config:DeleteConfigurationRecorder",
                "config:DeleteDeliveryChannel",
                "config:DeleteRetentionConfiguration",
                "config:PutConfigurationRecorder",
                "config:PutDeliveryChannel",
                "config:PutRetentionConfiguration",
                "config:StopConfigurationRecorder"
            ],
            "Resource": ["*"],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN":"arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Disallow Changes to AWS Config Rules Set Up by AWS Control Tower

This control disallows changes to AWS Config Rules that were implemented by AWS Control Tower when the landing zone was set up. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRCONFIGRULEPOLICY",
      "Effect": "Deny",
      "Action": [
        "config:PutConfigRule",
```

```
        "config:DeleteConfigRule",
        "config:DeleteEvaluationResults",
        "config:DeleteConfigurationAggregator",
        "config:PutConfigurationAggregator"
      ],
      "Resource": ["*"],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN": "arn:aws:iam::*:role/AWSControlTowerExecution"
        },
        "StringEquals": {
          "aws:ResourceTag/aws-control-tower": "managed-by-control-tower"
        }
      }
    }
  ]
}
```

Disallow Changes to AWS IAM Roles Set Up by AWS Control Tower and AWS CloudFormation

This control disallows changes to the AWS IAM roles that AWS Control Tower created when the landing zone was set up. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

Control update

An updated version has been released for the mandatory control `AWS-GR_IAM_ROLE_CHANGE_PROHIBITED`.

This change to the control is required because accounts in OUs that are being enrolled into AWS Control Tower must have the `AWSControlTowerExecution` role enabled. The previous version of the control prevents this role from being created.

AWS Control Tower updated the existing control to add an exception so that AWS CloudFormation StackSets can create the `AWSControlTowerExecution` role. As a second measure, this new control protects the StackSets role to prevent principals in the child account from gaining access.

The new control version performs the following actions, in addition to all actions provided in the previous version:

- Allows the `stacksets-exec-*` role (owned by AWS CloudFormation) to perform actions on IAM roles that were created by AWS Control Tower.
- Prevents changes to any IAM role in child accounts, where the IAM role name matches the pattern `stacksets-exec-*`.

The update to the control version affects your OUs and accounts as follows:

- If you extend governance to an OU, that incoming OU receives the updated version of the control as part of the registration process. You do not need to update your landing zone to get the latest version for this OU. AWS Control Tower applies the latest version automatically to OUs that register.
- If you update or repair your landing zone at any time after this release, your control will be updated to this version for future provisioning.
- OUs created in or registered with AWS Control Tower before this release date, and which are part of a landing zone that has not been repaired or updated after the release date, will continue to operate with the old version of the control, which blocks the creation of the `AWSControlTowerExecution` role.
- One consequence of this control update is that your OUs can be functioning with different versions of the control. Update your landing zone to apply the updated version of the control to your OUs uniformly.

The artifact of the updated control is the following SCP.

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Sid": "GRIAMROLEPOLICY",
        "Effect": "Deny",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:DeleteRolePermissionsBoundary",
          "iam:DeleteRolePolicy",
          "iam:DetachRolePolicy",
          "iam:PutRolePermissionsBoundary",
```

```
          "iam:PutRolePolicy",
          "iam:UpdateAssumeRolePolicy",
          "iam:UpdateRole",
          "iam:UpdateRoleDescription"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-controltower-*",
          "arn:aws:iam::*:role/*AWSControlTower*",
          "arn:aws:iam::*:role/stacksets-exec-*"    #this line is new
        ],
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalArn": [
                "arn:aws:iam::*:role/AWSControlTowerExecution",
                "arn:aws:iam::*:role/stacksets-exec-*"    #this line is new
         ]
       }
      }
    }
  ]
}
```

The former artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRIAMROLEPOLICY",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
```

```
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-controltower-*",
        "arn:aws:iam::*:role/*AWSControlTower*"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/AWSControlTowerExecution"
        }
      }
    }
  ]
}
```

Disallow Changes to AWS Lambda Functions Set Up by AWS Control Tower

This control disallows changes to AWS Lambda functions set up by AWS Control Tower. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRLAMBDAFUNCTIONPOLICY",
      "Effect": "Deny",
      "Action": [
        "lambda:AddPermission",
        "lambda:CreateEventSourceMapping",
        "lambda:CreateFunction",
        "lambda:DeleteEventSourceMapping",
        "lambda:DeleteFunction",
        "lambda:DeleteFunctionConcurrency",
        "lambda:PutFunctionConcurrency",
        "lambda:RemovePermission",
        "lambda:UpdateEventSourceMapping",
        "lambda:UpdateFunctionCode",
```

```
        "lambda:UpdateFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:*:*:function:aws-controltower-*"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/AWSControlTowerExecution"
        }
      }
    }
  ]
}
```

Disallow Changes to Amazon SNS Set Up by AWS Control Tower

This control disallows changes to Amazon SNS set up by AWS Control Tower. It protects the integrity of Amazon SNS notification settings for your landing zone. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRSNSTOPICPOLICY",
      "Effect": "Deny",
      "Action": [
        "sns:AddPermission",
        "sns:CreateTopic",
        "sns:DeleteTopic",
        "sns:RemovePermission",
        "sns:SetTopicAttributes"
      ],
      "Resource": [
        "arn:aws:sns:*:*:aws-controltower-*"
      ],
```

```
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/AWSControlTowerExecution"
        }
      }
    }
  ]
}
```

Disallow Changes to Amazon SNS Subscriptions Set Up by AWS Control Tower

This control disallows changes to Amazon SNS subscriptions set up by AWS Control Tower. It protects the integrity of Amazon SNS subscriptions settings for your landing zone, to trigger notifications for AWS Config Rules compliance changes. This is a preventive control with mandatory guidance. By default, this control is enabled in all OUs.

The artifact for this control is the following SCP.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRSNSSUBSCRIPTIONPOLICY",
      "Effect": "Deny",
      "Action": [
        "sns:Subscribe",
        "sns:Unsubscribe"
      ],
      "Resource": [
        "arn:aws:sns:*:*:aws-controltower-SecurityNotifications"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/AWSControlTowerExecution"
        }
      }
    }
  ]
```

```
}
```

Proactive controls

These controls are referred to as *proactive* because they check your resources – before the resources are deployed – to determine whether the new resources will comply with the controls that are activated in your environment.

Proactive controls are *optional controls* implemented with CloudFormation hooks and hooks managed by AWS Control Tower.

Proactive controls fall into four main **Categories**. In the AWS Control Tower console, you can view the controls in groups according to their assigned categories, which are:

- **Control objectives**: Specific purposes for implementing controls in your environment.
- **Frameworks**: Industry-standard compliance frameworks.
- **Services**: The AWS services that the control may govern.
- **Groups**: Groups of controls designed to help you meet a specific policy standard.

In this reference guide, the proactive controls are categorized according to their associated AWS services.

Behavior of proactive controls

Proactive controls check resources whenever those resources are created or updated by means of CloudFormation stack operations. Specifically, these proactive controls are implemented as `preCreate` and `preUpdate` hook handlers. As a consequence, these controls may not affect requests that are made directly to services through the AWS console, through AWS APIs, or through other means such as AWS SDKs, or other Infrastructure-as-Code (IaC) tools. For more information about when `preCreate` and `preUpdate` hooks operate, see CloudFormation hooks.

> **Limitation of hooks managed by CloudFormation**
>
> Proactive controls evaluate strings passed into the CloudFormation hook within the `targetNames` property. Secure strings and secrets are not resolved before they are sent to the hook, which prevents the proactive control from evaluating the string. For more

information about how the `targetNames` are passed to hooks, see CloudFormation Hooks structure overview.

When you follow an example template to set up a test for a proactive control in your environment, be aware that the template is created to test one specific control only. Other controls may not receive a PASS rating for that template. This behavior is expected. We recommend that you test proactive controls individually before you enable them in your environment.

Update your proactive control hooks

To update the way that AWS Control Tower handles the CloudFormation hooks for your enabled proactive controls, follow the steps given in this section.

After you complete this process, you can utilize the full capacity of CloudFormation hooks, without restriction by AWS Control Tower. It eliminates the need to apply the **CT.CLOUDFORMATION.PR.1** preventive control before you can enable proactive controls.

The first time that you enable a proactive control, AWS Control Tower turns on the hook that it requires, without restricting any other CloudFormation hooks that you may have deployed on AWS. Only AWS Control Tower can change the AWS Control Tower hook, but principals with the correct permissions can change other CloudFormation hooks in your environment.

If you enabled proactive controls before the launch of the service-linked hook integration, follow these steps.

To update your proactive control hooks

- Reset any one enabled proactive control on the current OU by calling the **ResetEnabledControl** API or using the console's **Reset control** button on the **Control** page.
- Navigate to the **CT.CLOUDFORMATION.PR.1** control in the AWS Control Tower controls library.
- Disable the **CT.CLOUDFORMATION.PR.1** control.

Repeat this procedure for each OU that has proactive controls enabled, if those controls were enabled before the launch of the service-linked hook integration.

Important

The **Reset** function resets control drift. **Reset** operates differently for proactive controls than for any other type of control in AWS Control Tower. When you reset any enabled proactive control on an OU, all of the enabled proactive controls for that OU are reset. This behavior happens because the artifacts for all enabled proactive controls are bundled together, and they are deployed together, each time the `ResetEnabledControl` API is called.

Topics

- Amazon API Gateway controls
- AWS Certificate Manager controls
- AWS AppSync controls
- Amazon Athena controls
- Amazon CloudFront controls
- AWS CloudTrail controls
- Amazon CloudWatch controls
- AWS CodeBuild controls
- AWS Database Migration Service (AWS DMS) controls
- Amazon DocumentDB controls
- Amazon DynamoDB controls
- DynamoDB Accelerator controls
- AWS Elastic Beanstalk controls
- Amazon Elastic Compute Cloud (Amazon EC2) controls
- Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling controls
- Amazon ElastiCache controls
- Amazon Elastic Container Registry controls
- Amazon Elastic Container Service controls
- Amazon Elastic File System controls
- Amazon Elastic Kubernetes Service (EKS) controls
- Elastic Load Balancing controls

- Amazon Elastic Map Reduce (Amazon EMR) controls
- AWS Glue controls
- Amazon GuardDuty controls
- AWS Identity and Access Management (IAM) controls
- AWS Key Management Service (AWS KMS) controls
- Amazon Kinesis controls
- AWS Lambda controls
- Amazon MQ controls
- Amazon Managed Streaming for Apache Kafka (Amazon MSK) controls
- Amazon Neptune controls
- AWS Network Firewall controls
- Amazon OpenSearch controls
- Amazon Relational Database Service (Amazon RDS) controls
- Amazon Redshift controls
- Amazon Simple Storage Service (Amazon S3) controls
- Amazon SageMaker AI controls
- Amazon Simple Queue Service (Amazon SQS) controls
- AWS Step Functions controls
- AWS WAF regional controls
- AWS WAF controls
- AWS WAFV2 controls

Amazon API Gateway controls

Topics

- [CT.APIGATEWAY.PR.1] Require an Amazon API Gateway REST and WebSocket API to have logging activated
- [CT.APIGATEWAY.PR.2] Require an Amazon API Gateway REST API stage to have AWS X-Ray tracing activated
- [CT.APIGATEWAY.PR.3] Require that an Amazon API Gateway REST API stage has encryption at rest configured for cache data

- [CT.APIGATEWAY.PR.4] Require an Amazon API Gateway V2 stage to have access logging activated
- [CT.APIGATEWAY.PR.5] Require Amazon API Gateway V2 Websocket and HTTP routes to specify an authorization type
- [CT.APIGATEWAY.PR.6] Require an Amazon API Gateway REST domain to use a security policy that specifies a minimum TLS protocol version of TLSv1.2

[CT.APIGATEWAY.PR.1] Require an Amazon API Gateway REST and WebSocket API to have logging activated

This control checks whether all methods in Amazon API Gateway stage have execution logging configured.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGateway::Stage`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.1 example templates

Explanation

Amazon API Gateway REST or WebSocket API stages should have relevant logs enabled. API Gateway REST and WebSocket API execution logging provides detailed records of requests made to API Gateway REST and WebSocket API stages. The stages include API integration backend responses, Lambda authorizer responses, and the `requestId` for AWS integration endpoints.

Usage considerations

- This control requires Amazon API Gateway stages to configure execution logging for all methods and resources (`HttpMethod` of `*` and `ResourcePath` of `/*`).

Remediation for rule failure

Configure execution logging on Amazon API Gateway stages with a `MethodSetting` that sets `LoggingLevel` to `ERROR` or `INFO` for all methods (`HttpMethod` of `*` and `ResourcePath` of `/*`). Ensure that you do not set `LoggingLevel` to `OFF` for any method setting.

The examples that follow show how to implement this remediation.

Amazon API Gateway Stage - Example

Amazon API Gateway stage configured with `error` level execution logging for all methods and resources. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApiGatewayStage": {
        "Type": "AWS::ApiGateway::Stage",
        "Properties": {
            "StageName": "Sample",
            "Description": "Sample Stage",
            "RestApiId": {
                "Ref": "RestApi"
            },
            "DeploymentId": {
                "Ref": "Deployment"
            },
            "MethodSettings": [
                {
                    "ResourcePath": "/*",
                    "HttpMethod": "*",
                    "LoggingLevel": "ERROR"
                }
            ]
        }
```

```
    }
}
```

YAML example

```
ApiGatewayStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: Sample
    Description: Sample Stage
    RestApiId: !Ref 'RestApi'
    DeploymentId: !Ref 'Deployment'
    MethodSettings:
      - ResourcePath: /*
        HttpMethod: '*'
        LoggingLevel: ERROR
```

CT.APIGATEWAY.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_v1_execution_logging_enabled_check
#
# Description:
#   This control checks whether all methods in Amazon API Gateway stage have execution
 logging configured.
#
# Reports on:
#   AWS::ApiGateway::Stage
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any API Gateway stage resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: In the stage resource, 'MethodSettings' is not present or is provided and
 is an empty list.
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: In the stage resource, Execution Logging is not configured for all HTTP
 Methods and API resources (In
#            'MethodSettings', 'LoggingLevel' is omitted, or not set to 'ERROR' or
 'INFO', for 'HttpMethod' of '*' and
#            'ResourcePath' of '/*' )
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: In the stage resource, Execution Logging is configured for all HTTP
 Methods and API resources (In
#            'MethodSettings', 'LoggingLevel' is set to 'ERROR' or 'INFO', for
 'HttpMethod' of '*' and
#            'ResourcePath' of '/*' )
#       And: 'LoggingLevel' has been set to 'OFF' for any other Method Setting
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: In the stage resource, Execution Logging is configured for all HTTP
 Methods and API resources (In
#            'MethodSettings', 'LoggingLevel' is set to 'ERROR' or 'INFO', for
 'HttpMethod' of '*' and
#            'ResourcePath' of '/*' )
#       And: 'LoggingLevel' has not been provided or set to 'ERROR' or 'INFO' for all
 other Method Settings
#      Then: PASS

#
```

```
# Constants
#
let API_GW_STAGE_TYPE = "AWS::ApiGateway::Stage"
let INPUT_DOCUMENT = this
let VALID_LOG_LEVELS = [ "ERROR", "INFO" ]

#
# Assignments
#
let api_gateway_stages = Resources.*[ Type == %API_GW_STAGE_TYPE ]

#
# Primary Rules
#
rule api_gw_v1_execution_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %api_gateway_stages not empty {
    check(%api_gateway_stages.Properties)
        <<
        [CT.APIGATEWAY.PR.1]: Require an Amazon API Gateway REST and WebSocket API to
 have logging activated
        [FIX]: Configure execution logging on Amazon API Gateway stages with a
 'MethodSetting' that sets 'LoggingLevel' to 'ERROR' or 'INFO' for all methods
 ('HttpMethod' of '*' and 'ResourcePath' of '/*'). Ensure that you do not set
 'LoggingLevel' to 'OFF' for any method setting.
        >>
}

rule api_gw_v1_execution_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %API_GW_STAGE_TYPE) {
    check(%INPUT_DOCUMENT.%API_GW_STAGE_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.1]: Require an Amazon API Gateway REST and WebSocket API to
 have logging activated
        [FIX]: Configure execution logging on Amazon API Gateway stages with a
 'MethodSetting' that sets 'LoggingLevel' to 'ERROR' or 'INFO' for all methods
 ('HttpMethod' of '*' and 'ResourcePath' of '/*'). Ensure that you do not set
 'LoggingLevel' to 'OFF' for any method setting.
        >>
}

#
# Parameterized Rules
#
rule check(api_gateway_stage) {
```

```
    %api_gateway_stage {
        # Scenario 2
        MethodSettings exists
        MethodSettings is_list
        MethodSettings not empty

        # Scenario 3
        # At least one wildcard entry exists with valid logging enabled
        some MethodSettings[*] {
            HttpMethod exists
            ResourcePath exists
            LoggingLevel exists

            HttpMethod == "*"
            ResourcePath == "/*"
            LoggingLevel in %VALID_LOG_LEVELS
        }

        # Scenario 4, 5
        # When other methods explictly set/override logging settings, ensure that
 logging is not disabled
        MethodSettings[*] {
            when LoggingLevel exists {
                LoggingLevel in %VALID_LOG_LEVELS
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APIGATEWAY.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: ExampleRestApi
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
```

```
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Example
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
      MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        LoggingLevel: ERROR
      - ResourcePath: "/"
        HttpMethod: GET
        LoggingLevel: INFO
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: ExampleRestApi
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
```

```
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Example
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
```

[CT.APIGATEWAY.PR.2] Require an Amazon API Gateway REST API stage to have AWS X-Ray tracing activated

This control ensures that AWS X-Ray tracing is enabled on Amazon API Gateway REST APIs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGateway::Stage`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.2 example templates

Explanation

AWS X-Ray active tracing enables a more rapid response to performance changes in the underlying infrastructure. Changes in performance could result in a lack of availability of the API. X-Ray active tracing provides real-time metrics of user requests that flow through your API Gateway REST API operations and connected services.

Remediation for rule failure

Set `TracingEnabled` to `true`.

The examples that follow show how to implement this remediation.

Amazon API Gateway Stage - Example

Amazon API Gateway stage configured with AWS X-Ray tracing enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApiGatewayStage": {
        "Type": "AWS::ApiGateway::Stage",
        "Properties": {
            "StageName": "Sample",
            "Description": "Sample Stage",
            "TracingEnabled": true,
            "RestApiId": {
                "Ref": "RestApi"
            },
            "DeploymentId": {
                "Ref": "Deployment"
            }
        }
    }
}
```

YAML example

```
ApiGatewayStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: Sample
    Description: Sample Stage
    TracingEnabled: true
    RestApiId: !Ref 'RestApi'
    DeploymentId: !Ref 'Deployment'
```

CT.APIGATEWAY.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_xray_enabled_check
#
# Description:
```

```
#   This control ensures that AWS X-Ray tracing is enabled on Amazon API Gateway REST
 APIs.
#
# Reports on:
#   AWS::ApiGateway::Stage
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any API Gateway stage resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: 'TracingEnabled' is not present on the API Gateway stage
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: 'TracingEnabled' is present on the API Gateway stage and is set to
 bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: 'TracingEnabled' is present on the API Gateway stage and is set to
 bool(true)
#      Then: PASS

#
# Constants
#
let API_GW_STAGE_TYPE = "AWS::ApiGateway::Stage"
let INPUT_DOCUMENT = this

#
# Assignments
#
```

```
let api_gateway_stages = Resources.*[ Type == %API_GW_STAGE_TYPE ]

#
# Primary Rules
#
rule api_gw_xray_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %api_gateway_stages not empty {
    check(%api_gateway_stages.Properties)
        <<
        [CT.APIGATEWAY.PR.2]: Require an Amazon API Gateway REST API stage to have AWS
 X-Ray tracing activated
        [FIX]: Set 'TracingEnabled' to 'true'.
        >>
}

rule api_gw_xray_enabled_check when is_cfn_hook(%INPUT_DOCUMENT, %API_GW_STAGE_TYPE) {
    check(%INPUT_DOCUMENT.%API_GW_STAGE_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.2]: Require an Amazon API Gateway REST API stage to have AWS
 X-Ray tracing activated
        [FIX]: Set 'TracingEnabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(api_gateway_stage) {
    %api_gateway_stage {
        # Scenario 2, 3, 4
        TracingEnabled exists
        TracingEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APIGATEWAY.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: Testing
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
```

```
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Dev
      Description: Dev Stage
      TracingEnabled: true
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: Testing
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
```

```
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Dev
      Description: Dev Stage
      TracingEnabled: false
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
```

[CT.APIGATEWAY.PR.3] Require that an Amazon API Gateway REST API stage has encryption at rest configured for cache data

This control checks whether an Amazon API Gateway REST API stage that has caching enabled also encrypts the caches.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGateway::Stage`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.3 example templates

Explanation

Encrypting data at rest reduces the risk that data stored on disk may be accessible by a user not authenticated to AWS. It adds another set of access controls to limit unauthorized users' ability to obtain the data. For example, API permissions are required to decrypt the data before it can be read.

For an added layer of security, API Gateway REST API caches should be encrypted at rest.

Usage considerations

- This control applies only to API Gateway stage resources with cache clustering enabled.
- Where cache clustering is enabled, this control requires cache encryption to be enabled for all resources and methods by specifying a `MethodSetting` entry with an `HttpMethod` of `*` and `ResourcePath` of `/*`.

Remediation for rule failure

Configure encryption on API Gateway caches with a `MethodSetting` that sets `CacheDataEncrypted` to true for all methods (`HttpMethod` of `*` and `ResourcePath` of `/*`). Ensure that you do not set `CacheDataEncrypted` to false for any method setting.

The examples that follow show how to implement this remediation.

API Gateway stage examples

This example shows the API Gateway stage configured to encrypt cache data for all methods (`HttpMethod` of `*` and `ResourcePath` of `/*`). The example is shown in JSON and in YAML.

JSON example

```
{
    "ApiGatewayStage": {
        "Type": "AWS::ApiGateway::Stage",
        "Properties": {
            "StageName": "Dev",
            "Description": "Development Stage",
            "CacheClusterEnabled": true,
            "CacheClusterSize": 0.5,
            "RestApiId": {
                "Ref": "RestApi"
            },
            "DeploymentId": {
                "Ref": "Deployment"
            },
            "MethodSettings": [
                {
                    "ResourcePath": "/*",
                    "HttpMethod": "*",
                    "CacheDataEncrypted": true
                },
                {
                    "ResourcePath": "/",
                    "HttpMethod": "POST"
                }
            ]
        }
    }
}
```

YAML example

```
ApiGatewayStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: Dev
    Description: Development Stage
    CacheClusterEnabled: true
    CacheClusterSize: 0.5
    RestApiId: !Ref 'RestApi'
    DeploymentId: !Ref 'Deployment'
    MethodSettings:
      - ResourcePath: /*
        HttpMethod: '*'
        CacheDataEncrypted: true
      - ResourcePath: /
        HttpMethod: POST
```

CT.APIGATEWAY.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_cache_encrypted_check
#
# Description:
#   This rule checks whether Amazon API Gateway REST API stages that have caching
 enabled also encrypt the caches.
#
# Reports on:
#   AWS::ApiGateway::Stage
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
```

```
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon API Gateway stage resources
#     Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway stage resource
#       And: 'CacheClusterEnabled' is not set, or is set to bool(false) on the API
 Gateway stage resource
#     Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway stage resource
#       And: 'CacheClusterEnabled' is set to bool(true) on the API Gateway stage
 resource
#       And: In the Stage resource, 'MethodSettings' is not present or is provided and
 is an empty list.
#     Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway stage resource
#       And: 'CacheClusterEnabled' is set to bool(true) on the API Gateway stage
 resource
#       And: In the stage resource, cache data encryption is not enabled for all HTTP
 methods and API resources (In
#           'MethodSettings', 'CacheDataEncrypted' is omitted or set to bool(false) for
 'HttpMethod' of '*' and
#           'ResourcePath' of '/*' )
#     Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: 'CacheClusterEnabled' is set to bool(true) on the API Gateway stage
 resource
#       And: In the stage resource, cache data encryption is configured for all
 'MethodSettings' (CacheDataEncrypted is
#           bool(true) for 'HttpMethod' of '*' and 'ResourcePath' of '/*')
#       And: 'CacheDataEncrypted' has been set to bool(false) for any other method
 settings
#     Then: FAIL
```

```
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway stage resource
#       And: 'CacheClusterEnabled' is set to bool(true) on the API Gateway stage
 resource
#       And: In the stage resource cache data encryption is configured for all
 'MethodSettings' (CacheDataEncrypted is
#           bool(true) for 'HttpMethod' of '*' and 'ResourcePath' of '/*')
#       And: 'CacheDataEncrypted' has not been provided or set to bool(true) for all
 other method settings
#     Then: PASS

#
# Constants
#
let API_GW_STAGE_TYPE = "AWS::ApiGateway::Stage"
let INPUT_DOCUMENT = this

#
# Assignments
#
let api_gateway_stages = Resources.*[ Type == %API_GW_STAGE_TYPE ]

#
# Primary Rules
#
rule api_gw_cache_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                       %api_gateway_stages not empty {
    check(%api_gateway_stages.Properties)
        <<
        [CT.APIGATEWAY.PR.3]: Require that an Amazon API Gateway REST API stage has
 encryption at rest configured for cache data
        [FIX]: Configure encryption on API Gateway caches with a 'MethodSetting'
 that sets 'CacheDataEncrypted' to true for all methods ('HttpMethod' of '*' and
 'ResourcePath' of '/*'). Ensure that you do not set 'CacheDataEncrypted' to false for
 any method setting.
        >>
}

rule api_gw_cache_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT, %API_GW_STAGE_TYPE)
 {
    check(%INPUT_DOCUMENT.%API_GW_STAGE_TYPE.resourceProperties)
        <<
```

```
        [CT.APIGATEWAY.PR.3]: Require that an Amazon API Gateway REST API stage has
 encryption at rest configured for cache data
        [FIX]: Configure encryption on API Gateway caches with a 'MethodSetting'
 that sets 'CacheDataEncrypted' to true for all methods ('HttpMethod' of '*' and
 'ResourcePath' of '/*'). Ensure that you do not set 'CacheDataEncrypted' to false for
 any method setting.
        >>
}

#
# Parameterized Rules
#
rule check(api_gateway_stage) {
    %api_gateway_stage [
        CacheClusterEnabled exists
        CacheClusterEnabled == true
    ] {
        # Scenario 2, 3, 4, 6
        cache_encrypted(this)
    }
}

rule cache_encrypted(api_gateway_stage) {
    %api_gateway_stage {
        MethodSettings exists
        MethodSettings is_list
        MethodSettings not empty

        some MethodSettings[*] {
            HttpMethod exists
            ResourcePath exists
            CacheDataEncrypted exists

            HttpMethod == "*"
            ResourcePath == "/*"
            CacheDataEncrypted == true
        }

        MethodSettings[*] {
            when CacheDataEncrypted exists {
                CacheDataEncrypted == true
            }
        }
    }
```

```
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APIGATEWAY.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: ExampleRestApi
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
```

```
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Example
      Description: Example Stage
      CacheClusterEnabled: true
      CacheClusterSize: 0.5
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
      MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        CacheDataEncrypted: true
      - ResourcePath: "/"
        HttpMethod: "POST"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: ExampleRestApi
  GetMethod:
    DependsOn: PutMethod
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  PutMethod:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: PUT
      RestApiId:
        Ref: RestApi
      ResourceId:
        Fn::GetAtt:
        - "RestApi"
        - "RootResourceId"
      AuthorizationType: NONE
      MethodResponses:
      - StatusCode: "200"
      Integration:
        Type: MOCK
  Deployment:
    DependsOn: GetMethod
    Type: 'AWS::ApiGateway::Deployment'
```

```
    Properties:
      RestApiId:
        Ref: RestApi
  ApiGatewayStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: Example
      Description: Example Stage
      CacheClusterEnabled: true
      CacheClusterSize: 0.5
      RestApiId:
        Ref: RestApi
      DeploymentId:
        Ref: Deployment
      MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        CacheDataEncrypted: false
```

[CT.APIGATEWAY.PR.4] Require an Amazon API Gateway V2 stage to have access logging activated

This control checks whether Amazon API Gateway V2 stages have access logging enabled. Access logging is supported for HTTP and WebSocket APIs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGatewayV2::Stage`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.4 example templates

Explanation

Access logging allows you to log who has called your API and how the caller gained access to the API. You can create your own log group or choose an existing log group that could be managed by API Gateway.

Remediation for rule failure

Provide an `AccessLogSettings` configuration, setting `DestinationArn` to the ARN of an Amazon CloudWatch log group and `Format` to a single line log format configuration.

The examples that follow show how to implement this remediation.

Amazon API Gateway HTTP API Stage - Example

Amazon API Gateway HTTP API stage configured to send API access logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "HttpApiStage": {
        "Type": "AWS::ApiGatewayV2::Stage",
        "Properties": {
            "StageName": "SampleStage",
            "Description": "Sample Stage",
            "ApiId": {
                "Ref": "HttpApi"
            },
            "AccessLogSettings": {
                "DestinationArn": {
                    "Fn::GetAtt": [
                        "LogGroup",
                        "Arn"
                    ]
                },
                "Format": "{\"requestId\":\"$context.requestId\", \"ip\":
 \"$context.identity.sourceIp\", \"user\":\"$context.identity.user\",\"requestTime\":
\"$context.requestTime\"}"
            }
        }
    }
}
```

YAML example

```
HttpApiStage:
  Type: AWS::ApiGatewayV2::Stage
  Properties:
    StageName: SampleStage
    Description: Sample Stage
    ApiId: !Ref 'HttpApi'
    AccessLogSettings:
      DestinationArn: !GetAtt 'LogGroup.Arn'
      Format: '{"requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
        "user":"$context.identity.user","requestTime":"$context.requestTime"}'
```

CT.APIGATEWAY.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_v2_access_logs_enabled_check
#
# Description:
#   This control checks whether Amazon API Gateway V2 stages have access logging
 enabled. Access logging is supported for HTTP and WebSocket APIs.
#
# Reports on:
#   AWS::ApiGatewayV2::Stage
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any APIGatewayV2 stage resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an APIGatewayV2 stage resource
#       And: 'AccessLogSettings' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an APIGatewayV2 stage resource
#       And: 'AccessLogSettings' has been provided
#       And: 'AccessLogSettings.DestinationArn' has not been provided, or has been
 provided as an empty string or
#            invalid local reference
#       And: 'AccessLogSettings.Format' is provided as a non-empty string
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an APIGatewayV2 stage resource
#       And: 'AccessLogSettings' has been provided
#       And: 'AccessLogSettings.DestinationArn' is provided as a non-empty string or
 valid local reference
#       And: 'AccessLogSettings.Format' has not been provided, or is an empty string
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an APIGatewayV2 stage resource
#       And: 'AccessLogSettings' has been provided
#       And: 'AccessLogSettings.DestinationArn' is provided as a non-empty string or
 valid local reference
#       And: 'AccessLogSettings.Format' is provided as a non-empty string
#      Then: PASS

#
# Constants
#
let API_GW_V2_STAGE_TYPE = "AWS::ApiGatewayV2::Stage"
let INPUT_DOCUMENT = this

#
# Assignments
#
let api_gateway_v2_stages = Resources.*[ Type == %API_GW_V2_STAGE_TYPE ]
```

```
#
# Primary Rules
#
rule api_gw_v2_access_logs_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %api_gateway_v2_stages not empty {
    check(%api_gateway_v2_stages.Properties)
        <<
        [CT.APIGATEWAY.PR.4]: Require an Amazon API Gateway V2 stage to have access
 logging activated
            [FIX]: Provide an 'AccessLogSettings' configuration, setting
 'DestinationArn' to the ARN of an Amazon CloudWatch log group and 'Format' to a single
 line log format configuration.
        >>
}

rule api_gw_v2_access_logs_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %API_GW_V2_STAGE_TYPE) {
    check(%INPUT_DOCUMENT.%API_GW_V2_STAGE_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.4]: Require an Amazon API Gateway V2 stage to have access
 logging activated
            [FIX]: Provide an 'AccessLogSettings' configuration, setting
 'DestinationArn' to the ARN of an Amazon CloudWatch log group and 'Format' to a single
 line log format configuration.
        >>
}

#
# Parameterized Rules
#
rule check(api_gateway_v2_stage) {
    %api_gateway_v2_stage {
        # Scenario 2
        AccessLogSettings exists
        AccessLogSettings is_struct

        AccessLogSettings {
            # Scenario 3
            DestinationArn exists
            check_is_string_and_not_empty(DestinationArn) or
            check_local_references(%INPUT_DOCUMENT, DestinationArn,
 "AWS::Logs::LogGroup")
```

```
            # Scenario 4, 5
            Format exists
            check_is_string_and_not_empty(Format)
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
```

```
        Type == %referenced_resource_type
    }
}
```

CT.APIGATEWAY.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      RetentionInDays: 7
  HttpApi:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Name: ExampleApi
      ProtocolType: HTTP
  HttpApiStage:
    Type: 'AWS::ApiGatewayV2::Stage'
    Properties:
      StageName: ExampleStage
      Description: Example Stage
      ApiId:
        Ref: HttpApi
      AccessLogSettings:
        DestinationArn:
          Fn::GetAtt:
          - "LogGroup"
          - "Arn"
        Format: >-
          {"requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
          "user":"$context.identity.user","requestTime":"$context.requestTime"}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  HttpApi:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Name: ExampleApi
      ProtocolType: HTTP
  HttpApiStage:
    Type: 'AWS::ApiGatewayV2::Stage'
    Properties:
      StageName: ExampleStage
      Description: Example Stage
      ApiId:
        Ref: HttpApi
```

[CT.APIGATEWAY.PR.5] Require Amazon API Gateway V2 Websocket and HTTP routes to specify an authorization type

This control checks whether Amazon API Gateway V2 API routes have an authorization type set.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGatewayV2::Route`, `AWS::ApiGatewayV2::ApiGatewayManagedOverrides`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.5 example templates

Explanation

API Gateway supports multiple mechanisms for controlling and managing access to your Websocket or HTTP API. By specifying an authorization type, you can restrict access to your API, to allow only required users or processes.

Usage considerations

- This control applies only to routes created by means of the `AWS::ApiGatewayV2::Route` resource, and to managed overrides that apply to HTTP API routes that are created through quick create.
- This control does not evaluate HTTP API routes imported using the `Body` or `BodyS3Location` properties of `AWS::ApiGatewayV2::API` resources.

Remediation for rule failure

For Amazon API Gateway V2 routes, set `AuthorizationType` to `AWS_IAM`, `JWT` or `CUSTOM`. For Amazon API Gateway V2 managed route overrides with `AuthorizationType`, set `AuthorizationType` to `AWS_IAM`, `JWT` or `CUSTOM`.

The examples that follow show how to implement this remediation.

Amazon API Gateway V2 Route - Example

Amazon API Gateway V2 route configured with AWS IAM authorization. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApiGatewayV2Route": {
        "Type": "AWS::ApiGatewayV2::Route",
        "Properties": {
            "ApiId": {
                "Ref": "WebsocketApi"
            },
            "RouteKey": "$connect",
            "AuthorizationType": "AWS_IAM"
        }
```

```
    }
}
```

YAML example

```
ApiGatewayV2Route:
  Type: AWS::ApiGatewayV2::Route
  Properties:
    ApiId: !Ref 'WebsocketApi'
    RouteKey: $connect
    AuthorizationType: AWS_IAM
```

The examples that follow show how to implement this remediation.

Amazon API Gateway V2 Managed Overrides - Example

Amazon API Gateway V2 managed overrides configured with AWS IAM authorization. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApiGatewayManagedOverride": {
        "Type": "AWS::ApiGatewayV2::ApiGatewayManagedOverrides",
        "Properties": {
            "ApiId": {
                "Ref": "HttpApi"
            },
            "Route": {
                "AuthorizationType": "AWS_IAM"
            }
        }
    }
}
```

YAML example

```
ApiGatewayManagedOverride:
  Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
  Properties:
    ApiId: !Ref 'HttpApi'
    Route:
      AuthorizationType: AWS_IAM
```

CT.APIGATEWAY.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_v2_authorization_type_configured_check
#
# Description:
#   This control checks whether Amazon API Gateway V2 API routes have an authorization
 type set.
#
# Reports on:
#   AWS::ApiGatewayV2::Route, AWS::ApiGatewayV2::ApiGatewayManagedOverrides
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon API Gateway V2 route or
 managed route overrides resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway V2 managed route
 overrides resource
```

```
#       And: In 'Route', 'AuthorizationType' has not been provided
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway V2 route resource
#       And: 'AuthorizationType' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway V2 route or managed
 route overrides resource
#       And: 'AuthorizationType' has been provided and set to a value other than
 'AWS_IAM', 'JWT' or 'CUSTOM'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon API Gateway V2 route or managed
 route overrides resource
#       And: 'AuthorizationType' has been provided and set to a value of 'AWS_IAM',
 'JWT' or 'CUSTOM'
#      Then: PASS

#
# Constants
#
let API_GW_ROUTE_TYPE = "AWS::ApiGatewayV2::Route"
let API_GW_MANAGED_OVERRIDE_TYPE = "AWS::ApiGatewayV2::ApiGatewayManagedOverrides"
let ALLOWED_AUTHORIZATION_TYPES = ["AWS_IAM", "JWT", "CUSTOM"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let api_route = Resources.*[ Type == %API_GW_ROUTE_TYPE ]
let api_override = Resources.*[ Type == %API_GW_MANAGED_OVERRIDE_TYPE ]

#
# Primary Rules
#
rule api_gw_v2_authorization_type_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %api_route not empty {
    check_api_route(%api_route.Properties)
         <<
```

```
        [CT.APIGATEWAY.PR.5]: Require Amazon API Gateway V2 Websocket and HTTP routes
 to specify an authorization type
           [FIX]: For Amazon API Gateway V2 routes, set 'AuthorizationType' to
 'AWS_IAM', 'JWT' or 'CUSTOM'. For Amazon API Gateway V2 managed route overrides with
 'AuthorizationType', set 'AuthorizationType' to 'AWS_IAM', 'JWT' or 'CUSTOM'.
        >>
}

rule api_gw_v2_authorization_type_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                    %api_override not empty {
    check_api_override(%api_override.Properties)
        <<
        [CT.APIGATEWAY.PR.5]: Require Amazon API Gateway V2 Websocket and HTTP routes
 to specify an authorization type
           [FIX]: For Amazon API Gateway V2 routes, set 'AuthorizationType' to
 'AWS_IAM', 'JWT' or 'CUSTOM'. For Amazon API Gateway V2 managed route overrides with
 'AuthorizationType', set 'AuthorizationType' to 'AWS_IAM', 'JWT' or 'CUSTOM'.
        >>
}

rule api_gw_v2_authorization_type_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %API_GW_ROUTE_TYPE) {
    check_api_route(%INPUT_DOCUMENT.%API_GW_ROUTE_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.5]: Require Amazon API Gateway V2 Websocket and HTTP routes
 to specify an authorization type
           [FIX]: For Amazon API Gateway V2 routes, set 'AuthorizationType' to
 'AWS_IAM', 'JWT' or 'CUSTOM'. For Amazon API Gateway V2 managed route overrides with
 'AuthorizationType', set 'AuthorizationType' to 'AWS_IAM', 'JWT' or 'CUSTOM'.
        >>
}

rule api_gw_v2_authorization_type_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %API_GW_MANAGED_OVERRIDE_TYPE) {
    check_api_override(%INPUT_DOCUMENT.
%API_GW_MANAGED_OVERRIDE_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.5]: Require Amazon API Gateway V2 Websocket and HTTP routes
 to specify an authorization type
           [FIX]: For Amazon API Gateway V2 routes, set 'AuthorizationType' to
 'AWS_IAM', 'JWT' or 'CUSTOM'. For Amazon API Gateway V2 managed route overrides with
 'AuthorizationType', set 'AuthorizationType' to 'AWS_IAM', 'JWT' or 'CUSTOM'.
        >>
```

```
}

#
# Parameterized Rules
#
rule check_api_route(api_route) {
    %api_route {
        # Scenario 3
        AuthorizationType exists

        # Scenario 4 and 5
        AuthorizationType in %ALLOWED_AUTHORIZATION_TYPES
    }
}

rule check_api_override(api_override) {
    %api_override [
        # Scenario 2
        Route exists
        Route is_struct
        Route {
            AuthorizationType exists
        }
    ]{
        check_api_route(Route)
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
     %doc.%RESOURCE_TYPE.resourceProperties exists
 }
```

CT.APIGATEWAY.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ApiGatewayV2Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: a1bcdef2gh
      RouteKey: $connect
      AuthorizationType: AWS_IAM
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ApiGatewayManagedOverride:
    Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
    Properties:
      ApiId: a1bcdef2gh
      Route:
        AuthorizationType: AWS_IAM
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ApiGatewayV2Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: a1bcdef2gh
      RouteKey: $connect
      AuthorizationType: NONE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ApiGatewayManagedOverride:
    Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
    Properties:
      ApiId: a1bcdef2gh
      Route:
        AuthorizationType: NONE
```

[CT.APIGATEWAY.PR.6] Require an Amazon API Gateway REST domain to use a security policy that specifies a minimum TLS protocol version of TLSv1.2

This control checks whether an Amazon API Gateway REST API domain name requires a minimum Transport Layer Security protocol version of TLSv1.2 by means of its security policy.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ApiGateway::DomainName`
- **CloudFormation guard rule:** CT.APIGATEWAY.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APIGATEWAY.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APIGATEWAY.PR.6 example templates

Explanation

The TLS protocol addresses network security problems, such as tampering and eavesdropping between a client and server. When your clients establish a TLS handshake to your API through the custom domain, you can choose a minimum Transport Layer Security (TLS) protocol version. This version is enforced for your Amazon API Gateway custom domain by setting a security policy, which is a predefined combination of minimum TLS version and cipher suite offered by Amazon API Gateway.

Usage considerations

- TLS protocol versions and ciphers used by Amazon API Gateway security policies depend on the type of API Gateway endpoint in use. For more about supported TLS protocol versions and ciphers for each endpoint type, review the Amazon API Gateway documentation.

Remediation for rule failure

Set the value of SecurityPolicy to TLS_1_2, or to adopt the default value, do not provide a value for SecurityPolicy.

The examples that follow show how to implement this remediation.

Amazon API Gateway Domain Name - Example

An Amazon API Gateway regional domain name configured with a security policy that requires a minimum of TLSv1.2 for API client connections. The example is shown in JSON and in YAML.

JSON example

```
{
    "DomainName": {
        "Type": "AWS::ApiGateway::DomainName",
        "Properties": {
            "DomainName": "example.com",
            "RegionalCertificateArn": {
                "Ref": "AcmCertificate"
            },
            "EndpointConfiguration": {
                "Types": [
                    "REGIONAL"
```

```
                ]
            },
            "SecurityPolicy": "TLS_1_2"
        }
    }
}
```

YAML example

```
DomainName:
  Type: AWS::ApiGateway::DomainName
  Properties:
    DomainName: example.com
    RegionalCertificateArn: !Ref 'AcmCertificate'
    EndpointConfiguration:
      Types:
        - REGIONAL
    SecurityPolicy: TLS_1_2
```

CT.APIGATEWAY.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   api_gw_domain_tls_check
#
# Description:
#   This control checks whether an Amazon API Gateway REST API domain name requires a
 minimum Transport Layer Security protocol version of TLSv1.2 by means of its security
 policy.
#
# Reports on:
#   AWS::ApiGateway::DomainName
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any API Gateway domain name resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway domain name resource
#       And: 'SecurityPolicy' has been provided and set to a security policy that
 allows
#            a minimum TLS protocol version earlier than TLSv1.2
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway domain name resource
#       And: 'SecurityPolicy' has not been provided
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an API Gateway domain name resource
#       And: 'SecurityPolicy' has been provided and set to a security policy that
 requires
#            a minimum TLS protocol version of TLSv1.2
#      Then: PASS

#
# Constants
#
let API_GW_DOMAIN_NAME_TYPE = "AWS::ApiGateway::DomainName"
let ALLOWED_SECURITY_POLICIES = ["TLS_1_2"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let api_gateway_domain_names = Resources.*[ Type == %API_GW_DOMAIN_NAME_TYPE ]

#
# Primary Rules
#
```

```
rule api_gw_domain_tls_check when is_cfn_template(%INPUT_DOCUMENT)
                                  %api_gateway_domain_names not empty {
    check(%api_gateway_domain_names.Properties)
        <<
        [CT.APIGATEWAY.PR.6]: Require an Amazon API Gateway REST domain to use a
 security policy that specifies a minimum TLS protocol version of TLSv1.2
        [FIX]: Set the value of SecurityPolicy to TLS_1_2, or to adopt the default
 value, do not provide a value for SecurityPolicy.
        >>
}

rule api_gw_domain_tls_check when is_cfn_hook(%INPUT_DOCUMENT,
 %API_GW_DOMAIN_NAME_TYPE) {
    check(%INPUT_DOCUMENT.%API_GW_DOMAIN_NAME_TYPE.resourceProperties)
        <<
        [CT.APIGATEWAY.PR.6]: Require an Amazon API Gateway REST domain to use a
 security policy that specifies a minimum TLS protocol version of TLSv1.2
        [FIX]: Set the value of SecurityPolicy to TLS_1_2, or to adopt the default
 value, do not provide a value for SecurityPolicy.
        >>
}

#
# Parameterized Rules
#
rule check(api_gateway_stage) {
    %api_gateway_stage {
        # Scenario 2, 3, 4
        SecurityPolicy not exists or
        SecurityPolicy in %ALLOWED_SECURITY_POLICIES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APIGATEWAY.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DomainName:
    Type: AWS::ApiGateway::DomainName
    Properties:
      DomainName: example.com
      RegionalCertificateArn: arn:aws:acm:us-west-2:123456789012:certificate/abcd1234-
efg5-1234-1234-1234abcdefgh
      EndpointConfiguration:
        Types:
        - REGIONAL
      SecurityPolicy: TLS_1_2
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DomainName:
    Type: AWS::ApiGateway::DomainName
    Properties:
      DomainName: example.com
      RegionalCertificateArn: arn:aws:acm:us-west-2:123456789012:certificate/abcd1234-
efg5-1234-1234-1234abcdefgh
      EndpointConfiguration:
        Types:
        - REGIONAL
      SecurityPolicy: TLS_1_0
```

AWS Certificate Manager controls

Topics

- [CT.ACM.PR.1] Require an AWS Private CA certificate to have a single domain name

[CT.ACM.PR.1] Require an AWS Private CA certificate to have a single domain name

This control checks whether any AWS Certificate Manager (ACM) Private CA certificates have wildcard domain names instead of single domain names.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CertificateManager::Certificate`
- **CloudFormation guard rule:** CT.ACM.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ACM.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ACM.PR.1 example templates

Explanation

AWS Private CA allows you to use wildcards (*) in the domain name, so you can protect several sites in the same domain. This type of certificate presents some risk, because if the private key of a certificate is compromised, all domain and subdomains with the compromised certificate are compromised. We recommend that you use single domain name certificates instead of wildcard certificates to reduce these associated risks.

Remediation for rule failure

Set `DomainName` and each entry within `SubjectAlternativeNames` to a fully qualified domain name (FQDN) that does not contain a wildcard (*).

The examples that follow show how to implement this remediation.

AWS Certificate Manager Private CA Certificate - Example One

AWS Certificate Manager Private CA certificate configured with a single domain and no subject alternative names. The example is shown in JSON and in YAML.

JSON example

```
{
    "Resources": {
        "ACMCertificate": {
            "Type": "AWS::CertificateManager::Certificate",
            "Properties": {
                "CertificateAuthorityArn": "arn:aws:acm-pca:us-
east-1:123456789012:certificate-authority/12345678-1234-1234-1234-123456789012",
                "DomainName": "example.com"
            }
        }
    }
}
```

YAML example

```
Resources:
  ACMCertificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      CertificateAuthorityArn: arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
      DomainName: example.com
```

The examples that follow show how to implement this remediation.

AWS Certificate Manager Private CA Certificate - Example Two

AWS Certificate Manager private CA certificate configured with a single domain and one subject alternative name. The example is shown in JSON and in YAML.

JSON example

```
{
    "Resources": {
        "ACMCertificate": {
            "Type": "AWS::CertificateManager::Certificate",
            "Properties": {
                "CertificateAuthorityArn": "arn:aws:acm-pca:us-
east-1:123456789012:certificate-authority/12345678-1234-1234-1234-123456789012",
                "DomainName": "example.com",
                "SubjectAlternativeNames": [
                    "www.example.com"
                ]
            }
        }
    }
}
```

YAML example

```
Resources:
  ACMCertificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      CertificateAuthorityArn: arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
      DomainName: example.com
      SubjectAlternativeNames:
        - www.example.com
```

CT.ACM.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   acm_certificate_domain_name_check
#
# Description:
#   This control checks whether any AWS Certificate Manager (ACM) Private CA
 certificates have wildcard domain names instead of single domain names.
#
# Reports on:
#   AWS::CertificateManager::Certificate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ACM certificate resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has not been provided
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has been provided
#       And: 'DomainName' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has been provided
#       And: 'SubjectAlternativeNames' has not been provided or provided as an empty
 list
#       And: 'DomainName' has been provided with a string that begins with a wildcard
 character ('*').
```

```
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has been provided
#       And: 'DomainName' has been provided with a string that does not begin with a
 wildcard character ('*').
#       And: 'SubjectAlternativeNames' has been provided as a non-empty list containing
 a string that
#            begins with a wildcard character ('*').
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has been provided
#       And: 'DomainName' has been provided with a string that does not begin with a
 wildcard character ('*').
#       And: 'SubjectAlternativeNames' has not been provided or provided as an empty
 list
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ACM certificate resource
#       And: 'CertificateAuthorityArn' has been provided
#       And: 'DomainName' has been provided with a string that does not begin with a
 wildcard character ('*').
#       And: 'SubjectAlternativeNames' has been provided as a non-empty list where no
 entries are strings that
#            begin with a wildcard character ('*').
#      Then: PASS

#
# Constants
#
let ACM_CERTIFICATE_TYPE = "AWS::CertificateManager::Certificate"
let WILDCARD_DOMAIN_NAME_REGEX_PATTERN = /^(\*\.).*$/
let INPUT_DOCUMENT = this

#
# Assignments
#
let acm_certificates = Resources.*[ Type == %ACM_CERTIFICATE_TYPE ]

#
```

```
# Primary Rules
#
rule acm_certificate_domain_name_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %acm_certificates not empty {
    check(%acm_certificates.Properties)
        <<
        [CT.ACM.PR.1]: Require an AWS Private CA certificate to have a single domain
 name
            [FIX]: Set 'DomainName' and each entry within 'SubjectAlternativeNames' to
 a fully qualified domain name (FQDN) that does not contain a wildcard (*).
        >>
}

rule acm_certificate_domain_name_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ACM_CERTIFICATE_TYPE) {
    check(%INPUT_DOCUMENT.%ACM_CERTIFICATE_TYPE.resourceProperties)
        <<
        [CT.ACM.PR.1]: Require an AWS Private CA certificate to have a single domain
 name
            [FIX]: Set 'DomainName' and each entry within 'SubjectAlternativeNames' to
 a fully qualified domain name (FQDN) that does not contain a wildcard (*).
        >>
}

#
# Parameterized Rules
#
rule check(acm_certificate) {
    %acm_certificate[
        CertificateAuthorityArn exists
    ] {
        # Scenario 2
        DomainName exists
        # Scenario 3 and 4
        check_wildcarded_domain(DomainName)
        check_subject_alternative_names(this)
    }
}

rule check_subject_alternative_names(acm_certificate) {
    %acm_certificate [
        SubjectAlternativeNames exists
        SubjectAlternativeNames is_list
        SubjectAlternativeNames not empty
```

```
    ] {
        SubjectAlternativeNames[*] {
            check_wildcarded_domain(this)
        }
    }
}

rule check_wildcarded_domain(domain) {
    %domain {
        this is_string
        this != %WILDCARD_DOMAIN_NAME_REGEX_PATTERN
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ACM.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ACMCertificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      CertificateAuthorityArn: arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
```

```
      DomainName: example.com
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ACMCertificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      CertificateAuthorityArn: arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
      DomainName: '*.example.com'
```

AWS AppSync controls

Topics

- [CT.APPSYNC.PR.1] Require an AWS AppSync GraphQL API to have logging enabled
- [CT.APPSYNC.PR.2] Require an AWS AppSync GraphQL API to be configured with private visibility
- [CT.APPSYNC.PR.3] Require that an AWS AppSync GraphQL API is not authenticated with API keys
- [CT.APPSYNC.PR.4] Require an AWS AppSync GraphQL API cache to have encryption in transit enabled.
- [CT.APPSYNC.PR.5] Require an AWS AppSync GraphQL API cache to have encryption at rest enabled.

[CT.APPSYNC.PR.1] Require an AWS AppSync GraphQL API to have logging enabled

This control checks whether an AWS AppSync GraphQL API has been configured to send request-level and field-level logs to Amazon CloudWatch Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::AppSync::GraphQLApi`
- **CloudFormation guard rule:** CT.APPSYNC.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APPSYNC.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.APPSYNC.PR.1 example templates

Explanation

AppSync logs are useful for debugging issues related to requests.

Remediation for rule failure

Within `LogConfig`, set `FieldLogLevel` to `ERROR`, `INFO`, `DEBUG`, or `ALL` and set `CloudWatchLogsRoleArn` to the ARN of an AWS IAM role configured to allow AWS AppSync to send logs to Amazon CloudWatch Logs.

The examples that follow show how to implement this remediation.

AWS AppSync GraphQL API - Example

An AWS AppSync GraphQL API configured to send GraphQL operations and tracing to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "GraphQLApi": {
        "Type": "AWS::AppSync::GraphQLApi",
        "Properties": {
            "Name": "SampleApi",
            "AuthenticationType": "AWS_IAM",
            "LogConfig": {
                "FieldLogLevel": "ALL",
                "CloudWatchLogsRoleArn": {
                    "Fn::GetAtt": [
```

```
                        "AppSyncLoggingRole",
                        "Arn"
                    ]
                }
            }
        }
    }
}
```

YAML example

```
GraphQLApi:
  Type: AWS::AppSync::GraphQLApi
  Properties:
    Name: SampleApi
    AuthenticationType: AWS_IAM
    LogConfig:
      FieldLogLevel: ALL
      CloudWatchLogsRoleArn: !GetAtt 'AppSyncLoggingRole.Arn'
```

CT.APPSYNC.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   appsync_logging_enabled_check
#
# Description:
#   This control checks whether an AWS AppSync GraphQL API has been configured to send
 request-level and field-level logs to Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::AppSync::GraphQLApi
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AppSync GraphQL API resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AppSync GraphQL API resource
#       And: 'LogConfig' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AppSync GraphQL API resource
#       And: 'LogConfig' has been provided
#       And: 'FieldLogLevel' in 'LogConfig' has not been provided or provided and set
 to a value other
#            than 'ERROR', 'INFO', 'DEBUG', or 'ALL'
#       And: 'CloudWatchLogsRoleArn' in 'LogConfig' has not been provided or provided
 and set to an empty
#            string or invalid local reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AppSync GraphQL API resource
#       And: 'LogConfig' has been provided
#       And: 'FieldLogLevel' in 'LogConfig' has been provided and set to 'ERROR',
 'INFO', 'DEBUG' or 'ALL'
#       And: 'CloudWatchLogsRoleArn' in 'LogConfig' has not been provided or provided
 and set to an empty
#            string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AppSync GraphQL API resource
#       And: 'LogConfig' has been provided
#       And: 'FieldLogLevel' in 'LogConfig' has not been provided or provided and set
 to a value other
#            than 'ERROR', 'INFO', 'DEBUG', or 'ALL'
#       And: 'CloudWatchLogsRoleArn' in 'LogConfig' has been provided and set to a non-
empty string or valid
```

```
#            local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AppSync GraphQL API resource
#       And: 'LogConfig' has been provided
#       And: 'FieldLogLevel' in 'LogConfig' has been provided and set to 'ERROR',
 'INFO', 'DEBUG', or 'ALL'
#       And: 'CloudWatchLogsRoleArn' in 'LogConfig' has been provided and set to a non-
empty string or valid
#            local reference
#      Then: PASS

#
# Constants
#
let APPSYNC_GRAPHQL_API_TYPE = "AWS::AppSync::GraphQLApi"
let ALLOWED_APPSYNC_LOG_LEVELS = [ "ERROR",  'INFO', 'DEBUG', "ALL" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let appsync_graphql_apis = Resources.*[ Type == %APPSYNC_GRAPHQL_API_TYPE ]

#
# Primary Rules
#
rule appsync_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %appsync_graphql_apis not empty {
    check(%appsync_graphql_apis.Properties)
        <<
        [CT.APPSYNC.PR.1]: Require an AWS AppSync GraphQL API to have logging enabled
        [FIX]: Within 'LogConfig', set 'FieldLogLevel' to 'ALL', 'INFO', 'DEBUG', or
 'ERROR' and set 'CloudWatchLogsRoleArn' to the ARN of an AWS IAM role configured to
 allow AWS AppSync to send logs to Amazon CloudWatch Logs.
        >>
}

rule appsync_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %APPSYNC_GRAPHQL_API_TYPE) {
    check(%INPUT_DOCUMENT.%APPSYNC_GRAPHQL_API_TYPE.resourceProperties)
        <<
        [CT.APPSYNC.PR.1]: Require an AWS AppSync GraphQL API to have logging enabled
```

```
        [FIX]: Within 'LogConfig', set 'FieldLogLevel' to 'ALL', 'INFO', 'DEBUG', or
 'ERROR' and set 'CloudWatchLogsRoleArn' to the ARN of an AWS IAM role configured to
 allow AWS AppSync to send logs to Amazon CloudWatch Logs.
        >>
}

#
# Parameterized Rules
#
rule check(appsync_graphql_api) {
    %appsync_graphql_api {
        # Scenario 2
        LogConfig exists
        LogConfig is_struct

        LogConfig {
            # Scenarios 3, 4, 5 and 6
            FieldLogLevel exists
            FieldLogLevel in %ALLOWED_APPSYNC_LOG_LEVELS

            CloudWatchLogsRoleArn exists
            check_is_string_and_not_empty(CloudWatchLogsRoleArn) or
            check_local_references(%INPUT_DOCUMENT, CloudWatchLogsRoleArn,
 "AWS::IAM::Role")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
```

```
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.APPSYNC.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
      LogConfig:
```

```
        FieldLogLevel: ALL
        CloudWatchLogsRoleArn:
          Fn::GetAtt:
          - AppSyncLoggingRole
          - Arn
  AppSyncLoggingRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - appsync.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: AppSyncLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
```

```
      LogConfig:
        FieldLogLevel: NONE
        CloudWatchLogsRoleArn:
          Fn::GetAtt:
          - AppSyncLoggingRole
          - Arn
  AppSyncLoggingRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - appsync.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: AppSyncLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
```

[CT.APPSYNC.PR.2] Require an AWS AppSync GraphQL API to be configured with private visibility

This control checks whether an AWS AppSync GraphQL API has been configured with private visibility.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::AppSync::GraphQLApi`
- **CloudFormation guard rule:** CT.APPSYNC.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APPSYNC.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APPSYNC.PR.2 example templates

Explanation

If you use Amazon Virtual Private Cloud (Amazon VPC), you can create AWS AppSync Private APIs, which are APIs that are accessible only from a Amazon VPC. With a Private API, you can restrict API access to your internal applications and connect to your GraphQL and Realtime endpoints without exposing data publicly.

Usage considerations

- This control requires AWS AppSync GraphQL APIs to be configured with private API features, so that they are accessible only from a Amazon VPC. If you require your AWS AppSync GraphQL APIs to be accessible from an AWS AppSync public endpoint, do not enable this control.

Remediation for rule failure

Set the Visibility property to PRIVATE.

The examples that follow show how to implement this remediation.

AWS AppSync Private API - Example

An AWS AppSync GraphQL API configured with private visibility. The example is shown in JSON and in YAML.

JSON example

```
{
    "GraphQLApi": {
        "Type": "AWS::AppSync::GraphQLApi",
        "Properties": {
            "Name": "SampleApi",
            "AuthenticationType": "AWS_IAM",
            "Visibility": "PRIVATE"
        }
    }
}
```

YAML example

```
GraphQLApi:
  Type: AWS::AppSync::GraphQLApi
  Properties:
    Name: SampleApi
    AuthenticationType: AWS_IAM
    Visibility: PRIVATE
```

CT.APPSYNC.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   appsync_api_private_visibility_check
#
# Description:
#   This control checks whether an AWS AppSync GraphQL API has been configured with
 private visibility.
#
# Reports on:
#   AWS::AppSync::GraphQLApi
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS AppSync GraphQL API resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'Visibility' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'Visibility' has been provided and set to a value other than 'PRIVATE'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'Visibility' has been provided and set to 'PRIVATE'
#      Then: PASS

#
# Constants
#
let APPSYNC_GRAPHQL_API_TYPE = "AWS::AppSync::GraphQLApi"
let ALLOWED_VISIBILITY_LEVELS = [ "PRIVATE" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let appsync_graphql_apis = Resources.*[ Type == %APPSYNC_GRAPHQL_API_TYPE ]

#
# Primary Rules
#
rule appsync_api_private_visibility_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %appsync_graphql_apis not empty {
    check(%appsync_graphql_apis.Properties)
        <<
```

```
        [CT.APPSYNC.PR.2]: Require an AWS AppSync GraphQL API to be configured with
 private visibility
        [FIX]: Set the Visibility property to PRIVATE.
        >>
}

rule appsync_api_private_visibility_check when is_cfn_hook(%INPUT_DOCUMENT,
 %APPSYNC_GRAPHQL_API_TYPE) {
    check(%INPUT_DOCUMENT.%APPSYNC_GRAPHQL_API_TYPE.resourceProperties)
        <<
        [CT.APPSYNC.PR.2]: Require an AWS AppSync GraphQL API to be configured with
 private visibility
        [FIX]: Set the Visibility property to PRIVATE.
        >>
}

#
# Parameterized Rules
#
rule check(appsync_graphql_api) {
    %appsync_graphql_api {
        # Scenario 2
        Visibility exists
        # Scenarios 3 and 4
        Visibility in %ALLOWED_VISIBILITY_LEVELS
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APPSYNC.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
      Visibility: PRIVATE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
      Visibility: GLOBAL
```

[CT.APPSYNC.PR.3] Require that an AWS AppSync GraphQL API is not authenticated with API keys

This control checks that an AWS AppSync GraphQL API has been configured with an authentication type other than API_KEY authentication.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::AppSync::GraphQLApi`
- **CloudFormation guard rule:** CT.APPSYNC.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APPSYNC.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APPSYNC.PR.3 example templates

Explanation

One way to control throttling for unauthenticated GraphQL endpoints is through the use of API keys. API keys are recommended only for development purposes, or in scenarios where it is safe to expose a public API. If static API keys are stolen, an API can become vulnerable to replay attacks.

Remediation for rule failure

Set the AuthenticationType property to a value other than API_KEY, and ensure no entry in the AdditionalAuthenticationProviders property has an AuthenticationType value of API_KEY.

The examples that follow show how to implement this remediation.

AWS AppSync GraphQL API - Example

An AWS AppSync GraphQL API configured with IAM authorization. The example is shown in JSON and in YAML.

JSON example

```
{
    "GraphQLApi": {
        "Type": "AWS::AppSync::GraphQLApi",
        "Properties": {
            "Name": "SampleApi",
            "AuthenticationType": "AWS_IAM"
        }
    }
}
```

YAML example

```
GraphQLApi:
  Type: AWS::AppSync::GraphQLApi
  Properties:
    Name: SampleApi
    AuthenticationType: AWS_IAM
```

CT.APPSYNC.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   appsync_authorization_check
#
# Description:
#   This control checks that an AWS AppSync GraphQL API has been configured with an
 authentication type other than API_KEY authentication.
#
# Reports on:
#   AWS::AppSync::GraphQLApi
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS AppSync GraphQL API resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'AuthenticationType' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'AuthenticationType' has been provided and is equal to 'API_KEY'
#       And: 'AdditionalAuthenticationProviders' has not been provided or provided as
 an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'AuthenticationType' has been provided and is equal to a value other than
 'API_KEY'
#       And: 'AdditionalAuthenticationProviders' has been provided as a non-empty list
#       And: An entry in 'AdditionalAuthenticationProviders' has 'AuthenticationType'
 equal to 'API_KEY'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'AuthenticationType' has been provided and is equal to a value other than
 'API_KEY'
#       And: 'AdditionalAuthenticationProviders' has not been provided or provided as
 an empty list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API resource
#       And: 'AuthenticationType' has been provided and is equal to a value other than
 'API_KEY'
#       And: 'AdditionalAuthenticationProviders' has been provided as a non-empty list
#       And: No entries in 'AdditionalAuthenticationProviders' have
 'AuthenticationType' equal to 'API_KEY'
#      Then: PASS

#
# Constants
#
let APPSYNC_GRAPHQL_API_TYPE = "AWS::AppSync::GraphQLApi"
let DISALLOWED_AUTHORIZATION_TYPES = [ "API_KEY" ]
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let appsync_graphql_apis = Resources.*[ Type == %APPSYNC_GRAPHQL_API_TYPE ]

#
# Primary Rules
#
rule appsync_authorization_check when is_cfn_template(%INPUT_DOCUMENT)
                                      %appsync_graphql_apis not empty {
    check(%appsync_graphql_apis.Properties)
        <<
        [CT.APPSYNC.PR.3]: Require that an AWS AppSync GraphQL API is not authenticated
 with API keys
        [FIX]: Set the AuthenticationType property to a value other than API_KEY,
 and ensure no entry in the AdditionalAuthenticationProviders property has an
 AuthenticationType value of API_KEY.
        >>
}

rule appsync_authorization_check when is_cfn_hook(%INPUT_DOCUMENT,
 %APPSYNC_GRAPHQL_API_TYPE) {
    check(%INPUT_DOCUMENT.%APPSYNC_GRAPHQL_API_TYPE.resourceProperties)
        <<
        [CT.APPSYNC.PR.3]: Require that an AWS AppSync GraphQL API is not authenticated
 with API keys
        [FIX]: Set the AuthenticationType property to a value other than API_KEY,
 and ensure no entry in the AdditionalAuthenticationProviders property has an
 AuthenticationType value of API_KEY.
        >>
}

#
# Parameterized Rules
#
rule check(appsync_graphql_api) {
    %appsync_graphql_api {
        # Scenarios 2, 3 and 5
        check_authentication_type(this)
    }

    %appsync_graphql_api [
        AdditionalAuthenticationProviders exists
        AdditionalAuthenticationProviders is_list
```

```
        AdditionalAuthenticationProviders not empty
    ] {
        AdditionalAuthenticationProviders[*] {
            # Scenarios 4 and 6
            check_authentication_type(this)
        }
    }
}

rule check_authentication_type(appsync_configuration) {
    %appsync_configuration {
        AuthenticationType exists
        AuthenticationType not in %DISALLOWED_AUTHORIZATION_TYPES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APPSYNC.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
```

```
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
      AdditionalAuthenticationProviders:
      - AuthenticationType: OPENID_CONNECT
        OpenIDConnectConfig:
          Issuer: https://example.com/
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: API_KEY
```

[CT.APPSYNC.PR.4] Require an AWS AppSync GraphQL API cache to have encryption in transit enabled.

This control checks whether an AWS AppSync API cache has encryption in transit enabled.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AppSync::ApiCache`
- **CloudFormation guard rule:** CT.APPSYNC.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APPSYNC.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APPSYNC.PR.4 example templates

Explanation

Enabling this feature ensures that requests between AWS AppSync, the cache, and the data sources (except insecure HTTP data sources) are encrypted at the network level. Because some processing is needed to encrypt and decrypt the data at the endpoints, in-transit encryption can affect performance.

Remediation for rule failure

Set the value of the TransitEncryptionEnabled property to true.

The examples that follow show how to implement this remediation.

AWS AppSync GraphQL API Cache - Example

An AWS AppSync GraphQL API cache configured with encryption in transit enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "GraphQLApiCache": {
        "Type": "AWS::AppSync::ApiCache",
        "Properties": {
            "ApiId": {
                "Fn::GetAtt": "GraphQLApi.ApiId"
```

```
            },
            "Type": "SMALL",
            "ApiCachingBehavior": "FULL_REQUEST_CACHING",
            "Ttl": 1200,
            "TransitEncryptionEnabled": true
        }
    }
}
```

YAML example

```
GraphQLApiCache:
  Type: AWS::AppSync::ApiCache
  Properties:
    ApiId: !GetAtt 'GraphQLApi.ApiId'
    Type: SMALL
    ApiCachingBehavior: FULL_REQUEST_CACHING
    Ttl: 1200
    TransitEncryptionEnabled: true
```

CT.APPSYNC.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   appsync_cache_encryption_in_transit_check
#
# Description:
#   This control checks whether an AWS AppSync API cache has encryption in transit
 enabled.
#
# Reports on:
#   AWS::AppSync::ApiCache
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS AppSync GraphQL API cache
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'TransitEncryptionEnabled' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'TransitEncryptionEnabled' been provided and is equal to a value other
 than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'TransitEncryptionEnabled' been provided and is equal to bool(true)
#      Then: PASS

#
# Constants
#
let APPSYNC_GRAPHQL_API_CACHE_TYPE = "AWS::AppSync::ApiCache"
let INPUT_DOCUMENT = this

#
# Assignments
#
let appsync_graphql_api_caches = Resources.*[ Type == %APPSYNC_GRAPHQL_API_CACHE_TYPE ]

#
# Primary Rules
#
rule appsync_cache_encryption_in_transit_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %appsync_graphql_api_caches not
 empty {
```

```
    check(%appsync_graphql_api_caches.Properties)
        <<
        [CT.APPSYNC.PR.4]: Require an AWS AppSync GraphQL API cache to have encryption
 in transit enabled.
        [FIX]: Set the value of the TransitEncryptionEnabled property to true.
        >>
}

rule appsync_cache_encryption_in_transit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %APPSYNC_GRAPHQL_API_CACHE_TYPE) {
    check(%INPUT_DOCUMENT.%APPSYNC_GRAPHQL_API_CACHE_TYPE.resourceProperties)
        <<
        [CT.APPSYNC.PR.4]: Require an AWS AppSync GraphQL API cache to have encryption
 in transit enabled.
        [FIX]: Set the value of the TransitEncryptionEnabled property to true.
        >>
}

#
# Parameterized Rules
#
rule check(appsync_graphql_api_cache) {
    %appsync_graphql_api_cache {
        # Scenario 2
        TransitEncryptionEnabled exists
        # Scenarios 3 and 4
        TransitEncryptionEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APPSYNC.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
  GraphQLApiCache:
    Type: AWS::AppSync::ApiCache
    Properties:
      ApiId:
        Fn::GetAtt: GraphQLApi.ApiId
      Type: SMALL
      ApiCachingBehavior: FULL_REQUEST_CACHING
      Ttl: 1200
      TransitEncryptionEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
  GraphQLApiCache:
    Type: AWS::AppSync::ApiCache
```

```
    Properties:
      ApiId:
        Fn::GetAtt: GraphQLApi.ApiId
      Type: SMALL
      ApiCachingBehavior: FULL_REQUEST_CACHING
      Ttl: 1200
      TransitEncryptionEnabled: false
```

[CT.APPSYNC.PR.5] Require an AWS AppSync GraphQL API cache to have encryption at rest enabled.

This control checks whether an AWS AppSync API cache has encryption at rest enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AppSync::ApiCache`
- **CloudFormation guard rule:** CT.APPSYNC.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.APPSYNC.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.APPSYNC.PR.5 example templates

Explanation

Data saved to disk from memory during swap operations is encrypted at the cache instance. Protecting data at rest is an important security best practice. It can mitigate the risk associated with unintended data exposure.

Remediation for rule failure

Set the value of the AtRestEncryptionEnabled property to true.

The examples that follow show how to implement this remediation.

AWS AppSync GraphQL API Cache - Example

An AWS AppSync GraphQL API cache configured with encryption at rest enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "GraphQLApiCache": {
        "Type": "AWS::AppSync::ApiCache",
        "Properties": {
            "ApiId": {
                "Fn::GetAtt": "GraphQLApi.ApiId"
            },
            "Type": "SMALL",
            "ApiCachingBehavior": "FULL_REQUEST_CACHING",
            "Ttl": 1200,
            "AtRestEncryptionEnabled": true
        }
    }
}
```

YAML example

```
GraphQLApiCache:
  Type: AWS::AppSync::ApiCache
  Properties:
    ApiId: !GetAtt 'GraphQLApi.ApiId'
    Type: SMALL
    ApiCachingBehavior: FULL_REQUEST_CACHING
    Ttl: 1200
    AtRestEncryptionEnabled: true
```

CT.APPSYNC.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   appsync_cache_encryption_at_rest_check
#
# Description:
#   This control checks whether an AWS AppSync API cache has encryption at rest
 enabled.
#
# Reports on:
#   AWS::AppSync::ApiCache
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS AppSync GraphQL API cache
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'AtRestEncryptionEnabled' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'AtRestEncryptionEnabled' been provided and is equal to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS AppSync GraphQL API cache resource
#       And: 'AtRestEncryptionEnabled' been provided and is equal to bool(true)
#      Then: PASS

#
# Constants
#
```

```
let APPSYNC_GRAPHQL_API_CACHE_TYPE = "AWS::AppSync::ApiCache"
let INPUT_DOCUMENT = this

#
# Assignments
#
let appsync_graphql_api_caches = Resources.*[ Type == %APPSYNC_GRAPHQL_API_CACHE_TYPE ]

#
# Primary Rules
#
rule appsync_cache_encryption_at_rest_check when is_cfn_template(%INPUT_DOCUMENT)
                                                 %appsync_graphql_api_caches not empty
 {
    check(%appsync_graphql_api_caches.Properties)
        <<
        [CT.APPSYNC.PR.5]: Require an AWS AppSync GraphQL API cache to have encryption
 at rest enabled.
        [FIX]: Set the value of the AtRestEncryptionEnabled property to true.
        >>
}

rule appsync_cache_encryption_at_rest_check when is_cfn_hook(%INPUT_DOCUMENT,
 %APPSYNC_GRAPHQL_API_CACHE_TYPE) {
    check(%INPUT_DOCUMENT.%APPSYNC_GRAPHQL_API_CACHE_TYPE.resourceProperties)
        <<
        [CT.APPSYNC.PR.5]: Require an AWS AppSync GraphQL API cache to have encryption
 at rest enabled.
        [FIX]: Set the value of the AtRestEncryptionEnabled property to true.
        >>
}

#
# Parameterized Rules
#
rule check(appsync_graphql_api_cache) {
    %appsync_graphql_api_cache {
        # Scenario 2
        AtRestEncryptionEnabled exists
        # Scenarios 3 and 4
        AtRestEncryptionEnabled == true
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.APPSYNC.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
  GraphQLApiCache:
    Type: AWS::AppSync::ApiCache
    Properties:
      ApiId:
        Fn::GetAtt: GraphQLApi.ApiId
      Type: SMALL
      ApiCachingBehavior: FULL_REQUEST_CACHING
      Ttl: 1200
      AtRestEncryptionEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GraphQLApi:
    Type: AWS::AppSync::GraphQLApi
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      AuthenticationType: AWS_IAM
  GraphQLApiCache:
    Type: AWS::AppSync::ApiCache
    Properties:
      ApiId:
        Fn::GetAtt: GraphQLApi.ApiId
      Type: SMALL
      ApiCachingBehavior: FULL_REQUEST_CACHING
      Ttl: 1200
      AtRestEncryptionEnabled: false
```

Amazon Athena controls

Topics

- [CT.ATHENA.PR.2] Require an Amazon Athena workgroup to encrypt Athena query results at rest with an AWS Key Management Service (KMS) key

[CT.ATHENA.PR.2] Require an Amazon Athena workgroup to encrypt Athena query results at rest with an AWS Key Management Service (KMS) key

This control checks whether an Amazon Athena workgroup is configured to encrypt query results at rest with an AWS KMS key.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Athena::WorkGroup`

- **CloudFormation guard rule:** CT.ATHENA.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ATHENA.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ATHENA.PR.2 example templates

Explanation

For an added layer of security, you can encrypt the results of Athena queries in the workgroup with AWS Key Management Service (KMS).

Usage considerations

- This control requires an Athena workgroup to override client settings by requiring the `EnforceWorkGroupConfiguration` property to be provided and set to true, or omitted to adopt the default value of true.

Remediation for rule failure

In the `WorkGroupConfiguration.ResultConfiguration` parameter, provide an `EncryptionConfiguration` configuration with an `EncryptionOption` set to a KMS-based encryption option, and with `KmsKey` set to the identifier or ARN of an AWS KMS key, or the name of an AWS KMS key alias.

The examples that follow show how to implement this remediation.

Amazon Athena workgroup - Example

Amazon Athena workgroup configured to encrypt Athena query results with AWS KMS (SSE_KMS). The example is shown in JSON and in YAML.

JSON example

```
{
    "AthenaWorkGroup": {
        "Type": "AWS::Athena::WorkGroup",
        "Properties": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-example"
            },
            "Description": "Example workgroup",
            "State": "ENABLED",
            "WorkGroupConfiguration": {
                "EnforceWorkGroupConfiguration": true,
                "ResultConfiguration": {
                    "EncryptionConfiguration": {
                        "KmsKey": {
                            "Ref": "Key"
                        },
                        "EncryptionOption": "SSE_KMS"
                    }
                }
            }
        }
    }
}
```

YAML example

```
AthenaWorkGroup:
  Type: AWS::Athena::WorkGroup
  Properties:
    Name: !Sub '${AWS::StackName}-example'
    Description: Example workgroup
    State: ENABLED
    WorkGroupConfiguration:
      EnforceWorkGroupConfiguration: true
      ResultConfiguration:
        EncryptionConfiguration:
          KmsKey: !Ref 'Key'
          EncryptionOption: SSE_KMS
```

CT.ATHENA.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   athena_workgroup_results_encrypted_at_rest_kms_check
#
# Description:
#   This control checks whether an Amazon Athena workgroup is configured to encrypt
 query results at rest with an AWS KMS key.
#
# Reports on:
#   AWS::Athena::WorkGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Athena workgroup resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has been
 provided and
#            set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has not been
 provided or provided
#            and set to bool(true)
#       And: 'EncryptionConfiguration' in 'WorkGroupConfiguration.ResultConfiguration'
 has not been provided
#      Then: FAIL
```

```
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has not been
 provided or provided
#            and set to bool(true)
#       And: 'EncryptionConfiguration' in 'WorkGroupConfiguration.ResultConfiguration'
 has been provided
#       And: 'EncryptionOption' in 'EncryptionConfiguration' has not been provided or
 provided as an empty string
#       And: 'KmsKey' in 'EncryptionConfiguration' has not been provided or provided as
 an empty string or
#            invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has not been
 provided or provided
#            and set to bool(true)
#       And: 'EncryptionConfiguration' in 'WorkGroupConfiguration.ResultConfiguration'
 has been provided
#       And: 'EncryptionOption' in 'EncryptionConfiguration' has been provided as a
 non-empty string
#       And: 'KmsKey' in 'EncryptionConfiguration' has not been provided or provided as
 an empty string or
#            invalid local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has not been
 provided or provided
#            and set to bool(true)
#       And: 'EncryptionConfiguration' in 'WorkGroupConfiguration.ResultConfiguration'
 has been provided
#       And: 'EncryptionOption' in 'EncryptionConfiguration' has not been provided or
 provided as an empty string
#       And: 'KmsKey' in 'EncryptionConfiguration' has been provided as a non-empty
 string or valid local reference to
#            a KMS key or key alias
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an Athena workgroup resource
#       And: 'EnforceWorkGroupConfiguration' in 'WorkGroupConfiguration' has not been
 provided or provided
#            and set to bool(true)
#       And: 'EncryptionConfiguration' in 'WorkGroupConfiguration.ResultConfiguration'
 has been provided
#       And: 'EncryptionOption' in 'EncryptionConfiguration' has been provided as a
 non-empty string
#       And: 'KmsKey' in 'EncryptionConfiguration' has been provided as a non-empty
 string or valid local reference to
#            a KMS key or key alias
#      Then: PASS

#
# Constants
#
let ATHENA_WORKGROUP_TYPE = "AWS::Athena::WorkGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let athena_workgroups = Resources.*[ Type == %ATHENA_WORKGROUP_TYPE ]

#
# Primary Rules
#
rule athena_workgroup_results_encrypted_at_rest_kms_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                                %athena_workgroups not
 empty {
    check(%athena_workgroups.Properties)
        <<
        [CT.ATHENA.PR.2]: Require an Amazon Athena workgroup to encrypt Athena query
 results at rest with an AWS Key Management Service (KMS) key
        [FIX]: In the 'WorkGroupConfiguration.ResultConfiguration' parameter, provide
 an 'EncryptionConfiguration' configuration with an 'EncryptionOption' set to a KMS-
based encryption option, and with 'KmsKey' set to the identifier or ARN of an AWS KMS
 key, or the name of an AWS KMS key alias.
        >>
}

rule athena_workgroup_results_encrypted_at_rest_kms_check when
 is_cfn_hook(%INPUT_DOCUMENT, %ATHENA_WORKGROUP_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%ATHENA_WORKGROUP_TYPE.resourceProperties)
        <<
        [CT.ATHENA.PR.2]: Require an Amazon Athena workgroup to encrypt Athena query
 results at rest with an AWS Key Management Service (KMS) key
        [FIX]: In the 'WorkGroupConfiguration.ResultConfiguration' parameter, provide
 an 'EncryptionConfiguration' configuration with an 'EncryptionOption' set to a KMS-
based encryption option, and with 'KmsKey' set to the identifier or ARN of an AWS KMS
 key, or the name of an AWS KMS key alias.
        >>
}

#
# Parameterized Rules
#
rule check(athena_workgroup) {
    %athena_workgroup {
        WorkGroupConfiguration exists
        WorkGroupConfiguration is_struct

        WorkGroupConfiguration {
            # Scenario 2
            EnforceWorkGroupConfiguration not exists or
            EnforceWorkGroupConfiguration == true

            ResultConfiguration exists
            ResultConfiguration is_struct
            ResultConfiguration {
                # Scenario 3
                EncryptionConfiguration exists
                EncryptionConfiguration is_struct

                EncryptionConfiguration {
                    # Scenarios 4, 5, 6 and 7
                    EncryptionOption exists
                    check_is_string_and_not_empty(EncryptionOption)

                    KmsKey exists
                    check_is_string_and_not_empty(KmsKey) or
                    check_local_references(%INPUT_DOCUMENT, KmsKey, "AWS::KMS::Key") or
                    check_local_references(%INPUT_DOCUMENT, KmsKey, "AWS::KMS::Alias")
                }
            }
        }
    }
```

```
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ATHENA.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM user permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeySpec: SYMMETRIC_DEFAULT
  AthenaWorkGroup:
    Type: AWS::Athena::WorkGroup
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      Description: Example workgroup
      State: ENABLED
      WorkGroupConfiguration:
        EnforceWorkGroupConfiguration: true
        ResultConfiguration:
          EncryptionConfiguration:
            KmsKey:
              Ref: Key
            EncryptionOption: SSE_KMS
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  AthenaWorkGroup:
    Type: AWS::Athena::WorkGroup
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      Description: Example workgroup
      State: ENABLED
      WorkGroupConfiguration:
        ResultConfiguration:
          EncryptionConfiguration:
            EncryptionOption: SSE_S3
```

Amazon CloudFront controls

Topics

- [CT.CLOUDFRONT.PR.1] Require an Amazon CloudFront distribution to have a default root object configured
- [CT.CLOUDFRONT.PR.2] Require any Amazon CloudFront distributions with Amazon S3 backed origins to have an origin access identity configured
- [CT.CLOUDFRONT.PR.3] Require an Amazon CloudFront distribution to have encryption in transit configured
- [CT.CLOUDFRONT.PR.4] Require an Amazon CloudFront distribution to have origin failover configured
- [CT.CLOUDFRONT.PR.5] Require any Amazon CloudFront distribution to have logging enabled
- [CT.CLOUDFRONT.PR.6] Require an Amazon CloudFront distribution to use custom SSL/TLS certificates
- [CT.CLOUDFRONT.PR.7] Require an Amazon CloudFront distribution to use SNI to serve HTTPS requests
- [CT.CLOUDFRONT.PR.8] Require an Amazon CloudFront distribution to encrypt traffic to custom origins

- [CT.CLOUDFRONT.PR.9] Require an Amazon CloudFront distribution to have a security policy of TLSv1.2 as a minimum
- [CT.CLOUDFRONT.PR.10] Require any Amazon CloudFrontdistributions with Amazon S3 backed origins to have origin access control configured
- [CT.CLOUDFRONT.PR.11] Require an Amazon CloudFront distribution to use updated SSL protocols between edge locations and custom origins

[CT.CLOUDFRONT.PR.1] Require an Amazon CloudFront distribution to have a default root object configured

This control checks whether an Amazon CloudFront distribution is configured to return a specific object that is the default root object.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.1 example templates

Explanation

A user could possibly request a distribution's root URL instead of an object in the distribution. In this situation, specifying a default root object can help you to avoid exposing the contents of your web distribution.

Remediation for rule failure

Specify a default root object in the `DefaultRootObject` property.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured with a default root object. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "DefaultRootObject": "index.html"
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
```

```
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      DefaultRootObject: index.html
```

CT.CLOUDFRONT.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_default_root_object_configured_check
#
# Description:
#   This control checks whether an Amazon CloudFront distribution is configured to
 return a specific object that is the default root object.
#
# Reports on:
#    AWS::CloudFront::Distribution
#
# Evaluates:
#    CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
```

```
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DefaultRootObject' is not present on the CloudFront distribution resource
 or is present and
#            is an empty string
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DefaultRootObject' is present on the CloudFront distribution resource and
 is a non-empty string
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_default_root_object_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %cloudfront_distributions not
 empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.1]: Require an Amazon CloudFront distribution to have a
 default root object configured
            [FIX]: Specify a default root object in the 'DefaultRootObject' property.
        >>
}

rule cloudfront_default_root_object_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
```

```
        [CT.CLOUDFRONT.PR.1]: Require an Amazon CloudFront distribution to have a
 default root object configured
            [FIX]: Specify a default root object in the 'DefaultRootObject' property.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            # Scenario 2
            DefaultRootObject exists
            # Scenario 3
            check_is_string_and_not_empty(DefaultRootObject)
        }
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
         ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

```
        DefaultRootObject: index.html
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.2] Require any Amazon CloudFront distributions with Amazon S3 backed origins to have an origin access identity configured

This control checks whether Amazon CloudFront distributions backed by Amazon S3 are configured with an origin access identity.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see: CT.CLOUDFRONT.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.2 example templates

Explanation

CloudFront OAI prevents users from gaining direct access to Amazon S3 bucket content. With direct access to an Amazon S3 bucket, a user bypasses the CloudFront distribution and any permissions that are applied to the underlying S3 bucket content.

Usage considerations

- This control applies only to Amazon CloudFront distributions that are configured with one or more origins that are backed by Amazon S3.

Remediation for rule failure

Configure Amazon S3 backed origins by means of the `Origins` property. For each origin backed by Amazon S3, configure an origin access identity by means of the `OriginAccessIdentity` property within an `S3OriginConfig` configuration.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution with an Amazon S3 bucket origin and origin access identity. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleS3Origin",
                        "DomainName": {
                            "Fn::GetAtt": [
                                "OriginBucket",
                                "RegionalDomainName"
                            ]
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": {
                                "Fn::Join": [
                                    "",
                                    [
                                        "origin-access-identity/cloudfront/",
                                        {
                                            "Ref": "OriginBucketOai"
                                        }
                                    ]
                                ]
                            }
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleS3Origin",
                    "CachePolicyId": {
```

```
                        "Ref": "CachePolicy"
                    }
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleS3Origin
          DomainName: !GetAtt 'OriginBucket.RegionalDomainName'
          S3OriginConfig:
            OriginAccessIdentity: !Join
              - ''
              - - origin-access-identity/cloudfront/
                - !Ref 'OriginBucketOai'
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleS3Origin
        CachePolicyId: !Ref 'CachePolicy'
```

CT.CLOUDFRONT.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_origin_access_identity_enabled_check
#
# Description:
```

```
#   This control checks whether Amazon CloudFront distributions backed by Amazon S3 are
 configured with an origin access identity.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: No S3 backed 'Origins' are provided on the CloudFront distribution
 resource or 'Origins' is not present on
#            the CloudFront distribution resource or is present and an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'S3Origin' is present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more S3 backed 'Origins' are configured on the CloudFront
 distribution resource
#       And: 'OriginAccessIdentity' is not present or is an empty string in the
 'S3OriginConfig' property or invalid
#            local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more S3 backed 'Origins' are provided on the CloudFront
 distribution resource
```

```
#       And: 'S3OriginConfig' is present with an 'OriginAccessIdentity for each S3
 backed 'Origin' on the
#            CloudFront distribution resource that is a non-empty string or valid local
 reference
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let S3_BUCKET_DNS_NAME_PATTERN = /(.*)\.s3(-external-\d|[-\.][a-z]*-[a-z]*-[0-9])?
\.amazonaws\.com(\.cn)?$/
let INPUT_DOCUMENT = this
#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_origin_access_identity_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %cloudfront_distributions not
 empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.2]: Require any Amazon CloudFront distributions with Amazon
 S3 backed origins to have an origin access identity configured
            [FIX]: Configure Amazon S3 backed origins by means of the 'Origins'
 property. For each origin backed by Amazon S3, configure an origin access identity by
 means of the 'OriginAccessIdentity' property within an 'S3OriginConfig' configuration.
        >>
}

rule cloudfront_origin_access_identity_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.2]: Require any Amazon CloudFront distributions with Amazon
 S3 backed origins to have an origin access identity configured
            [FIX]: Configure Amazon S3 backed origins by means of the 'Origins'
 property. For each origin backed by Amazon S3, configure an origin access identity by
 means of the 'OriginAccessIdentity' property within an 'S3OriginConfig' configuration.
```

```
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution[
        filter_cloudfront_distribution_with_legacy_s3_origins(this)
    ] {
        DistributionConfig {
            # Scenario 3
            S3Origin not exists
        }
    }

    %cloudfront_distribution[
        # Scenario 2
        filter_cloudfront_distribution_with_origins(this)
    ] {
        DistributionConfig {
            # Scenario 4
            Origins [
                DomainName == %S3_BUCKET_DNS_NAME_PATTERN or
                check_origin_domain_name_get_att(DomainName)
            ] {
                S3OriginConfig exists
                S3OriginConfig is_struct
                S3OriginConfig {
                    # Scenario 3 and 5
                    OriginAccessIdentity exists
                    check_is_string_and_not_empty(OriginAccessIdentity) or
                    check_local_oai(OriginAccessIdentity)
                }
            }
        }
    }
}

rule check_origin_domain_name_get_att(domain) {
  %domain {
    'Fn::GetAtt' {
        this is_list
        this not empty
```

```
        this[1] == "DomainName" or
        this[1] == "RegionalDomainName"
    }
    check_local_references(%INPUT_DOCUMENT, this, "AWS::S3::Bucket")
  }
}

rule check_local_oai(oai) {
    %oai {
        'Fn::Join' {
            this[1] exists
            this[1] is_list
            this[1] not empty
            some this[1].* {
                check_local_references(%INPUT_DOCUMENT, this,
 "AWS::CloudFront::CloudFrontOriginAccessIdentity")
            }
        } or
        'Fn::Sub' {
            when this is_list {
                this[1] exists
                this[1] is_struct
                some this[1].* {
                   check_local_references(%INPUT_DOCUMENT, this,
 "AWS::CloudFront::CloudFrontOriginAccessIdentity")
                }
            }
            when this is_string {
                check_is_string_and_not_empty(this)
            }
        }
    }
}

rule filter_cloudfront_distribution_with_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            Origins exists
            Origins is_list
            Origins not empty
        }
```

```
    }
}

rule filter_cloudfront_distribution_with_legacy_s3_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            S3Origin exists
        }
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
```

```
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CLOUDFRONT.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  OriginBucketOai:
    Type: AWS::CloudFront::CloudFrontOriginAccessIdentity
    Properties:
      CloudFrontOriginAccessIdentityConfig:
        Comment:
```

```
        Fn::Sub: ${AWS::StackName}-example-oai
  OriginBucket:
    Type: AWS::S3::Bucket
  OriginBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: OriginBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetObject'
            Effect: Allow
            Resource:
              Fn::Join:
              - ''
              - - 'arn:aws:s3:::'
                - Ref: OriginBucket
                - /*
            Principal:
              AWS:
                Fn::Join:
                - ''
                - - 'arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity '
                  - Ref: OriginBucketOai
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleS3Origin
          DomainName:
            Fn::GetAtt:
            - OriginBucket
            - RegionalDomainName
          S3OriginConfig:
            OriginAccessIdentity:
              Fn::Join:
              - ""
              - - "origin-access-identity/cloudfront/"
                - Ref: OriginBucketOai
        DefaultCacheBehavior:
```

```
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleS3Origin
          CachePolicyId:
            Ref: CachePolicy
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleS3Origin
          DomainName: amzn-s3-demo-bucket.s3.amazonaws.com
          S3OriginConfig: {}
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleS3Origin
          CachePolicyId:
            Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.3] Require an Amazon CloudFront distribution to have encryption in transit configured

This control checks whether your Amazon CloudFront distributions use HTTPS, either directly or through a redirection.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.3 example templates

Explanation

HTTPS (TLS) can help prevent potential attackers from attempting person-in-the-middle or similar attacks, which can eavesdrop on or manipulate network traffic. Only encrypted connections over HTTPS (TLS) should be allowed. Encrypting data in transit can affect performance. We recommend that you test your application with this feature to understand the performance profile and the impact of TLS.

Remediation for rule failure

Set `ViewerProtocolPolicy` in `DefaultCacheBehavior` and `CacheBehavior` to `https-only` or `redirect-to-https`.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example One

Amazon CloudFront distribution configured with a default cache behavior that requires viewer connections to use HTTPS. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
```

```
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
```

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example Two

Amazon CloudFront distribution configured with a cache behavior that redirects viewer HTTP connections to HTTPS. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
```

```
                },
                "CacheBehaviors": [
                    {
                        "ViewerProtocolPolicy": "redirect-to-https",
                        "TargetOriginId": "sampleOrigin",
                        "PathPattern": "*"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      CacheBehaviors:
        - ViewerProtocolPolicy: redirect-to-https
          TargetOriginId: sampleOrigin
          PathPattern: '*'
```

CT.CLOUDFRONT.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   cloudfront_viewer_policy_https_check
#
# Description:
#   This control checks whether your Amazon CloudFront distributions use HTTPS, either
 directly or through a redirection.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.DefaultCacheBehavior' is missing on the CloudFront
 distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.DefaultCacheBehavior' is present on the CloudFront
 distribution resource
#       And: 'ViewerProtocolPolicy' in 'DefaultCacheBehavior' is missing or set to a
 value other than 'https-only' or
#            'redirect-to-https' (e.g. 'allow-all')
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.CacheBehavior' is provided on the CloudFront
 distribution resource
```

```
#       And: 'ViewerProtocolPolicy' in the 'CacheBehavior' is  is missing or set to a
 value other than 'https-only' or
#            'redirect-to-https' (e.g. 'allow-all')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.DefaultCacheBehavior' is present on the CloudFront
 distribution resource
#       And: 'ViewerProtocolPolicy' in 'DefaultCacheBehavior' is set to 'https-only' or
 'redirect-to-https'
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.CacheBehavior' are provided on the CloudFront
 distribution resource as a non-empty list
#       And: 'ViewerProtocolPolicy' in the 'CacheBehavior' is set to 'https-only' or
 'redirect-to-https'
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let ALLOWED_VIEWER_PROTOCOL_POLICIES = [ "https-only", "redirect-to-https" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_viewer_policy_https_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %cloudfront_distributions not empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.3]: Require an Amazon CloudFront distribution to have
 encryption in transit configured
            [FIX]: Set 'ViewerProtocolPolicy' in 'DefaultCacheBehavior' and
 'CacheBehavior' to 'https-only' or 'redirect-to-https'.
```

```
        >>
}

rule cloudfront_viewer_policy_https_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.3]: Require an Amazon CloudFront distribution to have
 encryption in transit configured
            [FIX]: Set 'ViewerProtocolPolicy' in 'DefaultCacheBehavior' and
 'CacheBehavior' to 'https-only' or 'redirect-to-https'.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            DefaultCacheBehavior exists
            DefaultCacheBehavior is_struct

            DefaultCacheBehavior {
                # Scenarios 2 and 4
                check_viewer_protocol_policy(this)
            }

            when CacheBehaviors exists
                 CacheBehaviors is_list
                 CacheBehaviors not empty {

                    CacheBehaviors[*] {
                        # Scenarios 3 and 5
                        check_viewer_protocol_policy(this)
                    }
            }
        }
    }
}
```

```
rule check_viewer_protocol_policy(cache_behaviour) {
    %cache_behaviour {
        ViewerProtocolPolicy exists
        ViewerProtocolPolicy in %ALLOWED_VIEWER_PROTOCOL_POLICIES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
```

```
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
```

```
          HeaderBehavior: none
        QueryStringsConfig:
          QueryStringBehavior: none
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
      - Id: exampleOrigin
        DomainName: example.com
        CustomOriginConfig:
          OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: allow-all
        TargetOriginId: exampleOrigin
        CachePolicyId:
          Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.4] Require an Amazon CloudFront distribution to have origin failover configured

This control checks whether your Amazon CloudFront distribution is configured with an origin group that contains two origin group members.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.4 example templates

Explanation

CloudFront origin failover can increase availability. Origin failover automatically redirects traffic to a secondary origin if the primary origin is unavailable or if it returns specific HTTP response status codes.

Remediation for rule failure

Configure an origin group on the Amazon CloudFront Distribution with two origin group members.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example One

Amazon CloudFront distribution configured with an origin group that contains two origin group members. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "one.example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    },
                    {
                        "Id": "sampleOrigin2",
                        "DomainName": "two.example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
```

```
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "OriginGroups": {
                    "Quantity": 1,
                    "Items": [
                        {
                            "Id": "ExampleOriginGroup",
                            "FailoverCriteria": {
                                "StatusCodes": {
                                    "Items": [
                                        400
                                    ],
                                    "Quantity": 1
                                }
                            },
                            "Members": {
                                "Quantity": 2,
                                "Items": [
                                    {
                                        "OriginId": "sampleOrigin"
                                    },
                                    {
                                        "OriginId": "sampleOrigin2"
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
```

```
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: one.example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        - Id: sampleOrigin2
          DomainName: two.example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      OriginGroups:
        Quantity: 1
        Items:
          - Id: ExampleOriginGroup
            FailoverCriteria:
              StatusCodes:
                Items:
                  - 400
                Quantity: 1
            Members:
              Quantity: 2
              Items:
                - OriginId: sampleOrigin
                - OriginId: sampleOrigin2
```

CT.CLOUDFRONT.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   cloudfront_origin_failover_enabled_check
```

```
#
# Description:
#   This control checks whether your Amazon CloudFront distribution is configured with
 an origin group that contains two origin group members.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'OriginGroups' is not present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'OriginGroups' is present on the CloudFront distribution resource
#       And: 'Quantity' within 'OriginGroups' is 0
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'OriginGroups' is present on the CloudFront distribution resource
#       And: 'Quantity' within 'OriginGroups' is >= 1
#       And: 'Quantity' within 'Members' is < 2
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'OriginGroups' is present on the CloudFront distribution resource
#       And: 'Quantity' within 'OriginGroups' is >= 1
#       And: 'Quantity' within 'Members' is == 2
#      Then: PASS
```

```
#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_origin_failover_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                   %cloudfront_distributions not empty
 {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.4]: Require an Amazon CloudFront distribution to have origin
 failover configured
            [FIX]: Configure an origin group on the Amazon CloudFront Distribution with
 two origin group members.
        >>
}

rule cloudfront_origin_failover_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.4]: Require an Amazon CloudFront distribution to have origin
 failover configured
            [FIX]: Configure an origin group on the Amazon CloudFront Distribution with
 two origin group members.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
```

```
        DistributionConfig is_struct

        DistributionConfig {
            # Scenario 2
            OriginGroups exists
            OriginGroups is_struct

            OriginGroups {
                # Scenario 3
                Quantity exists
                Quantity >= 1

                Items exists
                Items is_list
                Items not empty

                Items[*] {
                    Members exists
                    Members is_struct
                    Members {
                        # Scenarios 4 and 5
                        Quantity == 2
                    }
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: one.example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        - Id: exampleOrigin2
          DomainName: two.example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
```

```
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        OriginGroups:
          Quantity: 1
          Items:
          - Id: ExampleOriginGroup
            FailoverCriteria:
              StatusCodes:
                Items:
                - 400
                Quantity: 1
            Members:
              Quantity: 2
              Items:
              - OriginId: exampleOrigin
              - OriginId: exampleOrigin2
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
```

```
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.5] Require any Amazon CloudFront distribution to have logging enabled

This control checks whether Amazon CloudFront distributions are configured with access logging.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.5 example templates

Explanation

CloudFront access logs provide detailed information about every user request that CloudFront receives. Each log contains information such as the date and time the request was received, the IP address of the viewer that made the request, the source of the request, and the port number of the request from the viewer.

These access logs are useful for applications such as security and access audits, and in forensic investigation.

Remediation for rule failure

Set `Bucket` in `DistributionConfig.Logging` to an Amazon S3 bucket that has been configured to receive Amazon CloudFront distribution access logs.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured with access logging enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
```

```
                    }
                },
                "Logging": {
                    "Bucket": {
                        "Fn::GetAtt": [
                            "LoggingBucket",
                            "RegionalDomainName"
                        ]
                    }
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      Logging:
        Bucket: !GetAtt 'LoggingBucket.RegionalDomainName'
```

CT.CLOUDFRONT.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   cloudfront_access_logs_enabled_check
#
# Description:
#   This control checks whether Amazon CloudFront distributions are configured with
 access logging.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.Logging.Bucket' configuration is not present on the
 CloudFront distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.Logging' configuration is present on the CloudFront
 distribution resource
#       And: 'Bucket' has been provided in the 'DistributionConfig.Logging'
 configuration with with an empty string or
#            invalid local reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.Logging' configuration is present on the CloudFront
 distribution resource
```

```
#       And: A 'Bucket' property has been provided within the
 'DistributionConfig.Logging' configuration with a
#            non-empty string or valid local stack reference
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_access_logs_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %cloudfront_distributions not empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.5]: Require any Amazon CloudFront distribution to have
 logging enabled
            [FIX]: Set 'Bucket' in 'DistributionConfig.Logging' to an Amazon S3 bucket
 that has been configured to receive Amazon CloudFront distribution access logs.
        >>
}

rule cloudfront_access_logs_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.5]: Require any Amazon CloudFront distribution to have
 logging enabled
            [FIX]: Set 'Bucket' in 'DistributionConfig.Logging' to an Amazon S3 bucket
 that has been configured to receive Amazon CloudFront distribution access logs.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            Logging exists
            Logging is_struct
            Logging {
                # Scenario 2
                Bucket exists

                # Scenarios 3 and 4
                check_is_string_and_not_empty(Bucket) or
                check_local_references(%INPUT_DOCUMENT, Bucket, %S3_BUCKET_TYPE)
            }
        }
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
```

```
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CLOUDFRONT.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
```

```
        QueryStringsConfig:
          QueryStringBehavior: none
  LoggingBucket:
    Type: AWS::S3::Bucket
    Properties: {}
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
              - 's3:PutBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
              - ''
              - - 'arn:aws:s3:::'
                - Ref: LoggingBucket
            Principal:
              AWS:
                Ref: AWS::AccountId
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        Logging:
          Bucket:
            Fn::GetAtt:
            - LoggingBucket
```

```
            - RegionalDomainName
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.6] Require an Amazon CloudFront distribution to use custom SSL/TLS certificates

This control checks whether the certificate associated with an Amazon CloudFront distribution is a custom SSL/TLS certificate.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.6 example templates

Explanation

Custom SSL/TLS certificates give your users access to content by using alternate domain names. You can store custom certificates in AWS Certificate Manager (recommended), or in IAM.

Usage considerations

- This control requires a viewer certificate configuration compatible only with Amazon CloudFront distributions that use `Aliases`, also known as alternate domain names or CNAMEs.

Remediation for rule failure

Provide a `ViewerCertificate` configuration with values for `AcmCertificateArn`, `MinimumProtocolVersion`, and `SslSupportMethod`.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured with an AWS Certificate Manager SSL certificate. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "ViewerCertificate": {
                    "AcmCertificateArn": {
                        "Ref": "ACMCertificate"
                    },
                    "MinimumProtocolVersion": "TLSv1.2_2021",
                    "SslSupportMethod": "sni-only"
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      ViewerCertificate:
        AcmCertificateArn: !Ref 'ACMCertificate'
        MinimumProtocolVersion: TLSv1.2_2021
        SslSupportMethod: sni-only
```

CT.CLOUDFRONT.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_custom_ssl_certificate_check
#
# Description:
#   This control checks whether the certificate associated with an Amazon CloudFront
 distribution is a custom SSL/TLS certificate.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is not present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'CloudFrontDefaultCertificate' is set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: One of 'AcmCertificateArn' or 'IamCertificateId' are not provided or
 provided as empty strings or invalid
#            local references
#       And: One of 'MinimumProtocolVersion' and 'SslSupportMethod' is not provided or
 provided as an empty string
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'AcmCertificateArn' or 'IamCertificateId' are provided in the
 'ViewerCertificate' configuration as
#             non-empty strings or 'AcmCertificateArn' is a valid local reference
#       And: 'MinimumProtocolVersion' and 'SslSupportMethod' are provided as non-empty
 strings
#      Then: PASS

#
# Constants
#
```

```
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_custom_ssl_certificate_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %cloudfront_distributions not empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.6]: Require an Amazon CloudFront distribution to use custom
 SSL/TLS certificates
            [FIX]: Provide a 'ViewerCertificate' configuration with values for
 'AcmCertificateArn', 'MinimumProtocolVersion', and 'SslSupportMethod'.
        >>
}

rule cloudfront_custom_ssl_certificate_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.6]: Require an Amazon CloudFront distribution to use custom
 SSL/TLS certificates
            [FIX]: Provide a 'ViewerCertificate' configuration with values for
 'AcmCertificateArn', 'MinimumProtocolVersion', and 'SslSupportMethod'.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            ViewerCertificate exists
            ViewerCertificate is_struct
```

```
            ViewerCertificate {
                CloudFrontDefaultCertificate not exists or
                CloudFrontDefaultCertificate == false

                check_custom_acm_certificate_provided(AcmCertificateArn,
 "AWS::CertificateManager::Certificate") or
                check_custom_iam_certificate_provided(IamCertificateId)

                MinimumProtocolVersion exists
                check_is_string_and_not_empty(MinimumProtocolVersion)

                SslSupportMethod exists
                check_is_string_and_not_empty(SslSupportMethod)
            }
        }
    }
}

rule check_custom_acm_certificate_provided(certificate, cfn_type) {
    %certificate {
        this exists
        check_is_string_and_not_empty(this) or
        check_local_references(%INPUT_DOCUMENT, this, %cfn_type)
    }
}

rule check_custom_iam_certificate_provided(certificate) {
    %certificate {
        this exists
        check_is_string_and_not_empty(this)
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CLOUDFRONT.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
```

```
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        ViewerCertificate:
          AcmCertificateArn:
            Ref: ACMCertificate
          MinimumProtocolVersion: TLSv1.2_2021
          SslSupportMethod: sni-only
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        ViewerCertificate:
          CloudFrontDefaultCertificate: true
```

[CT.CLOUDFRONT.PR.7] Require an Amazon CloudFront distribution to use SNI to serve HTTPS requests

This control checks whether your Amazon CloudFront distributions are configured to use SNI to serve HTTPS requests.

- **Control objective:** Encrypt data in transit, Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.7 example templates

Explanation

Server Name Indication (SNI) is an extension to the TLS protocol. It is supported by browsers and clients released after 2010. If you configure CloudFront to serve HTTPS requests using SNI, CloudFront associates your alternate domain name with an IP address for each edge location. When a viewer submits an HTTPS request for your content, DNS routes the request to the IP address for the correct edge location. The IP address for your domain name is determined during the SSL/TLS handshake negotiation; the IP address isn't dedicated to your distribution.

Usage considerations

- This control requires a viewer certificate configuration which is only compatible with Amazon CloudFront distributions that use `Aliases` (also known as alternate domain names or CNAMEs)

Remediation for rule failure

Within `ViewerCertificate`, set `SslSupportMethod` to `sni-only`, `MinimumProtocolVersion` to a protocol that supports SNI (`TLSv1` or greater), and `AcmCertificateArn` to the ARN of an AWS ACM certificate.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured to use SNI to serve HTTPS requests. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "ViewerCertificate": {
                    "AcmCertificateArn": {
                        "Ref": "ACMCertificate"
                    },
                    "MinimumProtocolVersion": "TLSv1.2_2021",
                    "SslSupportMethod": "sni-only"
```

```
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      ViewerCertificate:
        AcmCertificateArn: !Ref 'ACMCertificate'
        MinimumProtocolVersion: TLSv1.2_2021
        SslSupportMethod: sni-only
```

CT.CLOUDFRONT.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_sni_enabled_check
#
# Description:
```

```
#   This control checks whether your Amazon CloudFront distributions are configured to
 use SNI to serve HTTPS requests.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is not present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'CloudFrontDefaultCertificate' is set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'AcmCertificateArn' or 'IamCertificateId' are provided in the
 'ViewerCertificate' configuration
#       And: 'MinimumProtocolVersion' is provided in the 'ViewerCertificate'
 configuration with a protocol that does not
#            support SNI (SSLv3)
#       Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'AcmCertificateArn' or 'IamCertificateId' are provided in the
 'ViewerCertificate' configuration
```

```
#       And: 'MinimumProtocolVersion' is provided in the 'ViewerCertificate'
 configuration with a protocol that supports
#            SNI (TLSv1 or greater)
#       And: 'SslSupportMethod' is set to 'vip'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'ViewerCertificate' is present on the CloudFront distribution resource
#       And: 'AcmCertificateArn' or 'IamCertificateId' are provided in the
 'ViewerCertificate' configuration
#       And: 'MinimumProtocolVersion' is provided in the 'ViewerCertificate'
 configuration with a protocol that supports
#            SNI (TLSv1 or greater)
#       And: 'SslSupportMethod' is set to 'sni-only'
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let UNSUPPORTED_PROTOCOLS_FOR_SNI = [ "SSLv3" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_sni_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                       %cloudfront_distributions not empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.7]: Require an Amazon CloudFront distribution to use SNI to
 serve HTTPS requests
            [FIX]: Within 'ViewerCertificate', set 'SslSupportMethod' to 'sni-only',
 'MinimumProtocolVersion' to a protocol that supports SNI ('TLSv1' or greater), and
 'AcmCertificateArn' to the ARN of an AWS ACM certificate.
        >>
}
```

```
rule cloudfront_sni_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.7]: Require an Amazon CloudFront distribution to use SNI to
 serve HTTPS requests
            [FIX]: Within 'ViewerCertificate', set 'SslSupportMethod' to 'sni-only',
 'MinimumProtocolVersion' to a protocol that supports SNI ('TLSv1' or greater), and
 'AcmCertificateArn' to the ARN of an AWS ACM certificate.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            ViewerCertificate exists
            ViewerCertificate is_struct

            ViewerCertificate {
                CloudFrontDefaultCertificate not exists or
                CloudFrontDefaultCertificate == false

                check_custom_acm_certificate_provided(AcmCertificateArn,
 "AWS::CertificateManager::Certificate") or
                check_custom_iam_certificate_provided(IamCertificateId)

                MinimumProtocolVersion exists
                MinimumProtocolVersion not in %UNSUPPORTED_PROTOCOLS_FOR_SNI

                SslSupportMethod exists
                SslSupportMethod == "sni-only"
            }
        }
    }
}

rule check_custom_acm_certificate_provided(certificate, cfn_type) {
    %certificate {
```

```
        this exists
        check_is_string_and_not_empty(this) or
        check_local_references(%INPUT_DOCUMENT, this, %cfn_type)
    }
}

rule check_custom_iam_certificate_provided(certificate) {
    %certificate {
        this exists
        check_is_string_and_not_empty(this)
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
```

```
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CLOUDFRONT.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
```

```
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        ViewerCertificate:
          AcmCertificateArn:
            Ref: ACMCertificate
          MinimumProtocolVersion: TLSv1.2_2021
          SslSupportMethod: sni-only
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
```

```
        EnableAcceptEncodingGzip: false
        HeadersConfig:
          HeaderBehavior: none
        QueryStringsConfig:
          QueryStringBehavior: none
ACMCertificate:
  Type: "AWS::CertificateManager::Certificate"
  Properties:
    DomainName: example.com
    ValidationMethod: DNS
    DomainValidationOptions:
      - DomainName: www.example.com
        HostedZoneId: ZZZHHHHWWWWAAA
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
      - Id: exampleOrigin
        DomainName: example.com
        CustomOriginConfig:
          OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: exampleOrigin
        CachePolicyId:
          Ref: CachePolicy
      ViewerCertificate:
        AcmCertificateArn:
          Ref: ACMCertificate
        MinimumProtocolVersion: TLSv1
        SslSupportMethod: vip
```

[CT.CLOUDFRONT.PR.8] Require an Amazon CloudFront distribution to encrypt traffic to custom origins

This control checks whether your Amazon CloudFront distributions are encrypting traffic to custom origins.

- **Control objective:** Encrypt data in transit

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.8 example templates

Explanation

HTTPS (TLS) can help prevent eavesdropping or manipulation of network traffic. Only encrypted connections over HTTPS (TLS) should be allowed.

Usage considerations

- This control applies only to Amazon CloudFront distributions that have one or more origins configured.

Remediation for rule failure

For Amazon CloudFront custom origins, set `OriginProtocolPolicy` to `https-only` or match-viewer. `When setting` OriginProtocolPolicy `to` match-viewer, `do not set` ViewerProtocolPolicy `to` allow-all' for any cache behaviors.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example One

Amazon CloudFront distribution configured to require HTTPS connections to custom origins, by means of an origin protocol policy of `https-only`. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "CacheBehaviors": [
                    {
                        "ViewerProtocolPolicy": "https-only",
                        "TargetOriginId": "sampleOrigin",
                        "PathPattern": "*",
                        "CachePolicyId": {
                            "Ref": "CachePolicy"
                        }
                    }
                ],
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
```

```
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      CacheBehaviors:
        - ViewerProtocolPolicy: https-only
          TargetOriginId: sampleOrigin
          PathPattern: '*'
          CachePolicyId: !Ref 'CachePolicy'
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
```

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example Two

Amazon CloudFront distribution configured to require HTTPS connections to custom origins, by means of an origin protocol policy of `match-viewer`. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
```

```
                },
                "CacheBehaviors": [
                    {
                        "ViewerProtocolPolicy": "https-only",
                        "TargetOriginId": "sampleOrigin",
                        "PathPattern": "*",
                        "CachePolicyId": {
                            "Ref": "CachePolicy"
                        }
                    }
                ],
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "match-viewer"
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      CacheBehaviors:
        - ViewerProtocolPolicy: https-only
          TargetOriginId: sampleOrigin
          PathPattern: '*'
          CachePolicyId: !Ref 'CachePolicy'
```

```
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: match-viewer
```

CT.CLOUDFRONT.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_traffic_to_origin_encrypted_check
#
# Description:
#   This control checks whether your Amazon CloudFront distributions are encrypting
 traffic to custom origins.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIPs
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'Origins' is not present or is an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There are no 'Origins' with a 'CustomOriginConfig'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'CustomOrigin' is present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: At least one 'Origins' with a 'CustomOriginConfig' has an
 'OriginProtocolPolicy' of 'http-only'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: At least one 'Origins' with a 'CustomOriginConfig' has an
 'OriginProtocolPolicy' of 'match-viewer'
#       And: Any 'ViewerProtocolPolicy' is set to 'allow-all' for
 'DefaultCacheBehavior' or any configured
#            'CacheBehaviors'
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: All 'Origins' with a 'CustomOriginConfig' have an 'OriginProtocolPolicy'
 of 'https-only'
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront Distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: At least one 'Origins' with a 'CustomOriginConfig' has an
 'OriginProtocolPolicy' of 'match-viewer'
```

```
#       And: 'ViewerProtocolPolicy' is not set to 'allow-all' for both
 'DefaultCacheBehavior' and any configured
#            'CacheBehaviors'
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_traffic_to_origin_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %cloudfront_distributions not
 empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.8]: Require an Amazon CloudFront distribution to encrypt
 traffic to custom origins
            [FIX]: For Amazon CloudFront custom origins, set 'OriginProtocolPolicy' to
 'https-only' or match-viewer'. When setting 'OriginProtocolPolicy' to 'match-viewer',
 do not set 'ViewerProtocolPolicy' to 'allow-all' for any cache behaviors.
        >>
}

rule cloudfront_traffic_to_origin_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.8]: Require an Amazon CloudFront distribution to encrypt
 traffic to custom origins
            [FIX]: For Amazon CloudFront custom origins, set 'OriginProtocolPolicy' to
 'https-only' or match-viewer'. When setting 'OriginProtocolPolicy' to 'match-viewer',
 do not set 'ViewerProtocolPolicy' to 'allow-all' for any cache behaviors.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution[
        filter_cloudfront_distribution_with_legacy_origins(this)
    ] {
        DistributionConfig {
            # Scenario 4
            CustomOrigin not exists
        }
    }

    %cloudfront_distribution [
        # Scenario 2
        filter_cloudfront_distribution_with_origins(this)
    ] {
        let cloudfront_distro = this

        DistributionConfig {
            Origins [
                # Scenario 3
                CustomOriginConfig exists
                CustomOriginConfig is_struct
            ] {
                CustomOriginConfig {
                    # Scenario 5
                    OriginProtocolPolicy != "http-only"
                    # Scenario 6
                    OriginProtocolPolicy == "https-only" or
                    # Scenario 6 and 8

 match_viewer_policy_with_no_allow_all_viewer_protocol_policy(OriginProtocolPolicy,
 %cloudfront_distro)
                }
            }
        }
    }
}

rule
 match_viewer_policy_with_no_allow_all_viewer_protocol_policy(origin_protocol_policy,
 cloudfront_distribution) {
    %origin_protocol_policy {
```

```
        this == "match-viewer"

        %cloudfront_distribution {
            DistributionConfig {
                DefaultCacheBehavior exists
                DefaultCacheBehavior is_struct

                DefaultCacheBehavior {
                    check_viewer_protocol_policy(this)
                }

                when CacheBehaviors exists
                     CacheBehaviors is_list
                     CacheBehaviors not empty {

                        CacheBehaviors[*] {
                            check_viewer_protocol_policy(this)
                        }
                }
            }
        }
    }
}

rule check_viewer_protocol_policy(cache_behaviour) {
    %cache_behaviour {
        ViewerProtocolPolicy exists
        ViewerProtocolPolicy != "allow-all"
    }
}

rule filter_cloudfront_distribution_with_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            Origins exists
            Origins is_list
            Origins not empty
        }
    }
}
```

```
rule filter_cloudfront_distribution_with_legacy_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            CustomOrigin exists
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
```

```
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        CacheBehaviors:
        - ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          PathPattern: '*'
          CachePolicyId:
            Ref: CachePolicy
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
```

```
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        DefaultCacheBehavior:
          ViewerProtocolPolicy: allow-all
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: match-viewer
```

[CT.CLOUDFRONT.PR.9] Require an Amazon CloudFront distribution to have a security policy of TLSv1.2 as a minimum

This control checks whether your Amazon CloudFront distributions are using a minimum security policy and cipher suite of TLSv1.2 or greater for viewer connections.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`

- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.9 example templates

Explanation

AWS Control Tower recommends that you use SSL/TLS to communicate with AWS resources. We recommend TLS version 1.2 or later.

You can specify the security policy CloudFront will use for HTTPS connections with viewers. The security policy determines two settings: 1) the minimum SSL/TLS protocol that CloudFront can use to communicate with viewers, and 2) the ciphers that CloudFront can use to encrypt the content that it returns to viewers.

Usage considerations

- This control requires a viewer certificate configuration compatible only with Amazon CloudFront distributions that use `Aliases`, also known as alternate domain names or CNAMEs.

Remediation for rule failure

Provide a `ViewerCertificate` configuration with `MinimumProtocolVersion` set to TLSv1.2 or higher.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured to use TLS version 1.2 for viewer HTTPS connections. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only"
                        }
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "ViewerCertificate": {
                    "MinimumProtocolVersion": "TLSv1.2_2018",
                    "AcmCertificateArn": {
                        "Ref": "ACMCertificate"
                    },
                    "SslSupportMethod": "vip"
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
```

```
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      ViewerCertificate:
        MinimumProtocolVersion: TLSv1.2_2018
        AcmCertificateArn: !Ref 'ACMCertificate'
        SslSupportMethod: vip
```

CT.CLOUDFRONT.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_security_policy_check
#
# Description:
#   This control checks whether your Amazon CloudFront distributions are using a
 minimum security policy and cipher suite of TLSv1.2 or greater for viewer connections.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.ViewerCertificate' is not present on the CloudFront
 distribution resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.ViewerCertificate' is present on the CloudFront
 distribution resource
#       And: 'CloudFrontDefaultCertificate' in 'ViewerCertificate' is set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.ViewerCertificate' is present on the CloudFront
 distribution resource
#       And: 'CloudFrontDefaultCertificate' is not provided in 'ViewerCertificate' or
 provided and set to bool(false)
#       And: 'MinimumProtocolVersion' is not provided in 'ViewerCertificate' or
 provided as an empty string
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.ViewerCertificate' is present on the CloudFront
 distribution resource
#       And: 'CloudFrontDefaultCertificate' is not provided in 'ViewerCertificate' or
 provided and set to bool(false)
#       And: 'MinimumProtocolVersion' is provided in 'ViewerCertificate' and is to one
 of SSLv3, TLSv1,
#             TLSv1_2016, or TLSv1.1_2016
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'DistributionConfig.ViewerCertificate' is present on the CloudFront
 distribution resource
#       And: 'CloudFrontDefaultCertificate' is not provided in 'ViewerCertificate' or
 provided and set to bool(false)
```

```
#       And: 'MinimumProtocolVersion' is provided in 'ViewerCertificate' and is not set
 to SSLv3, TLSv1,
#            TLSv1_2016, or TLSv1.1_2016
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let INPUT_DOCUMENT = this
let NON_COMPLIANT_TLS_POLICIES_LIST = ["SSLv3", "TLSv1", "TLSv1_2016", "TLSv1.1_2016"]

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_security_policy_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %cloudfront_distributions not empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.9]: Require an Amazon CloudFront distribution to have a
 security policy of TLSv1.2 as a minimum
            [FIX]: Provide a 'ViewerCertificate' configuration with
 'MinimumProtocolVersion' set to TLSv1.2 or higher.
        >>
}

rule cloudfront_security_policy_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.9]: Require an Amazon CloudFront distribution to have a
 security policy of TLSv1.2 as a minimum
            [FIX]: Provide a 'ViewerCertificate' configuration with
 'MinimumProtocolVersion' set to TLSv1.2 or higher.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            # Scenario 2
            ViewerCertificate exists
            ViewerCertificate is_struct

            ViewerCertificate {
                # Scenario 3
                CloudFrontDefaultCertificate not exists or
                CloudFrontDefaultCertificate == false

                # Scenario 4, 5 and 6
                MinimumProtocolVersion exists

                check_is_string_and_not_empty(MinimumProtocolVersion)
                MinimumProtocolVersion not in %NON_COMPLIANT_TLS_POLICIES_LIST
            }
        }
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
```

```
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        ViewerCertificate:
          MinimumProtocolVersion: TLSv1.2_2018
          AcmCertificateArn:
            Ref: ACMCertificate
          SslSupportMethod: sni-only
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
```

```
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        ViewerCertificate:
          MinimumProtocolVersion: TLSv1
          AcmCertificateArn:
            Ref: ACMCertificate
          SslSupportMethod: vip
```

[CT.CLOUDFRONT.PR.10] Require any Amazon CloudFrontdistributions with Amazon S3 backed origins to have origin access control configured

This control checks whether your Amazon CloudFront distributions backed by Amazon S3 are configured to use an origin access control.

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.10 example templates

Explanation

CloudFront OAC prevents users from gaining direct access to an Amazon S3 bucket's content. Direct access an S3 bucket bypasses the CloudFront distribution and any permissions that are applied to the underlying S3 bucket content.

Usage considerations

- This control applies only to Amazon CloudFront distributions that have one or more origins backed by Amazon S3 configured.

Remediation for rule failure

The `Origins` property configures origins backed by Amazon S3. For each origin backed by Amazon S3, configure an origin access control identifier using the `OriginAccessControlId` property.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution with an Amazon S3 bucket origin, configured with an origin access control. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
```

```
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": {
                            "Fn::GetAtt": [
                                "OriginBucket",
                                "RegionalDomainName"
                            ]
                        },
                        "OriginAccessControlId": {
                            "Ref": "OriginAccessControl"
                        },
                        "S3OriginConfig": {}
                    }
                ],
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                }
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
        - Id: sampleOrigin
          DomainName: !GetAtt 'OriginBucket.RegionalDomainName'
          OriginAccessControlId: !Ref 'OriginAccessControl'
          S3OriginConfig: {}
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
```

```
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
```

CT.CLOUDFRONT.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_origin_access_control_enabled_check
#
# Description:
#   This control checks whether your Amazon CloudFront distributions backed by Amazon
 S3 are configured to use an origin access control.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: No S3 backed 'Origins' are provided on the CloudFront distribution
 resource or 'Origins' is not present on
#            the CloudFront distribution resource or is present and an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
```

```
#       And: 'S3Origin' is present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more S3 backed 'Origins' are configured on the CloudFront
 distribution resource
#       And: 'OriginAccessControlId' is not present for the 'Origin' or is an empty
 string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more S3 backed 'Origins' are provided on the CloudFront
 distribution resource
#       And: 'OriginAccessControlId' is present for each S3 backed 'Origin' and is a
 non-empty string or valid local
#            reference
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let CLOUDFRONT_ORIGIN_ACCESS_CONTROL_TYPE = "AWS::CloudFront::OriginAccessControl"
let S3_BUCKET_DNS_NAME_PATTERN = /(.*)\.s3(-external-\d|[-\.][a-z]*-[a-z]*-[0-9])?
\.amazonaws\.com(\.cn)?$/
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_origin_access_control_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %cloudfront_distributions not
 empty {
    check(%cloudfront_distributions.Properties)
        <<
```

```
        [CT.CLOUDFRONT.PR.10]: Require any Amazon CloudFront distributions with Amazon
 S3 backed origins to have origin access control configured
            [FIX]: The 'Origins' property configures origins backed by Amazon S3. For
 each origin backed by Amazon S3, configure an origin access control identifier using
 the 'OriginAccessControlId' property.
        >>
}

rule cloudfront_origin_access_control_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.10]: Require any Amazon CloudFront distributions with Amazon
 S3 backed origins to have origin access control configured
            [FIX]: The 'Origins' property configures origins backed by Amazon S3. For
 each origin backed by Amazon S3, configure an origin access control identifier using
 the 'OriginAccessControlId' property.
        >>
}

#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution[
        filter_cloudfront_distribution_with_legacy_s3_origins(this)
    ] {
        DistributionConfig {
            # Scenario 3
            S3Origin not exists
        }
    }

    %cloudfront_distribution[
        # Scenario 2
        filter_cloudfront_distribution_with_origins(this)
    ] {
        DistributionConfig {
            Origins [
                # Scenario 4
                DomainName == %S3_BUCKET_DNS_NAME_PATTERN or
                check_origin_domain_name_get_att(DomainName)
            ] {
                # Scenario 3 and 5
```

```
                OriginAccessControlId exists
                check_is_string_and_not_empty(OriginAccessControlId) or
                check_local_references(%INPUT_DOCUMENT, OriginAccessControlId,
 %CLOUDFRONT_ORIGIN_ACCESS_CONTROL_TYPE)
            }
        }
    }
}

rule filter_cloudfront_distribution_with_legacy_s3_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            S3Origin exists
        }
    }
}

rule filter_cloudfront_distribution_with_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            Origins exists
            Origins is_list
            Origins not empty
        }
    }
}

rule check_origin_domain_name_get_att(domain) {
  %domain {
    'Fn::GetAtt' {
        this is_list
        this not empty
        this[1] == "DomainName" or
        this[1] == "RegionalDomainName"
    }
    check_local_references(%INPUT_DOCUMENT, this, "AWS::S3::Bucket")
  }
}
```

```
#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CLOUDFRONT.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  OriginAccessControl:
    Type: AWS::CloudFront::OriginAccessControl
    Properties:
      OriginAccessControlConfig:
        Name:
          Fn::Sub: ${AWS::StackName}-example-oac
        OriginAccessControlOriginType: s3
        SigningBehavior: always
        SigningProtocol: sigv4
  OriginBucket:
    Type: AWS::S3::Bucket
  OriginBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: OriginBucket
      PolicyDocument:
        Version: 2012-10-17
```

```
        Statement:
          - Action:
              - 's3:GetObject'
            Effect: Allow
            Resource:
              Fn::Join:
              - ''
              - - 'arn:aws:s3:::'
                - Ref: OriginBucket
                - /*
            Principal:
              Service: cloudfront.amazonaws.com
            Condition:
              StringEquals:
                "AWS:SourceArn":
                  Fn::Join:
                  - ''
                  - - 'arn:aws:cloudfront::'
                    - Ref: AWS::AccountId
                    - ':distribution/'
                    - Ref: CloudFrontDistribution
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        Origins:
        - Id: exampleOrigin
          DomainName:
            Fn::GetAtt:
            - OriginBucket
            - RegionalDomainName
          OriginAccessControlId:
            Ref: OriginAccessControl
          S3OriginConfig: {}
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  OriginBucketOai:
    Type: AWS::CloudFront::CloudFrontOriginAccessIdentity
    Properties:
      CloudFrontOriginAccessIdentityConfig:
        Comment:
          Fn::Sub: ${AWS::StackName}-example-oai
  OriginBucket:
    Type: AWS::S3::Bucket
  OriginBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: OriginBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetObject'
            Effect: Allow
            Resource:
              Fn::Join:
```

```
            - ''
            - - 'arn:aws:s3:::'
              - Ref: OriginBucket
              - /*
          Principal:
            AWS:
              Fn::Join:
              - ''
              - - 'arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity '
                - Ref: OriginBucketOai
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      Origins:
      - Id: exampleS3Origin
        DomainName:
          Fn::GetAtt:
          - OriginBucket
          - RegionalDomainName
        S3OriginConfig:
          OriginAccessIdentity:
            Fn::Sub: "origin-access-identity/cloudfront/${OriginBucketOai}"
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: exampleS3Origin
        CachePolicyId:
          Ref: CachePolicy
```

[CT.CLOUDFRONT.PR.11] Require an Amazon CloudFront distribution to use updated SSL protocols between edge locations and custom origins

This control checks whether your Amazon CloudFront distributions are using deprecated SSL protocols for HTTPS communication between CloudFront edge locations and custom origins.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::CloudFront::Distribution`
- **CloudFormation guard rule:** CT.CLOUDFRONT.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDFRONT.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDFRONT.PR.11 example templates

Explanation

In 2015, the Internet Engineering Task Force (IETF) officially announced that SSL 3.0 should be deprecated, because the protocol is insufficiently secure. We recommend that you use TLSv1.2 or later for HTTPS communication to your custom origins.

Usage considerations

- This control applies only to Amazon CloudFront distributions that have one or more custom origins configured.

Remediation for rule failure

Remove deprecated SSL protocols from `OriginSSLProtocols` in `Origins` that have `CustomOriginConfig` configurations.

The examples that follow show how to implement this remediation.

Amazon CloudFront Distribution - Example

Amazon CloudFront distribution configured to use TLS v1.2 as an origin SSL protocol. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "CloudFrontDistribution": {
        "Type": "AWS::CloudFront::Distribution",
        "Properties": {
            "DistributionConfig": {
                "Enabled": false,
                "DefaultCacheBehavior": {
                    "ViewerProtocolPolicy": "https-only",
                    "TargetOriginId": "sampleOrigin",
                    "CachePolicyId": {
                        "Ref": "CachePolicy"
                    }
                },
                "Origins": [
                    {
                        "Id": "sampleOrigin",
                        "DomainName": "example.com",
                        "CustomOriginConfig": {
                            "OriginProtocolPolicy": "https-only",
                            "OriginSSLProtocols": [
                                "TLSv1.2"
                            ]
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Enabled: false
      DefaultCacheBehavior:
        ViewerProtocolPolicy: https-only
        TargetOriginId: sampleOrigin
        CachePolicyId: !Ref 'CachePolicy'
      Origins:
```

```
        - Id: sampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
            OriginSSLProtocols:
              - TLSv1.2
```

CT.CLOUDFRONT.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudfront_no_deprecated_ssl_protocols_check
#
# Description:
#   This control checks whether your Amazon CloudFront distributions are using
 deprecated SSL protocols for HTTPS communication between CloudFront edge locations and
 custom origins.
#
# Reports on:
#   AWS::CloudFront::Distribution
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudFront distribution resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'Origins' is not present or is an empty list
#      Then: SKIP
```

```
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There are no 'Origins' with a 'CustomOriginConfig'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: All 'Origins' with a 'CustomOriginConfig' have an 'OriginProtocolPolicy'
 of 'http-only'
#      Then: SKIP
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: 'CustomOrigin' is present on the CloudFront distribution resource
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: One or more 'Origins' with a 'CustomOriginConfig' have an
 'OriginProtocolPolicy' not equal to 'http-only'
#       And: 'OriginSSLProtocols' has not been specified or specified as an empty list
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
#       And: One or more 'Origins' with a 'CustomOriginConfig' have an
 'OriginProtocolPolicy' not equal to 'http-only'
#       And: 'OriginSSLProtocols' has been specified as a non-empty list and contains
 'SSLv3'
#      Then: FAIL
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudFront distribution resource
#       And: One or more 'Origins' has been configured
#       And: There one or more 'Origins' with a 'CustomOriginConfig'
```

```
#       And: One or more 'Origins' with a 'CustomOriginConfig' have an
 'OriginProtocolPolicy' not equal to 'http-only'
#       And: 'OriginSSLProtocols' has been specified as a non-empty list and does not
 contain 'SSLv3'
#      Then: PASS

#
# Constants
#
let CLOUDFRONT_DISTRIBUTION_TYPE = "AWS::CloudFront::Distribution"
let UNSUPPORTED_ORIGIN_SSL_PROTOCOLS = [ "SSLv3" ]
let OUT_OF_SCOPE_PROTOCOL_POLICIES = [ "http-only" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudfront_distributions = Resources.*[ Type == %CLOUDFRONT_DISTRIBUTION_TYPE ]

#
# Primary Rules
#
rule cloudfront_no_deprecated_ssl_protocols_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %cloudfront_distributions not
 empty {
    check(%cloudfront_distributions.Properties)
        <<
        [CT.CLOUDFRONT.PR.11]: Require an Amazon CloudFront distribution to use updated
 SSL protocols between edge locations and custom origins
            [FIX]: Remove deprecated SSL protocols from 'OriginSSLProtocols' in
 'Origins' that have 'CustomOriginConfig' configurations.
        >>
}

rule cloudfront_no_deprecated_ssl_protocols_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDFRONT_DISTRIBUTION_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDFRONT_DISTRIBUTION_TYPE.resourceProperties)
        <<
        [CT.CLOUDFRONT.PR.11]: Require an Amazon CloudFront distribution to use updated
 SSL protocols between edge locations and custom origins
            [FIX]: Remove deprecated SSL protocols from 'OriginSSLProtocols' in
 'Origins' that have 'CustomOriginConfig' configurations.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(cloudfront_distribution) {
    %cloudfront_distribution[
        filter_cloudfront_distribution_with_legacy_origins(this)
    ] {
        DistributionConfig {
            # Scenario 5
            CustomOrigin not exists
        }
    }

    %cloudfront_distribution[
        # Scenario 2
        filter_cloudfront_distribution_with_origins(this)
    ] {
        DistributionConfig {
            Origins [
                # Scenario 3 and 4
                CustomOriginConfig exists
                CustomOriginConfig is_struct
                filter_custom_origin_config(CustomOriginConfig)
            ] {
                CustomOriginConfig {
                    # Scenario 6, 7 and 8
                    OriginSSLProtocols exists
                    OriginSSLProtocols is_list
                    OriginSSLProtocols not empty
                    %UNSUPPORTED_ORIGIN_SSL_PROTOCOLS.* not in OriginSSLProtocols
                }
            }
        }
    }
}


rule filter_cloudfront_distribution_with_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
```

```
            Origins exists
            Origins is_list
            Origins not empty
        }
    }
}

rule filter_cloudfront_distribution_with_legacy_origins(cloudfront_distribution) {
    %cloudfront_distribution {
        DistributionConfig exists
        DistributionConfig is_struct

        DistributionConfig {
            CustomOrigin exists
        }
    }
}

rule filter_custom_origin_config(custom_origin_config) {
    %custom_origin_config {
        OriginProtocolPolicy exists
        OriginProtocolPolicy not in %OUT_OF_SCOPE_PROTOCOL_POLICIES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDFRONT.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
            OriginSSLProtocols:
            - TLSv1
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CachePolicy:
    Type: AWS::CloudFront::CachePolicy
    Properties:
      CachePolicyConfig:
        DefaultTTL: 20
        MaxTTL: 20
        MinTTL: 19
        Name:
          Fn::Sub: ${AWS::StackName}-example-cache-policy
ParametersInCacheKeyAndForwardedToOrigin:
          CookiesConfig:
            CookieBehavior: none
          EnableAcceptEncodingGzip: false
          HeadersConfig:
            HeaderBehavior: none
          QueryStringsConfig:
            QueryStringBehavior: none
  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Enabled: false
        DefaultCacheBehavior:
          ViewerProtocolPolicy: https-only
          TargetOriginId: exampleOrigin
          CachePolicyId:
            Ref: CachePolicy
        Origins:
        - Id: exampleOrigin
          DomainName: example.com
          CustomOriginConfig:
            OriginProtocolPolicy: https-only
            OriginSSLProtocols:
            - SSLv3
```

AWS CloudTrail controls

Topics

- [CT.CLOUDTRAIL.PR.1] Require an AWS CloudTrail trail to have encryption at rest activated
- [CT.CLOUDTRAIL.PR.2] Require an AWS CloudTrail trail to have log file validation activated
- [CT.CLOUDTRAIL.PR.3] Require an AWS CloudTrail trail to have an Amazon CloudWatch Logs log group configuration
- [CT.CLOUDTRAIL.PR.4] Require an AWS CloudTrail Lake event data store to enable encryption at rest with an AWS KMS key

[CT.CLOUDTRAIL.PR.1] Require an AWS CloudTrail trail to have encryption at rest activated

This control checks whether your AWS CloudTrail is configured to use the server-side encryption (SSE) AWS KMS key encryption.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudTrail::Trail`
- **CloudFormation guard rule:** CT.CLOUDTRAIL.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDTRAIL.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDTRAIL.PR.1 example templates

Explanation

For an added layer of security for your sensitive CloudTrail log files, you should use server-side encryption with AWS KMS keys (SSE-KMS) for your CloudTrail log files for encryption at rest. Note that by default, the log files delivered by CloudTrail to your buckets are encrypted by Amazon server-side encryption with Amazon S3-managed encryption keys (SSE-S3).

Remediation for rule failure

Set the `KMSKeyId` property to a valid KMS key.

The examples that follow show how to implement this remediation.

AWS CloudTrail trail - Example

AWS CloudTrail Trail configured to use server-side encryption with AWS KMS keys (SSE-KMS). The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudTrail": {
        "Type": "AWS::CloudTrail::Trail",
        "Properties": {
            "IsLogging": true,
            "KMSKeyId": {
                "Ref": "KMSKey"
            },
            "S3BucketName": {
                "Ref": "LoggingBucket"
            }
        }
    }
}
```

YAML example

```
CloudTrail:
  Type: AWS::CloudTrail::Trail
  Properties:
    IsLogging: true
    KMSKeyId: !Ref 'KMSKey'
    S3BucketName: !Ref 'LoggingBucket'
```

CT.CLOUDTRAIL.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloud_trail_encryption_enabled_check
#
# Description:
#   This rule checks whether AWS CloudTrail is configured to use the server-side
 encryption (SSE) AWS KMS key encryption.
#
# Reports on:
#   AWS::CloudTrail::Trail
#
# Evaluates:
#   AWS CloudFormation, AWS CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document does not contain any AWS CloudTrailtrails
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'KMSKeyId' is not present
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'KMSKeyId' has been provided and is set to an empty string or a non-valid
 local reference to a KMS key or
#            Alias
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'KMSKeyId' has been provided and is a non-empty string or a valid
 reference to a KMS key or Alias.
#      Then: PASS

#
# Constants
#
let CLOUDTRAIL_TRAIL_TYPE = "AWS::CloudTrail::Trail"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudtrail_trails = Resources.*[ Type == %CLOUDTRAIL_TRAIL_TYPE ]

#
# Primary Rules
#
rule cloud_trail_encryption_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %cloudtrail_trails not empty {
    check_cloudtrail_kms_key_configuration(%cloudtrail_trails.Properties)
        < <
        [CT.CLOUDTRAIL.PR.1]: Require an AWS CloudTrail trail to have encryption at
 rest activated
        [FIX]: Set the 'KMSKeyId' property to a valid KMS key.
        >>
}

rule cloud_trail_encryption_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDTRAIL_TRAIL_TYPE) {
    check_cloudtrail_kms_key_configuration(%INPUT_DOCUMENT.
%CLOUDTRAIL_TRAIL_TYPE.resourceProperties)
        <<
        [CT.CLOUDTRAIL.PR.1]: Require an AWS CloudTrail trail to have encryption at
 rest activated
        [FIX]: Set the 'KMSKeyId' property to a valid KMS key.
        >>
}

#
# Parameterized Rules
```

```
#
rule check_cloudtrail_kms_key_configuration(cloudtrail_trail){
    %cloudtrail_trail {
        # Scenario 2
        KMSKeyId exists
        # Scenario 3 and 4
        check_is_string_and_not_empty(KMSKeyId) or
        check_kms_key_id_local_ref(KMSKeyId)
    }
}

rule check_kms_key_id_local_ref(key_ref) {
    %key_ref {
      check_local_references(%INPUT_DOCUMENT, this, "AWS::KMS::Key") or
      check_local_references(%INPUT_DOCUMENT, this, "AWS::KMS::Alias")
    }
 }

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_RESOURCE_TYPE) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
```

```
        } or Ref {
            query_for_resource(%doc, this, %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_RESOURCE_TYPE) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_RESOURCE_TYPE
    }
}
```

CT.CLOUDTRAIL.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-cloudtrail-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
        - Sid: Allow CloudTrail to encrypt logs
          Effect: Allow
          Action: "kms:GenerateDataKey*"
          Principal:
```

```
            Service: "cloudtrail.amazonaws.com"
          Resource: '*'
          Condition:
            StringLike:
              "kms:EncryptionContext:aws:cloudtrail:arn": [
                Fn::Sub: "arn:aws:cloudtrail:*:${AWS::AccountId}:trail/*"
              ]
            StringEquals:
              "aws:SourceArn":
                Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
        - Sid: Allow CloudTrail to describe key
          Effect: Allow
          Principal:
            Service: "cloudtrail.amazonaws.com"
          Action: kms:DescribeKey
          Resource: '*'
        - Sid: Allow principals in the account to decrypt log files
          Effect: Allow
          Principal:
            AWS: "*"
          Action:
           - "kms:Decrypt"
           - "kms:ReEncryptFrom"
          Resource: "*"
          Condition:
            StringEquals:
              "kms:CallerAccount":
                Ref: AWS::AccountId
              "kms:EncryptionContext:aws:cloudtrail:arn":
                Fn::Sub: "arn:aws:cloudtrail:*:${AWS::AccountId}:trail/*"
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
```

```
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudTrail:
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      KMSKeyId:
        Ref: KMSKey
      S3BucketName:
        Ref: LoggingBucket
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
```

```
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudTrail:
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      S3BucketName:
        Ref: LoggingBucket
```

[CT.CLOUDTRAIL.PR.2] Require an AWS CloudTrail trail to have log file validation activated

This control checks whether log file integrity validation is enabled on an AWS CloudTrail trail.

- **Control objective:** Manage secrets
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudTrail::Trail`
- **CloudFormation guard rule:** CT.CLOUDTRAIL.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDTRAIL.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDTRAIL.PR.2 example templates

Explanation

CloudTrail log file validation creates a digitally-signed digest file that contains a hash of each log that CloudTrail writes to Amazon S3. You can use these digest files to determine whether a log file was changed, deleted, or unchanged after CloudTrail delivered the log.

AWS Control Tower recommends that you enable file validation on all trails. Log file validation provides additional integrity checks of CloudTrail logs.

Remediation for rule failure

Set the CloudTrail resource `EnableLogFileValidation` property to true.

The examples that follow show how to implement this remediation.

AWS CloudTrail trail - Example

AWS CloudTrail trail configured with log file validation. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudTrail": {
        "Type": "AWS::CloudTrail::Trail",
        "Properties": {
            "IsLogging": true,
            "S3BucketName": {
                "Ref": "LoggingBucket"
            },
            "KMSKeyId": {
                "Ref": "KMSKey"
            },
            "EnableLogFileValidation": true
        }
    }
}
```

YAML example

```
CloudTrail:
```

```
  Type: AWS::CloudTrail::Trail
  Properties:
    IsLogging: true
    S3BucketName: !Ref 'LoggingBucket'
    KMSKeyId: !Ref 'KMSKey'
    EnableLogFileValidation: true
```

CT.CLOUDTRAIL.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloud_trail_log_file_validation_enabled_check
#
# Description:
#   This control checks whether log file integrity validation is enabled on an AWS
 CloudTrail trail.
#
# Reports on:
#   AWS::CloudTrail::Trail
#
# Evaluates:
#   AWS CloudFormation, AWS CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document does not contain any CloudTrail trails
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'EnableLogFileValidation' is not present
```

```
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'EnableLogFileValidation' is present and and is set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains a CloudTrail trail resource
#       And: 'EnableLogFileValidation' is present and set to bool(true)
#      Then: PASS

#
# Constants
#
let CLOUDTRAIL_TRAIL_TYPE = "AWS::CloudTrail::Trail"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudtrail_trails = Resources.*[ Type == %CLOUDTRAIL_TRAIL_TYPE ]

#
# Primary Rules
#
rule cloud_trail_log_file_validation_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                      %cloudtrail_trails not empty {
    check(%cloudtrail_trails.Properties)
        <<
        [CT.CLOUDTRAIL.PR.2]: Require an AWS CloudTrail trail to have log file
 validation activated
        [FIX]: Set the CloudTrail resource 'EnableLogFileValidation' property to true.
        >>
}

rule cloud_trail_log_file_validation_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDTRAIL_TRAIL_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDTRAIL_TRAIL_TYPE.resourceProperties)
        <<
```

```
        [CT.CLOUDTRAIL.PR.2]: Require an AWS CloudTrail trail to have log file
 validation activated
        [FIX]: Set the CloudTrail resource 'EnableLogFileValidation' property to true.
        >>
}

#
# Parameterized Rules
#
rule check(cloudtrail_trail){
    %cloudtrail_trail {
        # Scenario 2
        EnableLogFileValidation exists
        # Scenario 3 and 4
        EnableLogFileValidation == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDTRAIL.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LoggingBucket:
```

```
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudTrail:
```

```
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      S3BucketName:
        Ref: LoggingBucket
      EnableLogFileValidation: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
```

```
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudTrail:
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      S3BucketName:
        Ref: LoggingBucket
      EnableLogFileValidation: false
```

[CT.CLOUDTRAIL.PR.3] Require an AWS CloudTrail trail to have an Amazon CloudWatch Logs log group configuration

This control checks whether your AWS CloudTrail trail is configured to send logs to Amazon CloudWatch Logs Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudTrail::Trail`
- **CloudFormation guard rule:** CT.CLOUDTRAIL.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDTRAIL.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDTRAIL.PR.3 example templates

Explanation

CloudTrail records AWS API calls that are made in a given account. The recorded information includes: the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters and the response elements returned by the AWS service.

CloudTrail uses Amazon S3 for log file storage and delivery. You can capture CloudTrail logs in a specified S3 bucket for long-term analysis. To perform real-time analysis, you can configure CloudTrail to send logs to CloudWatch Logs.

For a trail that is enabled in all AWS Regions in an account, CloudTrail sends log files from all of those Regions to a CloudWatch Logs log group.

AWS CloudTrail recommends that you send CloudTrail logs to CloudWatch Logs. Note that this recommendation is intended to ensure that account activity is captured, monitored, and appropriately alarmed on. You can use CloudWatch Logs to set this up with your AWS services. This recommendation does not preclude the use of a different solution.

Sending CloudTrail logs to CloudWatch Logs facilitates real-time and historic activity logging based on user, API, resource, and IP address. You can use this approach to establish alarms and notifications for anomalous or sensitivity account activity.

Remediation for rule failure

Set the `CloudWatchLogsLogGroupArn` and `CloudWatchLogsRoleArn` properties.

The examples that follow show how to implement this remediation.

AWS CloudTrail trail - Example

AWS CloudTrail trail configured to send events to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudTrail": {
        "Type": "AWS::CloudTrail::Trail",
        "Properties": {
            "IsLogging": true,
            "S3BucketName": {
                "Ref": "LoggingBucket"
            },
            "CloudWatchLogsRoleArn": {
                "Fn::GetAtt": [
                    "LogRole",
                    "Arn"
                ]
            },
            "CloudWatchLogsLogGroupArn": {
                "Fn::GetAtt": [
                    "LogGroup",
                    "Arn"
                ]
            }
        }
    }
}
```

YAML example

```
CloudTrail:
  Type: AWS::CloudTrail::Trail
  Properties:
    IsLogging: true
    S3BucketName: !Ref 'LoggingBucket'
    CloudWatchLogsRoleArn: !GetAtt 'LogRole.Arn'
    CloudWatchLogsLogGroupArn: !GetAtt 'LogGroup.Arn'
```

CT.CLOUDTRAIL.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloud_trail_cloud_watch_logs_enabled_check
#
# Description:
#   This rule checks whether AWS CloudTrail trails are configured to send logs to
 Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::CloudTrail::Trail
#
# Evaluates:
#   AWS CloudFormation, AWS CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document does not contain any AWS CloudTrail trails
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an AWS CloudTrail trail resource
#       And: 'CloudWatchLogsLogGroupArn' or 'CloudWatchLogsRoleArn' is not present
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an AWS CloudTrail trail resource
#       And: 'CloudWatchLogsLogGroupArn' is set to a non-empty string or a valid local
 reference to a log group
#       And: 'CloudWatchLogsRoleArn' is set to an empty string or a non-valid local
 reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
```

```
#       And: The input document contains an AWS CloudTrail trail resource
#       And: 'CloudWatchLogsLogGroupArn' is set to an empty string or an invalid local
 reference
#       And: 'CloudWatchLogsRoleArn' is set to a non-empty string or a valid local
 reference to an IAM role
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an AWS CloudTrail trail resource
#       And: 'CloudWatchLogsRoleArn' is set to a non-empty string or a valid local
 reference to an IAM role
#       And: 'CloudWatchLogsLogGroupArn' is set to a non-empty string or a valid local
 reference to a log group
#      Then: PASS

#
# Constants
#
let CLOUDTRAIL_TRAIL_TYPE = "AWS::CloudTrail::Trail"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudtrail_trails = Resources.*[ Type == %CLOUDTRAIL_TRAIL_TYPE ]

#
# Primary Rules
#
rule cloud_trail_cloud_watch_logs_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %cloudtrail_trails not empty {
    check_cloudtrail_log_group_configuration(%cloudtrail_trails.Properties)
        <<
        [CT.CLOUDTRAIL.PR.3]: Require an AWS CloudTrail trail to have an CloudTrail log
 group configuration
        [FIX]: Set the 'CloudWatchLogsLogGroupArn' and 'CloudWatchLogsRoleArn'
 properties.
        >>
}

rule cloud_trail_cloud_watch_logs_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDTRAIL_TRAIL_TYPE) {
```

```
    check_cloudtrail_log_group_configuration(%INPUT_DOCUMENT.
%CLOUDTRAIL_TRAIL_TYPE.resourceProperties)
        <<
        [CT.CLOUDTRAIL.PR.3]: Require an AWS CloudTrail trail to have an CloudTrail log
 group configuration
        [FIX]: Set the 'CloudWatchLogsLogGroupArn' and 'CloudWatchLogsRoleArn'
 properties.
        >>
}

#
# Parameterized Rules
#
rule check_cloudtrail_log_group_configuration(cloudtrail_trail) {
    %cloudtrail_trail {
        # Scenario 2
        CloudWatchLogsLogGroupArn exists
        CloudWatchLogsRoleArn exists

        # Scenario 3, 4 and 5
        check_cloudwatch_log_group_arn(CloudWatchLogsLogGroupArn)
        check_cloudwatch_log_role_arn(CloudWatchLogsRoleArn)
    }
}

rule check_cloudwatch_log_group_arn(log_group) {
   %log_group {
      check_is_string_and_not_empty(this) or
      check_local_references(%INPUT_DOCUMENT, this, "AWS::Logs::LogGroup")
    }
}

rule check_cloudwatch_log_role_arn(log_role) {
   %log_role {
      check_is_string_and_not_empty(this) or
      check_local_references(%INPUT_DOCUMENT, this, "AWS::IAM::Role")
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
```

```
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_RESOURCE_TYPE) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_RESOURCE_TYPE) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_RESOURCE_TYPE
    }
}
```

CT.CLOUDTRAIL.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
```

```
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudWatchLogsRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Sid: AssumeRole
          Effect: Allow
          Principal:
            Service: 'cloudtrail.amazonaws.com'
          Action: 'sts:AssumeRole'
      Policies:
      - PolicyName: 'cloudtrail-policy'
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - 'logs:CreateLogStream'
            - 'logs:PutLogEvents'
            Resource:
              Fn::GetAtt: [LogGroup, Arn]
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties: {}
  CloudTrail:
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      S3BucketName:
        Ref: LoggingBucket
      CloudWatchLogsRoleArn:
        Fn::GetAtt:
        - CloudWatchLogsRole
        - Arn
      CloudWatchLogsLogGroupArn:
        Fn::GetAtt:
        - LogGroup
        - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:GetBucketAcl'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - 'arn:aws:s3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
```

```
                - /*
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Condition:
              StringEquals:
                's3:x-amz-acl': 'bucket-owner-full-control'
                "aws:SourceArn":
                  Fn::Sub: "arn:aws:cloudtrail:${AWS::Region}:${AWS::AccountId}:trail/
${AWS::StackName}-example-trail"
  CloudTrail:
    Type: AWS::CloudTrail::Trail
    Properties:
      IsLogging: true
      TrailName:
        Fn::Sub: ${AWS::StackName}-example-trail
      S3BucketName:
        Ref: LoggingBucket
```

[CT.CLOUDTRAIL.PR.4] Require an AWS CloudTrail Lake event data store to enable encryption at rest with an AWS KMS key

This control checks whether a CloudTrail Lake event data store is encrypted at rest with a KMS key.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudTrail::EventDataStore`
- **CloudFormation guard rule:** CT.CLOUDTRAIL.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDTRAIL.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CLOUDTRAIL.PR.4 example templates

Explanation

Encrypting data at rest reduces the risk that a user not authenticated to AWS may obtain access to data stored on disk. For added control over encryption keys, you can use customer-managed keys from AWS KMS. You have full control over these KMS keys. You can establish and maintain their key policies, IAM policies, and grants, enable and disable the keys, rotate their cryptographic material, add tags, create aliases that refer to the KMS keys, and schedule the KMS keys for deletion.

Usage considerations

- All events in an AWS CloudTrail Lake event data store are encrypted by CloudTrail using a KMS key that AWS owns and manages for you. For added control over encryption keys, you can use customer-managed keys from AWS KMS. For more information, see AWS KMS Concepts in the *AWS KMS Developer Guide*.

Remediation for rule failure

Set the `KmsKeyId` parameter to the ARN of an AWS KMS customer-managed key, configured with permissions that allow the CloudTrail service principal to use the key.

The examples that follow show how to implement this remediation.

CloudTrail Lake event data store - Example

CloudTrail Lake event data store configured to encrypt data at rest with an AWS KMS key. The example is shown in JSON and in YAML.

JSON example

```
{
    "CloudTrailEventDataStore": {
        "Type": "AWS::CloudTrail::EventDataStore",
        "Properties": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-example"
            },
            "TerminationProtectionEnabled": false,
            "KmsKeyId": {
                "Fn::GetAtt": [
                    "KMSKey",
                    "Arn"
```

```
                ]
            }
        }
    }
}
```

YAML example

```
CloudTrailEventDataStore:
  Type: AWS::CloudTrail::EventDataStore
  Properties:
    Name: !Sub '${AWS::StackName}-example'
    TerminationProtectionEnabled: false
    KmsKeyId: !GetAtt 'KMSKey.Arn'
```

CT.CLOUDTRAIL.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   cloud_trail_event_datastore_encrypted_at_rest_kms_check
#
# Description:
#   This control checks whether a CloudTrail Lake event data store is encrypted at rest
 with a KMS key.
#
# Reports on:
#   AWS::CloudTrail::EventDataStore
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudTrail event data store
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudTrail event data store resource
#       And: 'KmsKeyId' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudTrail event data store resource
#       And: 'KmsKeyId' has been provided as an empty string or invalid local reference
#            to a KMS keyID or alias
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains a CloudTrail event data store resource
#       And: 'KmsKeyId' has been provided as a non-empty string or valid local
 reference
#            to a KMS keyID or alias
#      Then: PASS

#
# Constants
#
let CLOUDTRAIL_EVENT_DATASTORE_TYPE = "AWS::CloudTrail::EventDataStore"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudtrail_event_datastores = Resources.*[ Type ==
 %CLOUDTRAIL_EVENT_DATASTORE_TYPE ]

#
# Primary Rules
#
rule cloud_trail_event_datastore_encrypted_at_rest_kms_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %cloudtrail_event_datastores not empty {
    check(%cloudtrail_event_datastores.Properties)
```

```
        <<
        [CT.CLOUDTRAIL.PR.4]: Require an CloudTrail Lake event data store to enable
 encryption at rest with an AWS KMS key
        [FIX]: Set the 'KmsKeyId' parameter to the ARN of an AWS KMS customer-managed
 key, configured with permissions that allow the CloudTrail service principal to use
 the key.
        >>
}

rule cloud_trail_event_datastore_encrypted_at_rest_kms_check when
 is_cfn_hook(%INPUT_DOCUMENT, %CLOUDTRAIL_EVENT_DATASTORE_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDTRAIL_EVENT_DATASTORE_TYPE.resourceProperties)
        <<
        [CT.CLOUDTRAIL.PR.4]: Require an CloudTrail Lake event data store to enable
 encryption at rest with an AWS KMS key
        [FIX]: Set the 'KmsKeyId' parameter to the ARN of an AWS KMS customer-managed
 key, configured with permissions that allow the CloudTrail service principal to use
 the key.
        >>
}

#
# Parameterized Rules
#
rule check(cloudtrail_event_datastore) {
    %cloudtrail_event_datastore {
        # Scenario 2
        KmsKeyId exists

        # Scenario 3 and 4
        check_is_string_and_not_empty(KmsKeyId) or
        check_local_references(%INPUT_DOCUMENT, KmsKeyId, "AWS::KMS::Key") or
        check_local_references(%INPUT_DOCUMENT, KmsKeyId, "AWS::KMS::Alias")
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
```

```
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_RESOURCE_TYPE) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_RESOURCE_TYPE) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_RESOURCE_TYPE
    }
}
```

CT.CLOUDTRAIL.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
        - Sid: Allow CloudTrail to encrypt event data store
          Effect: Allow
          Principal:
            Service: "cloudtrail.amazonaws.com"
          Action:
          - "kms:GenerateDataKey"
          - "kms:Decrypt"
          Resource: "*"
      KeySpec: SYMMETRIC_DEFAULT
      EnableKeyRotation: true
  CloudTrailEventDataStore:
    Type: AWS::CloudTrail::EventDataStore
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      TerminationProtectionEnabled: false
      MultiRegionEnabled: false
      KmsKeyId:
        Fn::GetAtt:
        - KMSKey
        - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CloudTrailEventDataStore:
    Type: AWS::CloudTrail::EventDataStore
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      TerminationProtectionEnabled: false
      MultiRegionEnabled: false
```

Amazon CloudWatch controls

Topics

- [CT.CLOUDWATCH.PR.1] Require an Amazon CloudWatch alarm to have an action configured for the alarm state
- [CT.CLOUDWATCH.PR.2] Require an Amazon CloudWatch log group to be retained for at least one year
- [CT.CLOUDWATCH.PR.3] Require an Amazon CloudWatch log group to be encrypted at rest with an AWS KMS key
- [CT.CLOUDWATCH.PR.4] Require an Amazon CloudWatch alarm to have actions activated

[CT.CLOUDWATCH.PR.1] Require an Amazon CloudWatch alarm to have an action configured for the alarm state

This control checks whether an Amazon CloudWatch alarm has at least one action configured for the alarm state.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudWatch::Alarm`
- **CloudFormation guard rule:** CT.CLOUDWATCH.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDWATCH.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.CLOUDWATCH.PR.1 example templates

Explanation

AWS Control Tower recommends configuring actions for alarms to alert you automatically when an alarm is in the alarm state and the monitored metric is outside the defined threshold. This configuration ensures that alarms are monitored, and that necessary actions are taken when the alarm is triggered. Monitoring alarms help you identify unusual activities and respond quickly to security and operational issues. You can specify the actions an alarm should take when it goes into OK, ALARM, and INSUFFICIENT_DATA states. The most common CloudWatch alarm action in the alarm state is to notify one or more users by sending a message to an Amazon Simple Notification Service (Amazon SNS) topic.

Remediation for rule failure

Set `AlarmActions` to a list with one or more alarm action values.

The examples that follow show how to implement this remediation.

Amazon CloudWatch Alarm - Example

An Amazon CloudWatch alarm configured to notify an SNS topic when the CloudWatch alarm is in the alarm state. The example is shown in JSON and in YAML.

JSON example

```
{
    "Alarm": {
        "Type": "AWS::CloudWatch::Alarm",
        "Properties": {
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "EvaluationPeriods": 1,
            "Period": 300,
            "Threshold": 1.0,
            "Namespace": "AWS/Lambda",
            "MetricName": "Errors",
            "TreatMissingData": "missing",
```

```
            "Statistic": "Sum",
            "DatapointsToAlarm": 1,
            "ActionsEnabled": true,
            "AlarmActions": [
                {
                    "Ref": "Topic"
                }
            ]
        }
    }
}
```

YAML example

```
Alarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    ComparisonOperator: GreaterThanOrEqualToThreshold
    EvaluationPeriods: 1
    Period: 300
    Threshold: 1.0
    Namespace: AWS/Lambda
    MetricName: Errors
    TreatMissingData: missing
    Statistic: Sum
    DatapointsToAlarm: 1
    ActionsEnabled: true
    AlarmActions:
      - !Ref 'Topic'
```

CT.CLOUDWATCH.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudwatch_alarm_action_check
```

```
#
# Description:
#   This control checks whether an Amazon CloudWatch alarm has at least one action
 configured for the alarm state.
#
# Reports on:
#   AWS::CloudWatch::Alarm
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudWatch alarm resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'AlarmActions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'AlarmActions' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'AlarmActions' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let CLOUDWATCH_ALARM_TYPE = "AWS::CloudWatch::Alarm"
let INPUT_DOCUMENT = this

#
# Assignments
#
```

```
let cloudwatch_alarms = Resources.*[ Type == %CLOUDWATCH_ALARM_TYPE ]

#
# Primary Rules
#
rule cloudwatch_alarm_action_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %cloudwatch_alarms not empty {
    check(%cloudwatch_alarms.Properties)
        <<
        [CT.CLOUDWATCH.PR.1]: Require an Amazon CloudWatch alarm to have an action
 configured for the alarm state
        [FIX]: Set 'AlarmActions' to a list with one or more alarm action values.
        >>
}

rule cloudwatch_alarm_action_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDWATCH_ALARM_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDWATCH_ALARM_TYPE.resourceProperties)
        <<
        [CT.CLOUDWATCH.PR.1]: Require an Amazon CloudWatch alarm to have an action
 configured for the alarm state
        [FIX]: Set 'AlarmActions' to a list with one or more alarm action values.
        >>
}

#
# Parameterized Rules
#
rule check(cloudwatch_alarm){
    %cloudwatch_alarm {
        # Scenario 2
        AlarmActions exists
        # Scenarios 3 and 4
        AlarmActions is_list
        AlarmActions not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
```

```
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDWATCH.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties: {}
  Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      ComparisonOperator: GreaterThanOrEqualToThreshold
      EvaluationPeriods: 1
      Period: 300
      Threshold: 1.0
      Namespace: AWS/Lambda
      MetricName: Errors
      TreatMissingData: missing
      Statistic: Sum
      DatapointsToAlarm: 1
      ActionsEnabled: true
      AlarmActions:
      - Ref: Topic
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      ComparisonOperator: GreaterThanOrEqualToThreshold
      EvaluationPeriods: 1
      Period: 300
      Threshold: 1.0
      Namespace: AWS/Lambda
      MetricName: Errors
      TreatMissingData: missing
      Statistic: Sum
      DatapointsToAlarm: 1
      ActionsEnabled: true
```

[CT.CLOUDWATCH.PR.2] Require an Amazon CloudWatch log group to be retained for at least one year

This control checks whether an Amazon CloudWatch Log Group retention period is set to a value greater than or equal to 365 days.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Logs::LogGroup`
- **CloudFormation guard rule:** CT.CLOUDWATCH.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDWATCH.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.CLOUDWATCH.PR.2 example templates

Explanation

Amazon CloudWatch Logs centralizes the logs from all of your systems, applications, and AWS services in a single, highly scalable service. You can use Amazon CloudWatch Logs to monitor, store, and retrieve your log files from Amazon EC2 instances, CloudTrail, Route 53, and other sources. Retaining your logs for at least one year can help you comply with log retention standards.

Remediation for rule failure

Omit the field value of `RetentionInDays` to adopt the default retention setting of `Never expire`, or set `RetentionInDays` to an integer value greater than or equal to 365.

The examples that follow show how to implement this remediation.

Amazon CloudWatch Log Group - Example

An Amazon CloudWatch log group configured to retain logs for one year (365 days). The example is shown in JSON and in YAML.

JSON example

```
{
    "LogGroup": {
        "Type": "AWS::Logs::LogGroup",
        "Properties": {
            "RetentionInDays": 365
        }
    }
}
```

YAML example

```
LogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    RetentionInDays: 365
```

CT.CLOUDWATCH.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudwatch_log_group_retention_period_check
#
# Description:
#   This control checks whether an Amazon CloudWatch Log Group retention period is set
 to a value greater than or equal to 365 days.
#
# Reports on:
#   AWS::Logs::LogGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudWatch log group resources
#     Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch log group resource
#       And: 'RetentionInDays' has been provided and set to a non integer value or
#            integer value less than 365
#     Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch log group resource
#       And: 'RetentionInDays' has not been provided
#     Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains a CloudWatch log group resource
#       And: 'RetentionInDays' has been provided and set to an integer value greater
 than or equal to 365
#     Then: PASS

#
```

```
# Constants
#
let CLOUDWATCH_LOGS_TYPE = "AWS::Logs::LogGroup"
let MINIMUM_RETENTION_IN_DAYS = 365
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudwatch_log_groups = Resources.*[ Type == %CLOUDWATCH_LOGS_TYPE ]

#
# Primary Rules
#
rule cloudwatch_log_group_retention_period_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %cloudwatch_log_groups not empty
 {
    check(%cloudwatch_log_groups.Properties)
        <<
        [CT.CLOUDWATCH.PR.2]: Require an Amazon CloudWatch log group to be retained for
 at least one year
        [FIX]: Omit the field value of 'RetentionInDays' to adopt the default retention
 setting of 'Never expire', or set 'RetentionInDays' to an integer value greater than
 or equal to 365.
        >>
}

rule cloudwatch_log_group_retention_period_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDWATCH_LOGS_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDWATCH_LOGS_TYPE.resourceProperties)
        <<
        [CT.CLOUDWATCH.PR.2]: Require an Amazon CloudWatch log group to be retained for
 at least one year
        [FIX]: Omit the field value of 'RetentionInDays' to adopt the default retention
 setting of 'Never expire', or set 'RetentionInDays' to an integer value greater than
 or equal to 365.
        >>
}

#
# Parameterized Rules
#
rule check(cloudwatch_log_group){
    %cloudwatch_log_group {
```

```
        # Scenario 3
        RetentionInDays not exists or

        # Scenarios 2 and 4
        RetentionInDays >= %MINIMUM_RETENTION_IN_DAYS
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDWATCH.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      RetentionInDays: 365
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      RetentionInDays: 1
```

[CT.CLOUDWATCH.PR.3] Require an Amazon CloudWatch log group to be encrypted at rest with an AWS KMS key

This control checks whether an Amazon CloudWatch Logs log group is encrypted at rest with an AWS KMS key

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Logs::LogGroup`
- **CloudFormation guard rule:** CT.CLOUDWATCH.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDWATCH.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.CLOUDWATCH.PR.3 example templates

Explanation

Amazon CloudWatch Logs log groups are encrypted by default using server-side encryption. For added control over encryption keys, you can use customer-managed keys from AWS KMS. You have full control over these KMS keys, including establishing and maintaining their key policies, IAM policies, and grants, enabling and disabling the keys, rotating their cryptographic material, adding tags, creating aliases that refer to the KMS keys, and scheduling the KMS keys for deletion.

Remediation for rule failure

Set `KmsKeyId` to the ARN of an AWS KMS customer-managed key configured with permissions that allow the CloudWatch service principal to use the key.

The examples that follow show how to implement this remediation.

Amazon CloudWatch Logs Group - Example

An Amazon CloudWatch log group configured to encrypt logs with an AWS KMS customer-managed key. The example is shown in JSON and in YAML.

JSON example

```
{
    "LogGroup": {
        "Type": "AWS::Logs::LogGroup",
        "Properties": {
            "KmsKeyId": {
                "Fn::GetAtt": [
                    "KMSKey",
                    "Arn"
                ]
            }
        }
    }
}
```

YAML example

```
LogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    KmsKeyId: !GetAtt 'KMSKey.Arn'
```

CT.CLOUDWATCH.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudwatch_log_group_encrypted_check
#
# Description:
#   This control checks whether an Amazon CloudWatch log group is encrypted at rest
 with an AWS KMS key
#
# Reports on:
#   AWS::Logs::LogGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudWatch log group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch log group resource
#       And: 'KmsKeyId' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch log group resource
#       And: 'KmsKeyId' has been provided as an empty string or invalid local reference
 to a KMS Key
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch log group resource
#       And: 'KmsKeyId' has been provided as a non-empty string or valid local
 reference to a KMS Key
#      Then: PASS

#
```

```
# Constants
#
let CLOUDWATCH_LOGS_TYPE = "AWS::Logs::LogGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudwatch_log_groups = Resources.*[ Type == %CLOUDWATCH_LOGS_TYPE ]

#
# Primary Rules
#
rule cloudwatch_log_group_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %cloudwatch_log_groups not empty {
    check(%cloudwatch_log_groups.Properties)
        <<
        [CT.CLOUDWATCH.PR.3]: Require an Amazon CloudWatch log group to be encrypted at
 rest with an AWS KMS key
        [FIX]: Set 'KmsKeyId' to the ARN of an AWS KMS customer managed key configured
 with permissions that allow the CloudWatch service principal to use the key.
        >>
}

rule cloudwatch_log_group_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDWATCH_LOGS_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDWATCH_LOGS_TYPE.resourceProperties)
        <<
        [CT.CLOUDWATCH.PR.3]: Require an Amazon CloudWatch log group to be encrypted at
 rest with an AWS KMS key
        [FIX]: Set 'KmsKeyId' to the ARN of an AWS KMS customer managed key configured
 with permissions that allow the CloudWatch service principal to use the key.
        >>
}

#
# Parameterized Rules
#
rule check(cloudwatch_log_group){
    %cloudwatch_log_group {
        # Scenario 2
        KmsKeyId exists
        # Scenario 3 and 4
        check_is_string_and_not_empty(KmsKeyId) or
```

```
        check_local_references(%INPUT_DOCUMENT, KmsKeyId, "AWS::KMS::Key")
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %resource_type
    }
}
```

CT.CLOUDWATCH.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-cloudwatch-logs-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: "*"
        - Sid: Enable Logs
          Effect: Allow
          Principal:
            Service:
              Fn::Sub: logs.${AWS::Region}.amazonaws.com
          Action:
          - kms:Encrypt*
          - kms:Decrypt*
          - kms:ReEncrypt*
          - kms:GenerateDataKey*
          - kms:Describe
          Resource: "*"
          Condition:
            ArnEquals:
              kms:EncryptionContext:aws:logs:arn:
                Fn::Sub: arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:*
  LogGroup:
    Type: AWS::Logs::LogGroup
```

```
    Properties:
      KmsKeyId:
        Fn::GetAtt:
        - KMSKey
        - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties: {}
```

[CT.CLOUDWATCH.PR.4] Require an Amazon CloudWatch alarm to have actions activated

This control checks whether an Amazon CloudWatch alarm has actions enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CloudWatch::Alarm`
- **CloudFormation guard rule:** CT.CLOUDWATCH.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CLOUDWATCH.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.CLOUDWATCH.PR.4 example templates

Explanation

Alarm actions automatically alert you when a monitored metric is outside the defined threshold. If the alarm action is deactivated, no actions are executed when the alarm changes state, so you won't be alerted to changes in monitored metrics. AWS Control Tower recommends activating CloudWatch alarm actions to help you respond quickly to security and operational issues.

Remediation for rule failure

Set `ActionsEnabled` to `true` or do not provide the `ActionsEnabled` property.

The examples that follow show how to implement this remediation.

Amazon CloudWatch Alarm - Example

An Amazon CloudWatch alarm configured with alarm actions enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "Alarm": {
        "Type": "AWS::CloudWatch::Alarm",
        "Properties": {
            "AlarmActions": [
                {
                    "Ref": "Topic"
                }
            ],
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "EvaluationPeriods": 1,
            "Period": 300,
            "Threshold": 1.0,
            "Namespace": "AWS/Lambda",
            "MetricName": "Errors",
            "TreatMissingData": "missing",
            "Statistic": "Sum",
            "DatapointsToAlarm": 1,
            "ActionsEnabled": true
        }
    }
}
```

YAML example

```
Alarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmActions:
      - !Ref 'Topic'
    ComparisonOperator: GreaterThanOrEqualToThreshold
    EvaluationPeriods: 1
    Period: 300
    Threshold: 1.0
    Namespace: AWS/Lambda
    MetricName: Errors
    TreatMissingData: missing
    Statistic: Sum
    DatapointsToAlarm: 1
    ActionsEnabled: true
```

CT.CLOUDWATCH.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   cloudwatch_alarm_action_enabled_check
#
# Description:
#   This control checks whether an Amazon CloudWatch alarm has actions enabled.
#
# Reports on:
#   AWS::CloudWatch::Alarm
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CloudWatch alarm resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'ActionsEnabled' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'ActionsEnabled' has not been provided
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains a CloudWatch alarm resource
#       And: 'ActionsEnabled' has been provided with a value of bool(true)
#      Then: PASS

#
# Constants
#
let CLOUDWATCH_ALARM_TYPE = "AWS::CloudWatch::Alarm"
let INPUT_DOCUMENT = this

#
# Assignments
#
let cloudwatch_alarms = Resources.*[ Type == %CLOUDWATCH_ALARM_TYPE ]

#
# Primary Rules
#
rule cloudwatch_alarm_action_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %cloudwatch_alarms not empty {
    check(%cloudwatch_alarms.Properties)
        <<
        [CT.CLOUDWATCH.PR.4]: Require an Amazon CloudWatch alarm to have actions
 activated
        [FIX]: Set 'ActionsEnabled' to 'true' or do not provide the 'ActionsEnabled'
 property.
```

```
        >>
}

rule cloudwatch_alarm_action_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CLOUDWATCH_ALARM_TYPE) {
    check(%INPUT_DOCUMENT.%CLOUDWATCH_ALARM_TYPE.resourceProperties)
        <<
        [CT.CLOUDWATCH.PR.4]: Require an Amazon CloudWatch alarm to have actions
 activated
        [FIX]: Set 'ActionsEnabled' to 'true' or do not provide the 'ActionsEnabled'
 property.
        >>
}

#
# Parameterized Rules
#
rule check(cloudwatch_alarm){
    %cloudwatch_alarm {
        # Scenario 3
        ActionsEnabled not exists or
        # Scenarios 2 and 4
        ActionsEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CLOUDWATCH.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties: {}
  Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmActions:
      - Ref: Topic
      ComparisonOperator: GreaterThanOrEqualToThreshold
      EvaluationPeriods: 1
      Period: 300
      Threshold: 1.0
      Namespace: AWS/Lambda
      MetricName: Errors
      TreatMissingData: missing
      Statistic: Sum
      DatapointsToAlarm: 1
      ActionsEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties: {}
  Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmActions:
      - Ref: Topic
```

```
      ComparisonOperator: GreaterThanOrEqualToThreshold
      EvaluationPeriods: 1
      Period: 300
      Threshold: 1.0
      Namespace: AWS/Lambda
      MetricName: Errors
      TreatMissingData: missing
      Statistic: Sum
      DatapointsToAlarm: 1
      ActionsEnabled: false
```

AWS CodeBuild controls

Topics

- [CT.CODEBUILD.PR.1] Require OAuth on GitHub or Bitbucket source repository URLs for AWS CodeBuild projects
- [CT.CODEBUILD.PR.2] Require any AWS CodeBuild project environment variable to encrypt credentials in environment variables
- [CT.CODEBUILD.PR.3] Require any AWS CodeBuild project environment to have logging configured
- [CT.CODEBUILD.PR.5] Require encryption on all AWS CodeBuild project artifacts
- [CT.CODEBUILD.PR.6] Require encryption on all Amazon S3 logs for AWS CodeBuild projects

[CT.CODEBUILD.PR.1] Require OAuth on GitHub or Bitbucket source repository URLs for AWS CodeBuild projects

This control checks whether the GitHub or Bitbucket source repository URL contains either personal access tokens or a user name and password.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CodeBuild::Project`
- **CloudFormation guard rule:** CT.CODEBUILD.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CODEBUILD.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CODEBUILD.PR.1 example templates

Explanation

Authentication credentials should never be stored or transmitted in clear text or appear in the repository URL. Instead of personal access tokens or username and password, you should use OAuth to grant authorization for accessing GitHub or Bitbucket repositories. Using personal access tokens or a username and password could expose your credentials to unintended data exposure and unauthorized access.

Usage considerations

- This control applies only to AWS CodeBuild projects with a primary or secondary source type of `GitHub` or `Bitbucket`.

Remediation for rule failure

Remove any embedded credentials from repository URLs in AWS CodeBuild project source configurations. Instead, connect your CodeBuild projects to `GitHub` or `Bitbucket` repositories by configuring `GitHub Access Token` or `Bitbucket App Password` credentials in the AWS Management Console or AWS CLI.

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example One

AWS CodeBuild project configured with a GitHub primary source location that does not contain a personal access token. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n",
                "Type": "GITHUB",
                "Location": "https://github.com/username/repo.git"
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
```

```
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
      Type: GITHUB
      Location: https://github.com/username/repo.git
```

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example Two

AWS CodeBuild project configured with primary and secondary source locations that do not contain credentials or personal access tokens. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
```

```
                ]
            },
            "Source": {
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n",
                "Type": "BITBUCKET",
                "Location": "https://bitbucket.org/user/repo.git"
            },
            "SecondarySources": [
                {
                    "Type": "GITHUB",
                    "Location": "https://github.com/username/repo.git",
                    "SourceIdentifier": "GitHubSource"
                }
            ]
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
```

```
              - npm test
        artifacts:
          files:
            - '**/*'
      Type: BITBUCKET
      Location: https://bitbucket.org/user/repo.git
    SecondarySources:
      - Type: GITHUB
        Location: https://github.com/username/repo.git
        SourceIdentifier: GitHubSource
```

CT.CODEBUILD.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   codebuild_project_source_repo_url_check
#
# Description:
#   This control checks whether the GitHub or Bitbucket source repository URL contains
 either personal access tokens or a username and password.
#
# Reports on:
#   AWS::CodeBuild::Project
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CodeBuild project resources
#       Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a CodeBuild project resource
#       And: 'Source' configuration is not of 'Type' 'GITHUB' or 'BITBUCKET'
#       And: 'SecondarySources' configuration is not provided or is provided and does
 not have any item of 'Type'
#            'GITHUB' or 'BITBUCKET'
#       Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Source' configuration is of 'Type' 'GITHUB' or 'BITBUCKET'
#       And: 'Source' configuration has a 'Location' that contains credentials
 (username and password for BitBucket
#            and Access Token for GitHub)
#       Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'SecondarySources' configuration is provided
#       And: 'SecondarySources' configuration has one or more items of 'Type' 'GITHUB'
 or 'BITBUCKET'
#       And: 'SecondarySources' configuration has one or more items with 'Location'
 that contains credentials
#            (username and password for BitBucket and Access Token for GitHub)
#       Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Source' configuration is of 'Type' 'GITHUB' or 'BITBUCKET'
#       And: 'Source' configuration has a 'Location' that does not contain credentials
 (username and password for
#            BitBucket and Access Token for GitHub)
#       And: 'SecondarySources' configuration is not provided or is provided and does
 not have any item of 'Type'
#            'GITHUB' or 'BITBUCKET'
#       Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Source' configuration is of 'Type' 'GITHUB' or 'BITBUCKET'
#       And: 'Source' configuration has a 'Location' that does not contain credentials
 (username and password for
#            BitBucket and Access Token for GitHub)
#       And: 'SecondarySources' configuration is provided
```

```
#       And: 'SecondarySources' configuration has one or more items of 'Type' 'GITHUB'
 or 'BITBUCKET'
#       And: 'SecondarySources' configuration has one or more items with 'Location'
 that does not contain credentials
#            (username and password for BitBucket and Access Token for GitHub)
#       Then: PASS

#
# Constants
#
let CODEBUILD_PROJECT_TYPE = "AWS::CodeBuild::Project"
let INPUT_DOCUMENT = this
let GITHUB_COMPLIANT_URL_PATTERN = /^(http(s)?)(:\/\/github\.com\/)([^\/]+)\/([\w\.-]+)
(\.git)?$/
let BITBUCKET_COMPLIANT_URL_PATTERN = /^https?:\/\/bitbucket\.org/

#
# Assignments
#
let codebuild_project = Resources.*[ Type == %CODEBUILD_PROJECT_TYPE ]

#
# Primary Rules
#
rule codebuild_project_source_repo_url_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %codebuild_project not empty {
    check(%codebuild_project.Properties)
        <<
        [CT.CODEBUILD.PR.1]: Require OAuth on GitHub or Bitbucket source repository
 URLs for AWS CodeBuild projects
            [FIX]: Remove any embedded credentials from repository URLs in AWS
 CodeBuild project source configurations. Instead, connect your CodeBuild projects
 to 'GitHub' or 'Bitbucket' repositories by configuring 'GitHub Access Token' or
 'Bitbucket App Password' credentials in the AWS Management Console or AWS CLI.
        >>
}

rule codebuild_project_source_repo_url_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CODEBUILD_PROJECT_TYPE) {
    check(%INPUT_DOCUMENT.%CODEBUILD_PROJECT_TYPE.resourceProperties)
        <<
        [CT.CODEBUILD.PR.1]: Require OAuth on GitHub or Bitbucket source repository
 URLs for AWS CodeBuild projects
```

```
            [FIX]: Remove any embedded credentials from repository URLs in AWS
 CodeBuild project source configurations. Instead, connect your CodeBuild projects
 to 'GitHub' or 'Bitbucket' repositories by configuring 'GitHub Access Token' or
 'Bitbucket App Password' credentials in the AWS Management Console or AWS CLI.
        >>
}

#
# Parameterized Rules
#
rule check(codebuild_project) {
    %codebuild_project[
        filter_github_or_bitbucket_source_configuration(this) or
        filter_github_or_bitbucket_secondary_sources_configuration(this)
    ] {
        # Scenario 3, 5 and 6
        check_source(Source)
        # Scenario 4 and 6
        check_secondary_sources(this)
    }
}

rule filter_github_or_bitbucket_source_configuration(codebuild_project) {
    %codebuild_project {
        Source exists
        Source is_struct
        Source {
            Type == "GITHUB" or
            Type == "BITBUCKET"
        }
    }
}

rule filter_github_or_bitbucket_secondary_sources_configuration(codebuild_project) {
    %codebuild_project {
        SecondarySources exists
        SecondarySources is_list
        SecondarySources not empty

        some SecondarySources[*] {
            Type == "GITHUB" or
            Type == "BITBUCKET"
        }
    }
```

```
}

rule check_source(codebuild_source) {
    %codebuild_source [
        Type == "GITHUB"
    ] {
        Location exists
        Location == %GITHUB_COMPLIANT_URL_PATTERN
    }
    %codebuild_source [
        Type == "BITBUCKET"
    ] {
        Location exists
        Location == %BITBUCKET_COMPLIANT_URL_PATTERN
    }
}

rule check_secondary_sources(codebuild_project) {
    %codebuild_project [
        # Scenario 2
        SecondarySources exists
        SecondarySources is_list
        SecondarySources not empty
    ] {
        # Scenario 4 and 6
        SecondarySources[*] {
            check_source(this)
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CODEBUILD.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
```

```
            commands:
              - npm test
          artifacts:
            files:
              - '**/*'
        Type: GITHUB
        Location: https://github.com/username/repo.git
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        BuildSpec: |
          version: 0.2
          phases:
            install:
```

```
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
      Type: BITBUCKET
      Location: https://username:password@bitbucket.org/user/repo.git
```

[CT.CODEBUILD.PR.2] Require any AWS CodeBuild project environment variable to encrypt credentials in environment variables

This control checks whether AWS CodeBuild projects contain environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` stored as `PLAINTEXT`.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CodeBuild::Project`
- **CloudFormation guard rule:** CT.CODEBUILD.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CODEBUILD.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CODEBUILD.PR.2 example templates

Explanation

Authentication credentials AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY should never be stored in clear text, as this could lead to unintended data exposure and unauthorized access.

Usage considerations

- This control only applies to AWS CodeBuild projects configured with `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables

Remediation for rule failure

Use `PARAMETER_STORE` or `SECRETS_MANAGER` to store values for environment variables named `AWS_ACCESS_KEY_ID` or `AWS_SECRET_ACCESS_KEY`.

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example

AWS CodeBuild project configured to use credentials stored in AWS Secrets Manager. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER",
                "EnvironmentVariables": [
                    {
                        "Name": "AWS_ACCESS_KEY_ID",
                        "Type": "SECRETS_MANAGER",
                        "Value": "sample_secret:access_key_id"
                    },
                    {
                        "Name": "AWS_SECRET_ACCESS_KEY",
                        "Type": "SECRETS_MANAGER",
                        "Value": "sample_secret:secret_access_key"
```

```
                    }
                ]
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n"
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
      EnvironmentVariables:
        - Name: AWS_ACCESS_KEY_ID
          Type: SECRETS_MANAGER
          Value: sample_secret:access_key_id
        - Name: AWS_SECRET_ACCESS_KEY
          Type: SECRETS_MANAGER
          Value: sample_secret:secret_access_key
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: |
```

```
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
```

CT.CODEBUILD.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   codebuild_project_envvar_awscred_check
#
# Description:
#   This control checks whether AWS CodeBuild projects contain environment variables
 'AWS_ACCESS_KEY_ID' and 'AWS_SECRET_ACCESS_KEY' stored as 'PLAINTEXT'.
#
# Reports on:
#   AWS::CodeBuild::Project
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CodeBuild project resources
#     Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Environment' configuration does not contains 'EnvironmentVariables'
#     Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Environment' configuration contains 'EnvironmentVariables'
#       And: 'EnvironmentVariables' contain variables named 'AWS_ACCESS_KEY_ID' or
 'AWS_SECRET_ACCESS_KEY'
#       And: 'Type' is not provided for 'AWS_ACCESS_KEY_ID' and 'AWS_SECRET_ACCESS_KEY'
 environment variables or is
#            provided as an empty string.
#     Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Environment' configuration contains 'EnvironmentVariables'
#       And: 'EnvironmentVariables' contain variables named 'AWS_ACCESS_KEY_ID' or
 'AWS_SECRET_ACCESS_KEY'
#       And: 'Type' is set to 'PLAINTEXT' for 'AWS_ACCESS_KEY_ID' or
 'AWS_SECRET_ACCESS_KEY' environment variables
#     Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Environment' configuration contains 'EnvironmentVariables'
#       And: 'EnvironmentVariables' does not contain variables named
 'AWS_ACCESS_KEY_ID' or 'AWS_SECRET_ACCESS_KEY'
#     Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Environment' configuration contains 'EnvironmentVariables'
#       And: 'EnvironmentVariables' contain variables named 'AWS_ACCESS_KEY_ID' or
 'AWS_SECRET_ACCESS_KEY'
#       And: 'Type' is provided as a non-empty string and not set to 'PLAINTEXT' for
 'AWS_ACCESS_KEY_ID' or
#            'AWS_SECRET_ACCESS_KEY' environment variables
#     Then: PASS


#
# Constants
```

```
#
let CODEBUILD_PROJECT_TYPE = "AWS::CodeBuild::Project"
let AWS_CREDENTIAL_ENV_VAR_NAMES = [ "AWS_ACCESS_KEY_ID", "AWS_SECRET_ACCESS_KEY" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let codebuild_project = Resources.*[ Type == %CODEBUILD_PROJECT_TYPE ]

#
# Primary Rules
#
rule codebuild_project_envvar_awscred_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %codebuild_project not empty {
    check(%codebuild_project.Properties)
        <<
        [CT.CODEBUILD.PR.2]: Require any AWS CodeBuild project environment variable to
 encrypt credentials in environment variables
            [FIX]: Use 'PARAMETER_STORE' or 'SECRETS_MANAGER' to store values for
 environment variables named 'AWS_ACCESS_KEY_ID' or 'AWS_SECRET_ACCESS_KEY'.
        >>
}

rule codebuild_project_envvar_awscred_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CODEBUILD_PROJECT_TYPE) {
    check(%INPUT_DOCUMENT.%CODEBUILD_PROJECT_TYPE.resourceProperties)
        <<
        [CT.CODEBUILD.PR.2]: Require any AWS CodeBuild project environment variable to
 encrypt credentials in environment variables
            [FIX]: Use 'PARAMETER_STORE' or 'SECRETS_MANAGER' to store values for
 environment variables named 'AWS_ACCESS_KEY_ID' or 'AWS_SECRET_ACCESS_KEY'.
        >>
}

#
# Parameterized Rules
#
rule check(codebuild_project) {
    %codebuild_project [
        # Scenario 2
        filter_codebuild_projects_with_environment_variables(this)
    ] {
        Environment exists
```

```
        Environment is_struct
        Environment {
            EnvironmentVariables exists
            EnvironmentVariables is_list
            EnvironmentVariables not empty
            EnvironmentVariables [
                # Scenario 3, 4 and 6
                Name in %AWS_CREDENTIAL_ENV_VAR_NAMES
            ] {
                # Scenario 3
                Type exists
                check_is_string_and_not_empty(Type)
                # Scenario 4 and 6
                Type != "PLAINTEXT"
            }
        }
    }
}

rule filter_codebuild_projects_with_environment_variables(codebuild_project) {
    %codebuild_project {
        Environment exists
        Environment is_struct

        Environment {
            # Scenario 2
            EnvironmentVariables exists
            EnvironmentVariables is_list
            EnvironmentVariables not empty
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.CODEBUILD.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: AWS_ACCESS_KEY_ID
          Type: SECRETS_MANAGER
```

```
          Value: example_secret:access_key_id
        - Name: AWS_SECRET_ACCESS_KEY
          Type: SECRETS_MANAGER
          Value: example_secret:secret_access_key
        - Name: some_other_variable
          Type: PLAINTEXT
          Value: example
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
          artifacts:
            files:
              - '**/*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
```

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: AWS_ACCESS_KEY_ID
          Type: PLAINTEXT
          Value: EXAMPLE_ACCESS_KEY_ID
        - Name: AWS_SECRET_ACCESS_KEY
          Type: PLAINTEXT
          Value: EXAMPLE_SECRET_ACCESS_KEY
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
          artifacts:
            files:
              - '**/*'
```

[CT.CODEBUILD.PR.3] Require any AWS CodeBuild project environment to have logging configured

This control checks whether AWS CodeBuild projects environment has at least one logging option enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CodeBuild::Project`
- **CloudFormation guard rule:** CT.CODEBUILD.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CODEBUILD.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CODEBUILD.PR.3 example templates

Explanation

From a security perspective, logging is an important feature to enable, to assist future forensics efforts in case of a security incident. Correlating anomalies in CodeBuild projects with threat detections can increase confidence in the accuracy of those threat detections.

Remediation for rule failure

Set `LogsConfig` with a `CloudWatchLogs` or `S3Logs` configuration.

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example One

AWS CodeBuild project configured to enable logging, by means of Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
```

```
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n      -
 npm install\n  build:\n    commands:\n      - npm test\n"
            },
            "LogsConfig": {
                "CloudWatchLogs": {
                    "Status": "ENABLED"
                }
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
```

```
      BuildSpec: "version: 0.2\nphases:\n  install:\n    commands:\n      - npm install
\n\
        \  build:\n    commands:\n      - npm test\n"
    LogsConfig:
      CloudWatchLogs:
        Status: ENABLED
```

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example Two

AWS CodeBuild project configured to enable logging, by means of Amazon S3. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n      -
 npm install\n  build:\n    commands:\n      - npm test\n"
            },
            "LogsConfig": {
```

```
                "S3Logs": {
                    "Status": "ENABLED",
                    "Location": {
                        "Fn::Join": [
                            "/",
                            [
                                {
                                    "Ref": "S3Bucket"
                                },
                                "path/to/directory"
                            ]
                        ]
                    }
                }
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: "version: 0.2\nphases:\n  install:\n    commands:\n      - npm install
\n\
        \  build:\n    commands:\n      - npm test\n"
    LogsConfig:
      S3Logs:
        Status: ENABLED
        Location: !Join
          - /
```

```
          - - !Ref 'S3Bucket'
            - path/to/directory
```

CT.CODEBUILD.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   codebuild_project_logging_enabled_check
#
# Description:
#   This control checks whether AWS CodeBuild projects environment has at least one
 logging option enabled.
#
# Reports on:
#   AWS::CodeBuild::Project
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CodeBuild project resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is not provided on the CodeBuild project resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: Neither 'CloudWatchLogs' or 'S3Logs' are present in 'LogsConfig'
```

```
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'CloudWatchLogs' is not present in 'LogsConfig'
#       And: 'S3Logs' is present in 'LogsConfig' with 'Status' set to 'DISABLED'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'S3Logs' is not present in 'LogsConfig'
#       And: 'CloudWatchLogs' is present in 'LogsConfig' with 'Status' set to
 'DISABLED'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'CloudWatchLogs' and 'S3Logs' are present in 'LogsConfig' with 'Status'
 set to 'DISABLED'
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'CloudWatchLogs' is not present in 'LogsConfig'
#       And: 'S3Logs' is present in 'LogsConfig' with 'Status' set to 'ENABLED'
#       And: 'Location' has not been provided in 'S3Logs', or has been provided as an
 empty string or
#             invalid local reference
#      Then: FAIL
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'S3Logs' is not present in 'LogsConfig'
#       And: 'CloudWatchLogs' is present in 'LogsConfig' with 'Status' set to 'ENABLED'
#      Then: PASS
#   Scenario: 9
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
```

```
#       And: 'CloudWatchLogs' is not present in 'LogsConfig'
#       And: 'S3Logs' is present in 'LogsConfig' with 'Status' set to 'ENABLED'
#       And: 'Location' has been provided in 'S3Logs' as a non-empty string or valid
 local reference
#      Then: PASS
#   Scenario: 10
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'LogsConfig' is provided on the CodeBuild project resource
#       And: 'CloudWatchLogs' is present in 'LogsConfig' with 'Status' set to 'ENABLED'
#       And: 'S3Logs' is present in 'LogsConfig' with 'Status' set to 'ENABLED'
#       And: 'Location' has been provided in 'S3Logs' as a non-empty string or valid
 local reference
#      Then: PASS

#
# Constants
#
let CODEBUILD_PROJECT_TYPE = "AWS::CodeBuild::Project"
let INPUT_DOCUMENT = this

#
# Assignments
#
let codebuild_project = Resources.*[ Type == %CODEBUILD_PROJECT_TYPE ]

#
# Primary Rules
#
rule codebuild_project_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %codebuild_project not empty {
    check(%codebuild_project.Properties)
        <<
        [CT.CODEBUILD.PR.3]: Require any AWS CodeBuild project environment to have
 logging configured
            [FIX]: Set 'LogsConfig' with a 'CloudWatchLogs' or 'S3Logs' configuration.
        >>
}

rule codebuild_project_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CODEBUILD_PROJECT_TYPE) {
    check(%INPUT_DOCUMENT.%CODEBUILD_PROJECT_TYPE.resourceProperties)
        <<
```

```
        [CT.CODEBUILD.PR.3]: Require any AWS CodeBuild project environment to have
 logging configured
            [FIX]: Set 'LogsConfig' with a 'CloudWatchLogs' or 'S3Logs' configuration.
        >>
}

#
# Parameterized Rules
#
rule check(codebuild_project) {
    %codebuild_project {
        # Scenario 2
        LogsConfig exists
        LogsConfig is_struct

        LogsConfig {
            # Scenario 3
            check_cloudwatch_logs(this) or
            check_s3_logs(this)
        }
    }
}

rule check_cloudwatch_logs(codebuild_project) {
    %codebuild_project {
        # Scenario 4
        CloudWatchLogs exists
        CloudWatchLogs is_struct

        CloudWatchLogs {
            # Scenario 5, 6, 8 and 10
            Status exists
            Status == "ENABLED"
        }
    }
}

rule check_s3_logs(codebuild_project) {
    %codebuild_project {
        # Scenario 4
        S3Logs exists
        S3Logs is_struct

        S3Logs {
```

```
            # Scenario 4, 6, 9 and 10
            Status exists
            Status == "ENABLED"

            # Scenario 7, 9 and 10
            Location exists
            check_is_string_and_not_empty(Location) or
            check_local_references(%INPUT_DOCUMENT, Location, "AWS::S3::Bucket") or
            check_join_references(%INPUT_DOCUMENT, Location, "AWS::S3::Bucket")
        }
    }
}

rule check_join_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::Join' {
            this is_list
            this not empty
            some this[1][*] {
                check_local_references(%doc, this, %referenced_resource_type)
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
```

```
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.CODEBUILD.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      Path: /
```

```
      Policies:
      - PolicyName: CodeBuildProjectPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
      LogsConfig:
        CloudWatchLogs:
          Status: ENABLED
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
      LogsConfig:
        S3Logs:
          Status: DISABLED
        CloudWatchLogs:
          Status: DISABLED
```

[CT.CODEBUILD.PR.5] Require encryption on all AWS CodeBuild project artifacts

This control checks whether AWS CodeBuild projects are configured to encrypt artifacts.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CodeBuild::Project`
- **CloudFormation guard rule:** CT.CODEBUILD.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CODEBUILD.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CODEBUILD.PR.5 example templates

Explanation

Encryption of data at rest is a recommended best practice. It adds a layer of access management around your data. In case of a compromise to your CodeBuild artifacts, encryption at rest ensures that your data is protected from unintended access.

Usage considerations

- This control applies only to AWS CodeBuild projects configured to return primary or secondary artifacts as output.

Remediation for rule failure

Set the `EncryptionDisabled` property in `Artifacts` and any `SecondaryArtifacts` to `false`, or omit the `EncryptionDisabled` property.

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example One

AWS CodeBuild project configured to return primary artifacts as output with artifact encryption enabled, by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n"
            },
            "Artifacts": {
                "Type": "S3",
                "Location": {
                    "Ref": "S3Bucket"
                }
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
    Artifacts:
      Type: S3
      Location: !Ref 'S3Bucket'
```

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example Two

AWS CodeBuild project configured to return primary and secondary artifacts as output with artifact encryption enabled, by means of the `EncryptionDisabled` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
```

```
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
  - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n
  - '**/*'\n  secondary-artifacts:\n    secondaryArtifact:\n      files:\n        -
 'directory/file1'\n"
            },
            "Artifacts": {
                "Type": "S3",
                "EncryptionDisabled": false,
                "Location": {
                    "Ref": "S3Bucket"
                }
            },
            "SecondaryArtifacts": [
                {
                    "Type": "S3",
                    "EncryptionDisabled": false,
                    "ArtifactIdentifier": "secondaryArtifact",
                    "Location": {
                        "Ref": "S3Bucket"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
          secondary-artifacts:
            secondaryArtifact:
              files:
                - 'directory/file1'
    Artifacts:
      Type: S3
      EncryptionDisabled: false
      Location: !Ref 'S3Bucket'
    SecondaryArtifacts:
      - Type: S3
        EncryptionDisabled: false
        ArtifactIdentifier: secondaryArtifact
        Location: !Ref 'S3Bucket'
```

CT.CODEBUILD.PR.5 rule specification

```
# ###################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   codebuild_project_artifact_encryption_check
#
# Description:
#   This control checks whether AWS CodeBuild projects are configured to encrypt
 artifacts.
#
# Reports on:
#   AWS::CodeBuild::Project
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario 1:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CodeBuild project resources
#      Then: SKIP
#   Scenario 2:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Artifacts' configuration is provided and is of 'Type' 'NO_ARTIFACTS'
#       And: 'SecondaryArtifacts' configuration is not provided or provided with an
 empty list
#      Then: SKIP
#   Scenario 3:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Artifacts' configuration is provided and is of 'Type' 'NO_ARTIFACTS'
#       And: 'SecondaryArtifacts' configuration is provided as a non-empty list
#       And: All 'SecondaryArtifacts' entries have 'Type' set to 'NO_ARTIFACTS'
#      Then: SKIP
#   Scenario 4:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Artifacts' configuration is provided and is not of 'Type' 'NO_ARTIFACTS'
#       And: 'EncryptionDisabled' within 'Artifacts' configuration is provided and set
 to bool(true)
#      Then: FAIL
```

```
#   Scenario 5:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'SecondaryArtifacts' configuration is provided
#       And: There exists one or more items in 'SecondaryArtifacts' which have
 'EncryptionDisabled' set to bool(true)
#      Then: FAIL
#   Scenario 6:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'Artifacts.EncryptionDisabled' is not provided, or is set to bool(false)
#       And: There exists no item in 'SecondaryArtifacts' which has
 'EncryptionDisabled' set to bool(true)
#      Then: PASS

#
# Constants
#
let CODEBUILD_PROJECT_TYPE = "AWS::CodeBuild::Project"
let INPUT_DOCUMENT = this

#
# Assignments
#
let codebuild_project = Resources.*[ Type == %CODEBUILD_PROJECT_TYPE ]

#
# Primary Rules
#
rule codebuild_project_artifact_encryption_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %codebuild_project not empty {
    check(%codebuild_project.Properties)
        <<
        [CT.CODEBUILD.PR.5]: Require encryption on all AWS CodeBuild project artifacts
            [FIX]: Set the 'EncryptionDisabled' property in 'Artifacts' and any
 'SecondaryArtifacts' to 'false', or omit the 'EncryptionDisabled' property.
        >>
}

rule codebuild_project_artifact_encryption_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CODEBUILD_PROJECT_TYPE) {
    check(%INPUT_DOCUMENT.%CODEBUILD_PROJECT_TYPE.resourceProperties)
        <<
        [CT.CODEBUILD.PR.5]: Require encryption on all AWS CodeBuild project artifacts
```

```
            [FIX]: Set the 'EncryptionDisabled' property in 'Artifacts' and any
 'SecondaryArtifacts' to 'false', or omit the 'EncryptionDisabled' property.
        >>
}

#
# Parameterized Rules
#
rule check(codebuild_project) {
    %codebuild_project [
        filter_codebuild_projects(this)
    ] {
        Artifacts {
            # Scenario 4 and 6
            check_artifact(this)
        }
        # Scenario 5
        SecondaryArtifacts not exists or
        check_secondary_artifacts(this)
    }
}

rule check_secondary_artifacts(codebuild_project) {
    %codebuild_project {
        SecondaryArtifacts is_list
        SecondaryArtifacts[*] {
            # Scenario 5 and 6
            check_artifact(this)
        }
    }
}

rule check_artifact(artifact) {
    %artifact {
        EncryptionDisabled not exists or
        EncryptionDisabled == false
    }
}

rule filter_codebuild_projects(codebuild_project) {
    %codebuild_project {
        # Scenario 2 and 3
        Artifacts exists
        Artifacts is_struct
```

```
        Artifacts {
            filter_artifact(this)
        } or
        filter_secondary_artifacts(this)
    }
}

rule filter_secondary_artifacts(codebuild_project) {
    %codebuild_project {
        # Scenario 2
        SecondaryArtifacts exists
        SecondaryArtifacts is_list
        SecondaryArtifacts not empty
        SecondaryArtifacts[*] {
            # Scenario 3
            filter_artifact(this)
        }
    }
}

rule filter_artifact(artifact) {
    %artifact {
       Type exists
       Type != "NO_ARTIFACTS"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CODEBUILD.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: CodeBuildProjectPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - s3:PutObject
            - s3:GetBucketAcl
            - s3:GetBucketLocation
            Resource:
            - Fn::GetAtt:
              - S3Bucket
              - Arn
            - Fn::Join:
              - ""
              - - Fn::GetAtt:
                  - S3Bucket
```

```
                - Arn
              - "/*"
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
          artifacts:
            files:
              - '**/*'
      Artifacts:
        Type: S3
        Location:
          Ref: S3Bucket
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
```

```
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: CodeBuildProjectPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - s3:PutObject
            - s3:GetBucketAcl
            - s3:GetBucketLocation
            Resource:
            - Fn::GetAtt:
              - S3Bucket
              - Arn
            - Fn::Join:
              - ""
              - - Fn::GetAtt:
                  - S3Bucket
                  - Arn
                - "/*"
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
```

```
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
          artifacts:
            files:
              - '**/*'
      Artifacts:
        Type: S3
        EncryptionDisabled: true
        Location:
          Ref: S3Bucket
```

[CT.CODEBUILD.PR.6] Require encryption on all Amazon S3 logs for AWS CodeBuild projects

This control checks whether AWS CodeBuild projects configured with Amazon S3 logs have encryption enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::CodeBuild::Project`
- **CloudFormation guard rule:** CT.CODEBUILD.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.CODEBUILD.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.CODEBUILD.PR.6 example templates

Explanation

Encryption of data at rest is a recommended best practice. It adds a layer of access management around your data. In case of a compromise to your CodeBuild artifacts, encryption at rest ensures that your data is protected from unintended access.

Usage considerations

- This control applies only to AWS CodeBuild projects with log delivery to Amazon S3 enabled.

Remediation for rule failure

Set `EncryptionDisabled` in `S3Logs` to `false`, or do not specify the `EncryptionDisabled` property.

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example One

AWS CodeBuild project configured to encrypt logs delivered to an Amazon S3 logging destination, by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
```

```
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n"
            },
            "LogsConfig": {
                "S3Logs": {
                    "Status": "ENABLED",
                    "Location": {
                        "Ref": "S3Bucket"
                    }
                }
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
```

```
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
    LogsConfig:
      S3Logs:
        Status: ENABLED
        Location: !Ref 'S3Bucket'
```

The examples that follow show how to implement this remediation.

AWS CodeBuild Project - Example Two

AWS CodeBuild project configured to encrypt logs delivered to an Amazon S3 logging destination, by means of the `EncryptionDisabled` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "CodeBuildProject": {
        "Type": "AWS::CodeBuild::Project",
        "Properties": {
            "Artifacts": {
                "Type": "NO_ARTIFACTS"
            },
            "Environment": {
                "ComputeType": "BUILD_GENERAL1_SMALL",
                "Image": "aws/codebuild/standard:4.0",
                "Type": "LINUX_CONTAINER"
```

```
            },
            "ServiceRole": {
                "Fn::GetAtt": [
                    "CodeBuildServiceRole",
                    "Arn"
                ]
            },
            "Source": {
                "Type": "NO_SOURCE",
                "BuildSpec": "version: 0.2\nphases:\n  install:\n    commands:\n
 - npm install\n  build:\n    commands:\n      - npm test\nartifacts:\n  files:\n    -
 '**/*'\n"
            },
            "LogsConfig": {
                "S3Logs": {
                    "Status": "ENABLED",
                    "Location": {
                        "Ref": "S3Bucket"
                    },
                    "EncryptionDisabled": false
                }
            }
        }
    }
}
```

YAML example

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
      Type: LINUX_CONTAINER
    ServiceRole: !GetAtt 'CodeBuildServiceRole.Arn'
    Source:
      Type: NO_SOURCE
      BuildSpec: |
```

```
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
    LogsConfig:
      S3Logs:
        Status: ENABLED
        Location: !Ref 'S3Bucket'
        EncryptionDisabled: false
```

CT.CODEBUILD.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   codebuild_project_s3_logs_encrypted_check
#
# Description:
#   This control checks whether AWS CodeBuild projects configured with Amazon S3 logs
 have encryption enabled.
#
# Reports on:
#   AWS::CodeBuild::Project
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario 1:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any CodeBuild project resources
#      Then: SKIP
#   Scenario 2:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'S3Logs' in 'LogsConfig' configuration is not provided
#      Then: SKIP
#   Scenario 3:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'S3Logs' in 'LogsConfig' configuration is provided and its 'Status' is set
 to 'DISABLED'
#      Then: SKIP
#   Scenario 4:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'S3Logs' in 'LogsConfig' configuration is provided and its 'Status' is set
 to 'ENABLED'
#       And: 'EncryptionDisabled' within 'S3Logs' is provided and set to bool(true)
#      Then: FAIL
#   Scenario 5:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'S3Logs' in 'LogsConfig' configuration is provided and its 'Status' is set
 to 'ENABLED'
#       And: 'EncryptionDisabled' within 'S3Logs' is not provided
#      Then: PASS
#   Scenario 6:
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a CodeBuild project resource
#       And: 'S3Logs' in 'LogsConfig' configuration is provided and its 'Status' is set
 to 'ENABLED'
#       And: 'EncryptionDisabled' within 'S3Logs' is provided and set to bool(false)
#      Then: PASS

#
# Constants
#
let CODEBUILD_PROJECT_TYPE = "AWS::CodeBuild::Project"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let codebuild_project = Resources.*[ Type == %CODEBUILD_PROJECT_TYPE ]

#
# Primary Rules
#
rule codebuild_project_s3_logs_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %codebuild_project not empty {
    check(%codebuild_project.Properties)
        <<
        [CT.CODEBUILD.PR.6]: Require encryption on all Amazon S3 logs for AWS CodeBuild
 projects
            [FIX]: Set 'EncryptionDisabled' in 'S3Logs' to 'false', or do not specify
 the 'EncryptionDisabled' property.
        >>
}

rule codebuild_project_s3_logs_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %CODEBUILD_PROJECT_TYPE) {
    check(%INPUT_DOCUMENT.%CODEBUILD_PROJECT_TYPE.resourceProperties)
        <<
        [CT.CODEBUILD.PR.6]: Require encryption on all Amazon S3 logs for AWS CodeBuild
 projects
            [FIX]: Set 'EncryptionDisabled' in 'S3Logs' to 'false', or do not specify
 the 'EncryptionDisabled' property.
        >>
}

#
# Parameterized Rules
#
rule check(codebuild_project) {
    %codebuild_project [
        # Scenario 2 and 3
        filter_codebuild_projects(this)
    ] {
        LogsConfig {
            S3Logs {
                # Scenario 5
                EncryptionDisabled not exists or
                # Scenario 4 and 6
                EncryptionDisabled == false
            }
```

```
        }
    }
}

rule filter_codebuild_projects(codebuild_project) {
    %codebuild_project {
        LogsConfig exists
        LogsConfig is_struct
        LogsConfig {
            # Scenario 2 and 3
            S3Logs exists
            S3Logs is_struct
            S3Logs {
                Status exists
                # Scenario 3 and 4
                Status == "ENABLED"
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.CODEBUILD.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: CodeBuildProjectPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - s3:PutObject
            - s3:GetBucketAcl
            - s3:GetBucketLocation
            Resource:
              - Fn::GetAtt:
                - S3Bucket
                - Arn
              - Fn::Join:
                - ""
                - - Fn::GetAtt:
                    - S3Bucket
                    - Arn
                  - "/*"
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
  CodeBuildProject:
```

```
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
        Type: LINUX_CONTAINER
      ServiceRole:
        Fn::GetAtt:
        - CodeBuildServiceRole
        - Arn
      Source:
        Type: NO_SOURCE
        BuildSpec: |
          version: 0.2
          phases:
            install:
              commands:
                - npm install
            build:
              commands:
                - npm test
          artifacts:
            files:
              - '**/*'
      LogsConfig:
        S3Logs:
          Status: ENABLED
          Location:
            Ref: S3Bucket
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  CodeBuildServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: CodeBuildProjectPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - s3:PutObject
            - s3:GetBucketAcl
            - s3:GetBucketLocation
            Resource:
              - Fn::GetAtt:
                - S3Bucket
                - Arn
              - Fn::Join:
                - ""
                - - Fn::GetAtt:
                    - S3Bucket
                    - Arn
                  - "/*"
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: NO_ARTIFACTS
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:4.0
```

```
      Type: LINUX_CONTAINER
    ServiceRole:
      Fn::GetAtt:
      - CodeBuildServiceRole
      - Arn
    Source:
      Type: NO_SOURCE
      BuildSpec: |
        version: 0.2
        phases:
          install:
            commands:
              - npm install
          build:
            commands:
              - npm test
        artifacts:
          files:
            - '**/*'
    LogsConfig:
      S3Logs:
        Status: ENABLED
        Location:
          Ref: S3Bucket
        EncryptionDisabled: true
```

AWS Database Migration Service (AWS DMS) controls

Topics

- [CT.DMS.PR.1] Require that a public AWS DMS replication instance is not public
- [CT.DMS.PR.2] Require an AWS Database Migration Service (DMS) Endpoint to encrypt connections for source and target endpoints

[CT.DMS.PR.1] Require that a public AWS DMS replication instance is not public

This control checks whether your AWS DMS replication instance is public.

- **Control objective:** Limit network access, Enforce least privilege
- **Implementation:** CloudFormation Guard Rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::DMS::ReplicationInstance`
- **CloudFormation guard rule:** CT.DMS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DMS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DMS.PR.1 example templates

Explanation

A private replication instance has a private IP address that you cannot access outside of the replication network. You use a private instance when both source and target databases are in the same network that is connected to the replication instance's VPC. The network can be connected to the VPC by using a VPN, AWS Direct Connect, or VPC peering.

Remediation for rule failure

Set `PubliclyAccessible` to `false`.

The examples that follow show how to implement this remediation.

AWS DMS Replication Instance - Example

AWS DMS replication instance configured with public access disabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "DMSReplicationInstance": {
        "Type": "AWS::DMS::ReplicationInstance",
        "Properties": {
            "ReplicationInstanceClass": "dms.t3.micro",
            "PubliclyAccessible": false
        }
    }
}
```

YAML example

```
DMSReplicationInstance:
  Type: AWS::DMS::ReplicationInstance
  Properties:
    ReplicationInstanceClass: dms.t3.micro
    PubliclyAccessible: false
```

CT.DMS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
# Rule Identifier:
#   dms_replication_instance_not_public_check
#
# Description:
#   This control checks whether your AWS DMS replication instance is public.
#
# Reports on:
#   AWS::DMS::ReplicationInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS DMS replication instance
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS replication instance resource
```

```
#       And: 'PubliclyAccessible' is not present on the AWS DMS replication instance
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS replication instance resource
#       And: 'PubliclyAccessible' is present on the AWS DMS replication instance
#            and is set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS replication instance resource
#       And: 'PubliclyAccessible' is present on the AWS DMS replication instance
#            and is set to bool(false)
#      Then: PASS

#
# Constants
#
let DMS_REPLICATION_INSTANCE_TYPE = "AWS::DMS::ReplicationInstance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let dms_replication_instances = Resources.*[ Type == %DMS_REPLICATION_INSTANCE_TYPE ]

#
# Primary Rules
#
rule dms_replication_instance_not_public_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %dms_replication_instances not
 empty {
    check(%dms_replication_instances.Properties)
        <<
        [CT.DMS.PR.1]: Require that a public AWS DMS replication instance is not public
        [FIX]: Set 'PubliclyAccessible' to 'false'.
        >>
}

rule dms_replication_instance_not_public_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DMS_REPLICATION_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%DMS_REPLICATION_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.DMS.PR.1]: Require that a public AWS DMS replication instance is not public
```

```
        [FIX]: Set 'PubliclyAccessible' to 'false'.
        >>
}

#
# Parameterized Rules
#
rule check(dms_replication_instances) {
    %dms_replication_instances {
        # Scenario 2
        PubliclyAccessible exists
        # Scenario 3 and 4
        PubliclyAccessible == false
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DMS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DMSReplicationInstance:
    Type: AWS::DMS::ReplicationInstance
    Properties:
```

```
      ReplicationInstanceClass: dms.t3.micro
      PubliclyAccessible: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DMSReplicationInstance:
    Type: AWS::DMS::ReplicationInstance
    Properties:
      ReplicationInstanceClass: dms.t3.micro
      PubliclyAccessible: true
```

[CT.DMS.PR.2] Require an AWS Database Migration Service (DMS) Endpoint to encrypt connections for source and target endpoints

This control checks whether an AWS Database Migration Service (AWS DMS) Endpoint is configured to encrypt connections for source and target endpoints by using Secure Sockets Layer (SSL).

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DMS::Endpoint`
- **CloudFormation guard rule:** CT.DMS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DMS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DMS.PR.2 example templates

Explanation

You can encrypt connections for source and target endpoints by using Secure Sockets Layer (SSL). By enabling encryption in-transit with SSL, you can protect the confidentiality of data during AWS DMS data migrations.

Usage considerations

- This control applies only to AWS DMS endpoints with an EngineName property of `mysql`, `oracle`, `postgres`, `mariadb`, `aurora`, `aurora-postgresql`, `db2, sybase, mongodb, docdb, or sqlserver'`.
- Not all SSL modes work with all database endpoints. See Using SSL with AWS Database Migration Service in the *AWS Database Migration Service User Guide* for information on which SSL modes are supported for each database engine, and limitations of using SSL with AWS DMS.

Remediation for rule failure

Set the value of the SslMode property to a supported encryption mode for the endpoint engine (one of require, verify-ca, or verify-full).

The examples that follow show how to implement this remediation.

AWS DMS Endpoint - Example

An AWS DMS endpoint configured with a `postgres` database target and connection encryption using SSL (TLS). The example is shown in JSON and in YAML.

JSON example

```
{
    "Endpoint": {
        "Type": "AWS::DMS::Endpoint",
        "Properties": {
            "DatabaseName": "sample-db",
            "EndpointType": "target",
            "Username": {
                "Fn::Sub": "{{resolve:secretsmanager:${DMSEndpointSecret}::username}}"
            },
            "Password": {
                "Fn::Sub": "{{resolve:secretsmanager:${DMSEndpointSecret}::password}}"
```

```
            },
            "Port": 1234,
            "ServerName": "server.db.example.com",
            "EngineName": "postgres",
            "SslMode": "require"
        }
    }
}
```

YAML example

```
Endpoint:
  Type: AWS::DMS::Endpoint
  Properties:
    DatabaseName: sample-db
    EndpointType: target
    Username: !Sub '{{resolve:secretsmanager:${DMSEndpointSecret}::username}}'
    Password: !Sub '{{resolve:secretsmanager:${DMSEndpointSecret}::password}}'
    Port: 1234
    ServerName: server.db.example.com
    EngineName: postgres
    SslMode: require
```

CT.DMS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   dms_endpoint_ssl_configured_check
#
# Description:
#   This control checks whether an AWS Database Migration Service (AWS DMS) Endpoint is
 configured to encrypt connections for source and target endpoints by using
#   Secure Sockets Layer (SSL).
#
# Reports on:
```

```
#   AWS::DMS::Endpoint
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS DMS endpoint resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS endpoint resource
#       And: 'EngineName' has been set to an engine other than an engine that supports
 configuration
#            of SSL connections via 'SslMode' (values other than 'mysql', 'oracle',
 'postgres', 'mariadb',
#            'aurora', 'aurora-postgresql', 'db2, 'sybase', 'mongodb', 'docdb',
 'sqlserver')
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS endpoint resource
#       And: 'EngineName' has been set to an engine that supports configuration of SSL
 connections via 'SslMode'
#            ('mysql', 'oracle', 'postgres', 'mariadb', 'aurora', 'aurora-postgresql',
#            'db2, 'sybase', 'mongodb', 'docdb', 'sqlserver')
#       And: 'SslMode' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a AWS DMS endpoint resource
#       And: 'EngineName' has been set to an engine that supports configuration of SSL
 connections via 'SslMode'
#            ('mysql', 'oracle', 'postgres', 'mariadb', 'aurora', 'aurora-postgresql',
#            'db2, 'sybase', 'mongodb', 'docdb', 'sqlserver')
#       And: 'SslMode' has been provided and set to a value other than 'require',
 'verify-ca' or 'verify-full'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a AWS DMS endpoint resource
#       And: 'EngineName' has been set to an engine that supports configuration of SSL
 connections via 'SslMode'
#            ('mysql', 'oracle', 'postgres', 'mariadb', 'aurora', 'aurora-postgresql',
#            'db2, 'sybase', 'mongodb', 'docdb', 'sqlserver')
#       And: 'SslMode' has been provided and set to 'require', 'verify-ca' or 'verify-
full'
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let DMS_ENDPOINT_TYPE = "AWS::DMS::Endpoint"
let DMS_ENGINE_NAMES_WITH_SSL_SUPPORT = [
    "mysql",
    "oracle",
    "postgres",
    "mariadb",
    "aurora",
    "aurora-postgresql",
    "db2",
    "sybase",
    "mongodb",
    "docdb",
    "sqlserver"
]
let ALLOWED_DMS_SSL_MODES = [
    "require",
    "verify-ca",
    "verify-full"
]

#
# Assignments
#
let dms_endpoints = Resources.*[ Type == %DMS_ENDPOINT_TYPE ]

#
# Primary Rules
#
rule dms_endpoint_ssl_configured_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %dms_endpoints not empty {
    check(%dms_endpoints.Properties)
```

```
        <<
        [CT.DMS.PR.2]: Require an AWS Database Migration Service (AWS DMS) Endpoint to
 encrypt connections for source and target endpoints
        [FIX]: Set the value of the SslMode property to a supported encryption mode for
 the endpoint engine (one of require, verify-ca, or verify-full).
        >>
}

rule dms_endpoint_ssl_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DMS_ENDPOINT_TYPE) {
    check(%INPUT_DOCUMENT.%DMS_ENDPOINT_TYPE.resourceProperties)
        <<
        [CT.DMS.PR.2]: Require an AWS Database Migration Service (AWS DMS) Endpoint to
 encrypt connections for source and target endpoints
        [FIX]: Set the value of the SslMode property to a supported encryption mode for
 the endpoint engine (one of require, verify-ca, or verify-full).
        >>
}

#
# Parameterized Rules
#
rule check(dms_endpoint) {
    %dms_endpoint [
        # Scenario 2
        EngineName exists
        EngineName in %DMS_ENGINE_NAMES_WITH_SSL_SUPPORT
    ] {
        # Scenario 3
        SslMode exists
        # Scenarios 4 and 5
        SslMode in %ALLOWED_DMS_SSL_MODES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DMS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DMSEndpointSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Example DMS endpoint secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\;+%{},'
  Endpoint:
    Type: AWS::DMS::Endpoint
    Properties:
      DatabaseName: example-db
      EndpointType: target
      Username:
        Fn::Sub: '{{resolve:secretsmanager:${DMSEndpointSecret}::username}}'
      Password:
        Fn::Sub: '{{resolve:secretsmanager:${DMSEndpointSecret}::password}}'
      Port: 1234
      ServerName: server.db.example.com
      EngineName: postgres
      SslMode: require
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DMSEndpointSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Example DMS endpoint secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\;+%{},'
  Endpoint:
    Type: AWS::DMS::Endpoint
    Properties:
      DatabaseName: example-db
      EndpointType: target
      Username:
        Fn::Sub: '{{resolve:secretsmanager:${DMSEndpointSecret}::username}}'
      Password:
        Fn::Sub: '{{resolve:secretsmanager:${DMSEndpointSecret}::password}}'
      Port: 1234
      ServerName: server.db.example.com
      EngineName: postgres
```

Amazon DocumentDB controls

Topics

- [CT.DOCUMENTDB.PR.1] Require an Amazon DocumentDB cluster to be encrypted at rest
- [CT.DOCUMENTDB.PR.2] Require an Amazon DocumentDB cluster to have a backup retention period greater than or equal to seven days

[CT.DOCUMENTDB.PR.1] Require an Amazon DocumentDB cluster to be encrypted at rest

This control checks whether storage encryption is enabled for an Amazon DocumentDB (with MongoDB compatibility) cluster.

- **Control objective:** Encrypt data at rest

- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DocDB::DBCluster`
- **CloudFormation guard rule:** CT.DOCUMENTDB.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DOCUMENTDB.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.DOCUMENTDB.PR.1 example templates

Explanation

You encrypt data at rest in your Amazon DocumentDB cluster by specifying the storage encryption option when you create your cluster. Storage encryption is enabled cluster-wide, and it is applied to all instances, including the primary instance and any replicas. It also is applied to your cluster's storage volume, data, indexes, logs, automated backups, and snapshots.

Clusters that you create using AWS CloudFormation have encryption at rest turned off by default. Therefore, you must explicitly enable encryption at rest using the `StorageEncrypted` property.

Remediation for rule failure

Set the value of the `StorageEncrypted` parameter to true.

The examples that follow show how to implement this remediation.

Amazon DocumentDB Cluster - Example

An Amazon DocumentDB cluster configured with storage encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "DocumentDBCluster": {
        "Type": "AWS::DocDB::DBCluster",
        "Properties": {
```

```
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:
${DocumentDBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:
${DocumentDBClusterSecret}::password}}"
            },
            "StorageEncrypted": true
        }
    }
}
```

YAML example

```
DocumentDBCluster:
  Type: AWS::DocDB::DBCluster
  Properties:
    MasterUsername: !Sub '{{resolve:secretsmanager:
${DocumentDBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${DocumentDBClusterSecret}::password}}'
    StorageEncrypted: true
```

CT.DOCUMENTDB.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   docdb_cluster_encrypted_check
#
# Description:
#   This control checks whether storage encryption is enabled for an Amazon DocumentDB
 (with MongoDB compatibility) cluster.
#
# Reports on:
```

```
#   AWS::DocDB::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Document DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Document DB cluster resource
#       And: 'StorageEncrypted' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Document DB cluster resource
#       And: 'StorageEncrypted' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Document DB cluster resource
#       And: 'StorageEncrypted' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let DOCUMENT_DB_CLUSTER_TYPE = "AWS::DocDB::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let document_db_clusters = Resources.*[ Type == %DOCUMENT_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
```

```
rule docdb_cluster_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %document_db_clusters not empty {
    check(%document_db_clusters.Properties)
        <<
        [CT.DOCUMENTDB.PR.1]: Require an Amazon DocumentDB cluster to be encrypted at
 rest
        [FIX]: Set the value of the 'StorageEncrypted' parameter to true.
        >>
}

rule docdb_cluster_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DOCUMENT_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%DOCUMENT_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.DOCUMENTDB.PR.1]: Require an Amazon DocumentDB cluster to be encrypted at
 rest
        [FIX]: Set the value of the 'StorageEncrypted' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(document_db_cluster) {
    %document_db_cluster {
        # Scenario 2
        StorageEncrypted exists
        # Scenarios 3 and 4
        StorageEncrypted == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.DOCUMENTDB.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DocumentDBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: \"@/\\
  DocumentDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::password}}'
      StorageEncrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DocumentDBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
```

```
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: \"@/\\
  DocumentDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::password}}'
      StorageEncrypted: false
```

[CT.DOCUMENTDB.PR.2] Require an Amazon DocumentDB cluster to have a backup retention period greater than or equal to seven days

This control checks whether an Amazon DocumentDB cluster retention period is set to seven or more days (>=7). The default retention period is one day.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DocDB::DBCluster`
- **CloudFormation guard rule:** CT.DOCUMENTDB.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DOCUMENTDB.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.DOCUMENTDB.PR.2 example templates

Explanation

Amazon DocumentDB creates daily automatic snapshots of your cluster during your cluster's backup window. Amazon DocumentDB saves the automatic snapshots of your cluster according to the backup retention period that you specify, allowing you to restore to any point within the

backup retention period. This daily snapshot strengthens the resilience of your systems, and it can help you recover quickly from a security incident.

Remediation for rule failure

Set the value of the `BackupRetentionPeriod` parameter to an integer value between 7 and 35 days (inclusive).

The examples that follow show how to implement this remediation.

Amazon DocumentDB Cluster - Example

An Amazon DocumentDB cluster configured with a backup retention period of seven (7) days. The example is shown in JSON and in YAML.

JSON example

```
{
    "DocumentDBCluster": {
        "Type": "AWS::DocDB::DBCluster",
        "Properties": {
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:
${DocumentDBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:
${DocumentDBClusterSecret}::password}}"
            },
            "BackupRetentionPeriod": 7
        }
    }
}
```

YAML example

```
DocumentDBCluster:
  Type: AWS::DocDB::DBCluster
  Properties:
```

```
    MasterUsername: !Sub '{{resolve:secretsmanager:
${DocumentDBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${DocumentDBClusterSecret}::password}}'
    BackupRetentionPeriod: 7
```

CT.DOCUMENTDB.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   docdb_cluster_backup_retention_check
#
# Description:
#   This control checks whether an Amazon DocumentDB cluster retention period is set to
 seven or more days (>=7).
#
# Reports on:
#   AWS::DocDB::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Document DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Document DB cluster resource
#       And: 'BackupRetentionPeriod' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a Document DB cluster resource
#       And: 'BackupRetentionPeriod' has been provided and set to an integer value less
 than seven (<7)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Document DB cluster resource
#       And: 'BackupRetentionPeriod' has been provided and set to an integer value
 greater than or equal to seven (>=7)
#      Then: PASS

#
# Constants
#
let DOCUMENT_DB_CLUSTER_TYPE = "AWS::DocDB::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let document_db_clusters = Resources.*[ Type == %DOCUMENT_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule docdb_cluster_backup_retention_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %document_db_clusters not empty {
    check(%document_db_clusters.Properties)
        <<
        [CT.DOCUMENTDB.PR.2]: Require an Amazon DocumentDB cluster to have automatic
 backups enabled
        [FIX]: Set the value of the 'BackupRetentionPeriod' parameter to an integer
 value between 7 and 35 days (inclusive).
        >>
}

rule docdb_cluster_backup_retention_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DOCUMENT_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%DOCUMENT_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.DOCUMENTDB.PR.2]: Require an Amazon DocumentDB cluster to have automatic
 backups enabled
        [FIX]: Set the value of the 'BackupRetentionPeriod' parameter to an integer
 value between 7 and 35 days (inclusive).
```

```
        >>
}

#
# Parameterized Rules
#
rule check(document_db_cluster) {
    %document_db_cluster {
        # Scenario 2
        BackupRetentionPeriod exists
        # Scenarios 3 and 4
        BackupRetentionPeriod >= 7
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DOCUMENTDB.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DocumentDBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
```

```
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '\"@/\\'
  DocumentDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::password}}'
      BackupRetentionPeriod: 7
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DocumentDBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '\"@/\\'
  DocumentDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DocumentDBClusterSecret}::password}}'
      BackupRetentionPeriod: 1
```

Amazon DynamoDB controls

Topics

- [CT.DYNAMODB.PR.1] Require that point-in-time recovery for an Amazon DynamoDB table is activated
- [CT.DYNAMODB.PR.2] Require an Amazon DynamoDB table to be encrypted at rest using an AWS KMS key

[CT.DYNAMODB.PR.1] Require that point-in-time recovery for an Amazon DynamoDB table is activated

This control checks whether point-in-time recovery (PITR) is enabled for an Amazon DynamoDB table.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DynamoDB::Table`
- **CloudFormation guard rule:** CT.DYNAMODB.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DYNAMODB.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DYNAMODB.PR.1 example templates

Explanation

Backups help you to recover more quickly from a security incident. They also strengthen the resilience of your systems. Amazon DynamoDB point-in-time recovery (PITR) automates backups for DynamoDB tables, which can reduce the time required to recover from accidental delete or write operations. DynamoDB tables that have PITR enabled can be restored to any point in time within the last 35 days.

Remediation for rule failure

Provide a `PointInTimeRecoverySpecification` configuration and set `PointInTimeRecoveryEnabled` to `true`.

The examples that follow show how to implement this remediation.

Amazon DynamoDB Table - Example

Amazon DynamoDB table configured with point-in-time recovery activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "DynamoDBTable": {
        "Type": "AWS::DynamoDB::Table",
        "Properties": {
            "AttributeDefinitions": [
                {
                    "AttributeName": "PK",
                    "AttributeType": "S"
                }
            ],
            "BillingMode": "PAY_PER_REQUEST",
            "KeySchema": [
                {
                    "AttributeName": "PK",
                    "KeyType": "HASH"
                }
            ],
            "PointInTimeRecoverySpecification": {
                "PointInTimeRecoveryEnabled": true
            }
        }
    }
}
```

YAML example

```
DynamoDBTable:
  Type: AWS::DynamoDB::Table
  Properties:
    AttributeDefinitions:
      - AttributeName: PK
```

```
        AttributeType: S
    BillingMode: PAY_PER_REQUEST
    KeySchema:
      - AttributeName: PK
        KeyType: HASH
    PointInTimeRecoverySpecification:
      PointInTimeRecoveryEnabled: true
```

CT.DYNAMODB.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
# Rule Identifier:
#   dynamodb_table_pitr_enabled_check
#
# Description:
#   This control checks whether point-in-time recovery (PITR) is enabled for an Amazon
 DynamoDB table.
#
# Reports on:
#   AWS::DynamoDB::Table
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Paramaeters:
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any DynamoDB table resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contains a DynamoDB table resource
#        And: 'PointInTimeRecoverySpecification' is not present on the DynamoDB table
 resource
#       Then: FAIL
```

```
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contains a DynamoDB table resource
#        And: 'PointInTimeRecoverySpecification' is present on the DynamoDB table
 resource
#        And: 'PointInTimeRecoveryEnabled' in 'PointInTimeRecoverySpecification' is
 missing or is a value
#              other than bool(true)
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contains a DynamoDB table resource
#        And: 'PointInTimeRecoverySpecification' is present on the DynamoDB table
 resource
#        And: 'PointInTimeRecoveryEnabled' in 'PointInTimeRecoverySpecification' is
 present and set to bool(true)
#       Then: PASS

#
# Constants
#
let DYNAMODB_TABLE_TYPE = "AWS::DynamoDB::Table"
let INPUT_DOCUMENT = this

#
# Assignments
#
let dynamodb_tables = Resources.*[ Type == %DYNAMODB_TABLE_TYPE ]

#
# Primary Rules
#
rule dynamodb_table_pitr_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %dynamodb_tables not empty {
    check(%dynamodb_tables.Properties)
        <<
        [CT.DYNAMODB.PR.1]: Require that point-in-time recovery for an Amazon DynamoDB
 table is activated
            [FIX]: Provide a 'PointInTimeRecoverySpecification' configuration and set
 'PointInTimeRecoveryEnabled' to 'true'.
        >>
}
```

```
rule dynamodb_table_pitr_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DYNAMODB_TABLE_TYPE) {
    check(%INPUT_DOCUMENT.%DYNAMODB_TABLE_TYPE.resourceProperties)
        <<
        [CT.DYNAMODB.PR.1]: Require that point-in-time recovery for an Amazon DynamoDB
 table is activated
            [FIX]: Provide a 'PointInTimeRecoverySpecification' configuration and set
 'PointInTimeRecoveryEnabled' to 'true'.
        >>
}

rule check(dynamodb_table) {
    %dynamodb_table {
        # Scenario 2
        PointInTimeRecoverySpecification exists
        PointInTimeRecoverySpecification is_struct

        # Scenario 3 and 4
        PointInTimeRecoverySpecification {
            PointInTimeRecoveryEnabled exists
            PointInTimeRecoveryEnabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, DYNAMODB_TABLE_TYPE) {
    %doc.%DYNAMODB_TABLE_TYPE.resourceProperties exists
}
```

CT.DYNAMODB.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: "PK"
          AttributeType: "S"
      BillingMode: "PAY_PER_REQUEST"
      KeySchema:
        - AttributeName: "PK"
          KeyType: "HASH"
      PointInTimeRecoverySpecification:
        PointInTimeRecoveryEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: "PK"
          AttributeType: "S"
      BillingMode: "PAY_PER_REQUEST"
      KeySchema:
        - AttributeName: "PK"
          KeyType: "HASH"
      PointInTimeRecoverySpecification:
        PointInTimeRecoveryEnabled: false
```

[CT.DYNAMODB.PR.2] Require an Amazon DynamoDB table to be encrypted at rest using an AWS KMS key

This control checks whether your Amazon DynamoDB table is encrypted with an AWS Key Management Service (KMS) key.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DynamoDB::Table`
- **CloudFormation guard rule:** CT.DYNAMODB.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DYNAMODB.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.DYNAMODB.PR.2 example templates

Explanation

Amazon DynamoDB encryption at rest provides an additional layer of data protection, because it always secures your data in an encrypted table - including its primary key, local and global secondary indexes, streams, global tables, backups, and DynamoDB Accelerator (DAX) clusters, whenever the data is stored in durable media.

Encryption at rest integrates with AWS KMS for managing the encryption keys that are used to encrypt your tables.

Usage considerations

- This control requires only that KMS keys are used for server-side encryption. It does not check the properties of the KMS key used, such as whether the KMS key is customer-managed or service-managed.

Remediation for rule failure

Provide a `SSESpecification` configuration and set `SSEEnabled` to `true`.

The examples that follow show how to implement this remediation.

Amazon DynamoDB Table - Example

An Amazon DynamoDB table configured to encrypt data at rest with AWS Key Management Service (KMS) keys. The example is shown in JSON and in YAML.

JSON example

```
{
    "DynamoDBTable": {
        "Type": "AWS::DynamoDB::Table",
        "Properties": {
            "AttributeDefinitions": [
                {
                    "AttributeName": "PK",
                    "AttributeType": "S"
                }
            ],
            "BillingMode": "PAY_PER_REQUEST",
            "KeySchema": [
                {
                    "AttributeName": "PK",
                    "KeyType": "HASH"
                }
            ],
            "SSESpecification": {
                "SSEEnabled": true
            }
        }
    }
}
```

YAML example

```
DynamoDBTable:
```

```
  Type: AWS::DynamoDB::Table
  Properties:
    AttributeDefinitions:
      - AttributeName: PK
        AttributeType: S
    BillingMode: PAY_PER_REQUEST
    KeySchema:
      - AttributeName: PK
        KeyType: HASH
    SSESpecification:
      SSEEnabled: true
```

CT.DYNAMODB.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   dynamodb_table_encrypted_kms_check
#
# Description:
#   This control checks whether your Amazon DynamoDB table is encrypted with an AWS Key
 Management Service (KMS) key.
#
# Reports on:
#   AWS::DynamoDB::Table
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any DynamoDB table resources
#      Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DynamoDB table resources
#       And: 'SSEEnabled' in 'SSESpecification' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DynamoDB table resources
#       And: 'SSEEnabled' in 'SSESpecification' has been provided and set to a value
 other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DynamoDB table resources
#       And: 'SSEEnabled' in 'SSESpecification' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let DYNAMODB_TABLE_TYPE = "AWS::DynamoDB::Table"
let INPUT_DOCUMENT = this

#
# Assignments
#
let dynamodb_tables = Resources.*[ Type == %DYNAMODB_TABLE_TYPE ]

#
# Primary Rules
#
rule dynamodb_table_encrypted_kms_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %dynamodb_tables not empty {
    check(%dynamodb_tables.Properties)
        <<
        [CT.DYNAMODB.PR.2]: Require an Amazon DynamoDB table to be encrypted at rest
 using an AWS KMS key
        [FIX]: Provide a 'SSESpecification' configuration and set 'SSEEnabled' to
 'true'.
        >>
}

rule dynamodb_table_encrypted_kms_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DYNAMODB_TABLE_TYPE) {
    check(%INPUT_DOCUMENT.%DYNAMODB_TABLE_TYPE.resourceProperties)
```

```
        <<
        [CT.DYNAMODB.PR.2]: Require an Amazon DynamoDB table to be encrypted at rest
 using an AWS KMS key
        [FIX]: Provide a 'SSESpecification' configuration and set 'SSEEnabled' to
 'true'.
        >>
}

rule check(dynamodb_table) {
    %dynamodb_table {
        # Scenario 2
        SSESpecification exists
        SSESpecification is_struct

        # Scenarios 3 and 4
        SSESpecification {
            SSEEnabled exists
            SSEEnabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, DYNAMODB_TABLE_TYPE) {
    %doc.%DYNAMODB_TABLE_TYPE.resourceProperties exists
}
```

CT.DYNAMODB.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
      - AttributeName: PK
        AttributeType: S
      BillingMode: PAY_PER_REQUEST
      KeySchema:
      - AttributeName: PK
        KeyType: HASH
      SSESpecification:
        SSEEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
      - AttributeName: PK
        AttributeType: S
      BillingMode: PAY_PER_REQUEST
      KeySchema:
      - AttributeName: PK
        KeyType: HASH
      SSESpecification:
        SSEEnabled: false
```

DynamoDB Accelerator controls

Topics

- [CT.DAX.PR.1] Require encryption at rest for all Amazon DynamoDB Accelerator (DAX) clusters

- [CT.DAX.PR.2] Require an Amazon DAX cluster to deploy nodes to at least three Availability Zones
- [CT.DAX.PR.3] Require an Amazon DAX cluster to encrypt data in transit with Transport Layer Security (TLS)

[CT.DAX.PR.1] Require encryption at rest for all Amazon DynamoDB Accelerator (DAX) clusters

This control checks whether Amazon DynamoDB Accelerator (DAX) clusters are encrypted at rest.

> **Note**
>
> The control CT.DAX.PR.1 cannot be activated from home Regions Canada (Central) Region, Europe (Stockholm) Region, and Asia Pacific (Seoul) Region, because the `AWS::DAX::Cluster` resource type is not available in those Regions. If your home Region is not one of these three, you can activate the control for these three Regions from another home Region, if these three Regions are governed by AWS Control Tower in your landing zone. For example, if your home Region is US West (Oregon) Region, you can deploy the control to Canada (Central) Region, if Canada (Central) Region is governed by AWS Control Tower.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DAX::Cluster`
- **CloudFormation guard rule:** CT.DAX.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DAX.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DAX.PR.1 example templates

Explanation

Encrypting data at rest reduces the risk that data stored on disk may be accessible to a user who is not authenticated to AWS. Encryption adds another set of access controls, which limits the ability of unauthorized users to gain access to the data. For example, API permissions must decrypt the data before it can be read.

Remediation for rule failure

Provide an `SSESpecification` configuration with `SSEEnabled` set to `true`.

The examples that follow show how to implement this remediation.

Amazon DAX Cluster - Example

Amazon DAX cluster configured with server-side encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "DAXCluster": {
        "Type": "AWS::DAX::Cluster",
        "Properties": {
            "IAMRoleARN": {
                "Fn::GetAtt": [
                    "DAXServiceRole",
                    "Arn"
                ]
            },
            "NodeType": "dax.t3.small",
            "ReplicationFactor": 1,
            "SSESpecification": {
                "SSEEnabled": true
            }
        }
    }
}
```

YAML example

```
DAXCluster:
  Type: AWS::DAX::Cluster
  Properties:
    IAMRoleARN: !GetAtt 'DAXServiceRole.Arn'
    NodeType: dax.t3.small
    ReplicationFactor: 1
    SSESpecification:
      SSEEnabled: true
```

CT.DAX.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   dax_cluster_encryption_enabled_check
#
# Description:
#   This control checks whether Amazon DynamoDB Accelerator (DAX) clusters are
 encrypted at rest.
#
# Reports on:
#   AWS::DAX::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any DAX Cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains at least one DAX Cluster resource
```

```
#       And: 'SSESpecification' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains at least one DAX Cluster resource
#       And: 'SSESpecification' has been provided and 'SSESpecification.SSEEnabled' is
 missing or has been set to a
#            value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains at least one DAX Cluster resource
#       And: 'SSESpecification' has been provided and 'SSESpecification.SSEEnabled' is
 present and has been set to
#            bool(true)
#      Then: PASS

#
# Constants
#
let DAX_CLUSTER_TYPE = "AWS::DAX::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let dax_clusters = Resources.*[ Type == %DAX_CLUSTER_TYPE ]

#
# Primary Rules
#
rule dax_cluster_encryption_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %dax_clusters not empty {
    check(%dax_clusters.Properties)
        <<
        [CT.DAX.PR.1]: Require encryption at rest for all Amazon DynamoDB Accelerator
 (DAX) clusters
            [FIX]: Provide an 'SSESpecification' configuration with 'SSEEnabled' set to
 'true'.
        >>
}

rule dax_cluster_encryption_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DAX_CLUSTER_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%DAX_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.DAX.PR.1]: Require encryption at rest for all Amazon DynamoDB Accelerator
 (DAX) clusters
            [FIX]: Provide an 'SSESpecification' configuration with 'SSEEnabled' set to
 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(dax_cluster) {
    %dax_cluster {
        # Scenario 2
        SSESpecification exists
        SSESpecification is_struct

        # Scenario 3 and 4
        SSESpecification {
            SSEEnabled exists
            SSEEnabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DAX.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DAXServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DAXCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
```

```
        Fn::GetAtt: [DAXServiceRole, Arn]
      NodeType: dax.t3.small
      ReplicationFactor: 1
      SSESpecification:
        SSEEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DAXServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
```

```
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DAXCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
        Fn::GetAtt: [DAXServiceRole, Arn]
      NodeType: dax.t3.small
      ReplicationFactor: 1
      SSESpecification:
        SSEEnabled: false
```

[CT.DAX.PR.2] Require an Amazon DAX cluster to deploy nodes to at least three Availability Zones

This control checks whether an Amazon DAX cluster is configured to deploy cluster nodes to at least three Availability Zones.

- **Control objective:** Improve resiliency, Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DAX::Cluster`
- **CloudFormation guard rule:** CT.DAX.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DAX.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DAX.PR.2 example templates

Explanation

AWS Control Tower recommends that you deploy your Amazon DAX clusters in multiple Availability Zones. This deployment technique allows you to design and operate applications and databases that fail over between Availability Zones automatically, without interruption. For production usage,

we strongly recommend that you deploy DAX across at least three nodes, with each node placed into a different Availability Zone.

Remediation for rule failure

Set the `ReplicationFactor` parameter to an integer value greater than or equal to three (>= 3), and set the `AvailabilityZones` parameter to a list containing three unique Availability Zone entries.

The examples that follow show how to implement this remediation.

Amazon DAX cluster - Example

Amazon DAX cluster configured with three nodes and three distinct Availability Zones. The example is shown in JSON and in YAML.

JSON example

```
{
    "DaxCluster": {
        "Type": "AWS::DAX::Cluster",
        "Properties": {
            "IAMRoleARN": {
                "Fn::GetAtt": [
                    "DaxDynamoAccessRole",
                    "Arn"
                ]
            },
            "NodeType": "dax.t3.small",
            "ReplicationFactor": 3,
            "AvailabilityZones": [
                {
                    "Fn::Select": [
                        0,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                {
                    "Fn::Select": [
                        1,
                        {
```

```
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                {
                    "Fn::Select": [
                        2,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
DaxCluster:
  Type: AWS::DAX::Cluster
  Properties:
    IAMRoleARN: !GetAtt 'DaxDynamoAccessRole.Arn'
    NodeType: dax.t3.small
    ReplicationFactor: 3
    AvailabilityZones:
      - !Select
        - 0
        - !GetAZs ''
      - !Select
        - 1
        - !GetAZs ''
      - !Select
        - 2
        - !GetAZs ''
```

CT.DAX.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   dax_cluster_multi_az_check
#
# Description:
#   This control checks whether an Amazon DAX cluster is configured to deploy cluster
 nodes to at least three Availability Zones.
#
# Reports on:
#   AWS::DAX::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any DAX cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DAX cluster resource
#       And: 'ReplicationFactor' has not been provided or has been provided as an
 integer
#            value less than three (< 3)
#       And: 'AvailabilityZones' has not been provided or provided as an empty list or
#            list with less than three unique entires
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DAX cluster resource
#       And: 'ReplicationFactor' has been provided as an integer value greater than or
#            equal to three (>= 3)
#       And: 'AvailabilityZones' has not been provided or provided as an empty list or
#            list with less than three unique entires
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a DAX cluster resource
#       And: 'ReplicationFactor' has not been provided or has been provided as an
 integer
#            value less than three (< 3)
#       And: 'AvailabilityZones' has been provided as a list with three or more unique
 entires
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DAX cluster resource
#       And: 'ReplicationFactor' has been provided as an integer value greater than or
#            equal to three (>= 3)
#       And: 'AvailabilityZones' has been provided as a list with three or more unique
 entires
#      Then: PASS

#
# Constants
#
let DAX_CLUSTER_TYPE = "AWS::DAX::Cluster"
let MINIMUM_NODE_COUNT = 3
let INPUT_DOCUMENT = this

#
# Assignments
#
let dax_clusters = Resources.*[ Type == %DAX_CLUSTER_TYPE ]

#
# Primary Rules
#
rule dax_cluster_multi_az_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %dax_clusters not empty {
    check(%dax_clusters.Properties)
        <<
        [CT.DAX.PR.2]: Require an Amazon DAX cluster to deploy nodes to at least three
 Availability Zones
        [FIX]: Set the 'ReplicationFactor' parameter to an integer value greater
 than or equal to three (>= 3), and set the 'AvailabilityZones' parameter to a list
 containing three unique Availability Zone entries.
        >>
}

rule dax_cluster_multi_az_check when is_cfn_hook(%INPUT_DOCUMENT, %DAX_CLUSTER_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%DAX_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.DAX.PR.2]: Require an Amazon DAX cluster to deploy nodes to at least three
 Availability Zones
        [FIX]: Set the 'ReplicationFactor' parameter to an integer value greater
 than or equal to three (>= 3), and set the 'AvailabilityZones' parameter to a list
 containing three unique Availability Zone entries.
        >>
}

#
# Parameterized Rules
#
rule check(dax_cluster) {
    %dax_cluster {
        # Scenario 2
        ReplicationFactor exists

        AvailabilityZones exists
        AvailabilityZones is_list
        AvailabilityZones not empty

        # Scenarios 3, 4 and 5
        ReplicationFactor >= %MINIMUM_NODE_COUNT

        AvailabilityZones[0] exists
        AvailabilityZones[1] exists
        AvailabilityZones[2] exists

        let az_one = AvailabilityZones[0]
        let az_two = AvailabilityZones[1]
        let az_three = AvailabilityZones[2]

        check_az_is_unique(%az_one, %az_two, %az_three)
        check_az_is_unique(%az_two, %az_one, %az_three)
    }
}

rule check_az_is_unique(az, first_az, second_az) {
    %az not in %first_az
    %az not in %second_az
}

#
```

```
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DAX.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DaxDynamoAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
```

```
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DaxCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
        Fn::GetAtt:
        - DaxDynamoAccessRole
        - Arn
      NodeType: dax.t3.small
      ReplicationFactor: 3
      AvailabilityZones:
      - Fn::Select:
        - 0
        - Fn::GetAZs: ''
      - Fn::Select:
        - 1
        - Fn::GetAZs: ''
      - Fn::Select:
        - 2
        - Fn::GetAZs: ''
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DaxDynamoAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DaxCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
        Fn::GetAtt:
        - DaxDynamoAccessRole
        - Arn
      NodeType: dax.t3.small
      ReplicationFactor: 3
```

[CT.DAX.PR.3] Require an Amazon DAX cluster to encrypt data in transit with Transport Layer Security (TLS)

This control checks whether an Amazon DynamoDB Accelerator (DAX) cluster endpoint is configured to encrypt data in transit with Transport Layer Security (TLS).

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::DAX::Cluster`
- **CloudFormation guard rule:** CT.DAX.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.DAX.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.DAX.PR.3 example templates

Explanation

Amazon DynamoDB Accelerator (DAX) supports encryption in transit of data between your application and your DAX cluster, so that you can use DAX in applications with stringent encryption requirements. DAX encryption in transit ensures that all requests and responses between the application and the cluster are encrypted by transport level security (TLS), and that connections to the cluster can be authenticated by verification of a cluster x509 certificate.

Usage considerations

- To enable encryption in transit between your application and DAX cluster, be sure to use a recent version of any of the DAX clients that support TLS in your application.

  Encryption in transit cannot be enabled on an existing DAX cluster. To use encryption in transit in an existing DAX application, create a new cluster with encryption in transit enabled, shift your application's traffic to it, then delete the old cluster.

Remediation for rule failure

Set the value of the ClusterEndpointEncryptionType property to TLS.

The examples that follow show how to implement this remediation.

Amazon DAX Cluster - Example

An Amazon DAX cluster configured to encrypt data in transit. The example is shown in JSON and in YAML.

JSON example

```
{
    "DaxCluster": {
        "Type": "AWS::DAX::Cluster",
        "Properties": {
            "IAMRoleARN": {
                "Fn::GetAtt": [
                    "DaxDynamoAccessRole",
                    "Arn"
                ]
            },
            "NodeType": "dax.t3.small",
            "ReplicationFactor": 3,
            "ClusterEndpointEncryptionType": "TLS"
        }
    }
}
```

YAML example

```
DaxCluster:
  Type: AWS::DAX::Cluster
  Properties:
    IAMRoleARN: !GetAtt 'DaxDynamoAccessRole.Arn'
    NodeType: dax.t3.small
    ReplicationFactor: 3
    ClusterEndpointEncryptionType: TLS
```

CT.DAX.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   dax_tls_endpoint_encryption_check
#
# Description:
#   This control checks whether an Amazon DAX cluster endpoint is configured to encrypt
 data in transit with Transport Layer Security (TLS).
#
# Reports on:
#   AWS::DAX::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any DAX cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DAX cluster resource
#       And: 'ClusterEndpointEncryptionType' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a DAX cluster resource
#       And: 'ClusterEndpointEncryptionType' has been provided and set to a value other
 than 'TLS'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a DAX cluster resource
#       And: 'ClusterEndpointEncryptionType' has been provided and set to 'TLS'
#      Then: PASS

#
# Constants
#
let DAX_CLUSTER_TYPE = "AWS::DAX::Cluster"
let ALLOWED_CLUSTER_ENCRYPTION_TYPES = [ "TLS" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let dax_clusters = Resources.*[ Type == %DAX_CLUSTER_TYPE ]

#
# Primary Rules
#
rule dax_tls_endpoint_encryption_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %dax_clusters not empty {
    check(%dax_clusters.Properties)
        <<
        [CT.DAX.PR.3]: Require an Amazon DAX cluster to encrypt data in transit with
 Transport Layer Security (TLS)
        [FIX]: Set the value of the ClusterEndpointEncryptionType property to TLS.
        >>
}

rule dax_tls_endpoint_encryption_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DAX_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%DAX_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.DAX.PR.3]: Require an Amazon DAX cluster to encrypt data in transit with
 Transport Layer Security (TLS)
        [FIX]: Set the value of the ClusterEndpointEncryptionType property to TLS.
        >>
}

#
# Parameterized Rules
#
rule check(dax_cluster) {
    %dax_cluster {
```

```
        # Scenario 2
        ClusterEndpointEncryptionType exists

        # Scenarios 3 and 4
        ClusterEndpointEncryptionType in %ALLOWED_CLUSTER_ENCRYPTION_TYPES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.DAX.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DaxDynamoAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
```

```
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DaxCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
        Fn::GetAtt:
        - DaxDynamoAccessRole
        - Arn
      NodeType: dax.t3.small
      ReplicationFactor: 3
      ClusterEndpointEncryptionType: TLS
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DaxDynamoAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: dax.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: DynamoAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - dynamodb:DescribeTable
            - dynamodb:PutItem
            - dynamodb:GetItem
            - dynamodb:UpdateItem
            - dynamodb:DeleteItem
            - dynamodb:Query
            - dynamodb:Scan
            - dynamodb:BatchGetItem
            - dynamodb:BatchWriteItem
            - dynamodb:ConditionCheckItem
            Resource:
              Fn::Sub: arn:${AWS::Partition}:dynamodb:${AWS::Region}:
${AWS::AccountId}:*
  DaxCluster:
    Type: AWS::DAX::Cluster
    Properties:
      IAMRoleARN:
        Fn::GetAtt:
        - DaxDynamoAccessRole
        - Arn
      NodeType: dax.t3.small
      ReplicationFactor: 3
      ClusterEndpointEncryptionType: NONE
```

AWS Elastic Beanstalk controls

Topics

- [CT.ELASTICBEANSTALK.PR.1] Require AWS Elastic Beanstalk environments to have enhanced health reporting enabled
- [CT.ELASTICBEANSTALK.PR.2] Require an AWS Elastic Beanstalk environment to have managed platform updates configured
- [CT.ELASTICBEANSTALK.PR.3] Require an AWS Elastic Beanstalk environment to have a logging configuration

[CT.ELASTICBEANSTALK.PR.1] Require AWS Elastic Beanstalk environments to have enhanced health reporting enabled

This control checks whether AWS Elastic Beanstalk environments and configuration templates are configured for enhanced health reporting.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticBeanstalk::Environment`, `AWS::ElasticBeanstalk::ConfigurationTemplate`
- **CloudFormation guard rule:** CT.ELASTICBEANSTALK.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICBEANSTALK.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICBEANSTALK.PR.1 example templates

Explanation

Elastic Beanstalk enhanced health reporting enables a more rapid response to changes in the health of the underlying infrastructure. These changes could result in a lack of availability of the application.

Elastic Beanstalk enhanced health reporting provides a status descriptor to gauge the severity of the identified issues and identify possible causes to investigate. The Elastic Beanstalk health agent,

included in supported Amazon Machine Images (AMIs), evaluates logs and metrics of environment EC2 instances.

Remediation for rule failure

For AWS Elastic Beanstalk environments, configure an `OptionSetting` with `Namespace` set to `aws:elasticbeanstalk:healthreporting:system`, `OptionName` set to `SystemType`, and `Value` set to `enhanced`. For AWS Elastic Beanstalk configuration templates, configure an `OptionSetting` with `Namespace` set to `aws:elasticbeanstalk:healthreporting:system`, `OptionName` set to `SystemType`, and `Value` set to `enhanced`. Omit this setting to adopt the default value of `enhanced`.

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Environment - Example

AWS Elastic Beanstalk environment configured with enhanced health reporting enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkEnvironment": {
        "Type": "AWS::ElasticBeanstalk::Environment",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:elasticbeanstalk:healthreporting:system",
                    "OptionName": "SystemType",
                    "Value": "enhanced"
                },
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                }
```

```
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkEnvironment:
  Type: AWS::ElasticBeanstalk::Environment
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    OptionSettings:
      - Namespace: aws:elasticbeanstalk:healthreporting:system
        OptionName: SystemType
        Value: enhanced
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
```

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Configuration Template - Example One

AWS Elastic Beanstalk configuration template configured with enhanced health reporting, enabled by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkConfigurationTemplate": {
        "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "OptionSettings": [
```

```
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
```

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Configuration Template - Example Two

AWS Elastic Beanstalk configuration template configured with enhanced health reporting, enabled by means of an entry in the `OptionSettings` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkConfigurationTemplate": {
        "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
        "Properties": {
```

```
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:elasticbeanstalk:healthreporting:system",
                    "OptionName": "SystemType",
                    "Value": "enhanced"
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    OptionSettings:
      - Namespace: aws:elasticbeanstalk:healthreporting:system
        OptionName: SystemType
        Value: enhanced
```

CT.ELASTICBEANSTALK.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elastic_beanstalk_enhanced_health_reporting_enabled_check
#
# Description:
```

```
#   This control checks whether AWS Elastic Beanstalk environments and configuration
 templates are configured for 'enhanced' health reporting.
#
# Reports on:
#   AWS::ElasticBeanstalk::Environment, AWS::ElasticBeanstalk::ConfigurationTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Beanstalk environment
 resources or
#            Elastic Beanstalk configuration template resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk environment resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk environment resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:healthreporting:system'
#            and an 'OptionName' property with value of 'SystemType'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk environment resource or
 an Elastic Beanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: An entry in the 'OptionSettings' list has a 'Namespace' property with a
 value of
#            'aws:elasticbeanstalk:healthreporting:system'
#       And: That same entry has an 'OptionName' property with a value of 'SystemType'
```

```
#       And: That same entry has a 'Value' property with a value of anything other than
 'enhanced', or the 'Value'
#            property is not provided.
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk configuration template
 resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk configuration template
 resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:healthreporting:system'
#            and an 'OptionName' property with value of 'SystemType'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Beanstalk environment resource or
 an Elastic Beanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: Every entry in the 'OptionSettings' list that has both a 'Namespace'
 property with a value of
#            'aws:elasticbeanstalk:healthreporting:system'
#            and an 'OptionName' property with a value of 'SystemType' also has a
 'Value' property with a value of
#            'enhanced'
#      Then: PASS

#
# Constants
#
let ELASTIC_BEANSTALK_ENVIRONMENT_TYPE = "AWS::ElasticBeanstalk::Environment"
let ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE =
 "AWS::ElasticBeanstalk::ConfigurationTemplate"
let ELASTIC_BEANSTALK_ENHANCED_HEALTH_REPORTING_NAMESPACE =
 "aws:elasticbeanstalk:healthreporting:system"
let ELASTIC_BEANSTALK_SYSTEM_TYPE_OPTION_NAME = "SystemType"
```

```
let ELASTIC_BEANSTALK_ENHANCED_VALUE = "enhanced"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_beanstalk_environments = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE ]
let elastic_beanstalk_configuration_templates = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule elastic_beanstalk_enhanced_health_reporting_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elastic_beanstalk_environments not empty {
    check_elastic_beanstalk_environments(%elastic_beanstalk_environments.Properties)
        <<
        [CT.ELASTICBEANSTALK.PR.1]: Require AWS Elastic Beanstalk environments to have
 enhanced health reporting enabled
        [FIX]: For AWS Elastic Beanstalk environments, configure an 'OptionSetting'
 with 'Namespace' set to 'aws:elasticbeanstalk:healthreporting:system', 'OptionName'
 set to 'SystemType', and 'Value' set to 'enhanced'. For AWS Elastic Beanstalk
 configuration templates, configure an 'OptionSetting' with 'Namespace' set to
 'aws:elasticbeanstalk:healthreporting:system', 'OptionName' set to 'SystemType', and
 'Value' set to 'enhanced'. Omit this setting to adopt the default value of 'enhanced'.
        >>
}

rule elastic_beanstalk_enhanced_health_reporting_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elastic_beanstalk_configuration_templates not empty {

 check_elastic_beanstalk_configuration_templates(%elastic_beanstalk_configuration_templates.Pro
        <<
        [CT.ELASTICBEANSTALK.PR.1]: Require AWS Elastic Beanstalk environments to have
 enhanced health reporting enabled
        [FIX]: For AWS Elastic Beanstalk environments, configure an 'OptionSetting'
 with 'Namespace' set to 'aws:elasticbeanstalk:healthreporting:system', 'OptionName'
 set to 'SystemType', and 'Value' set to 'enhanced'. For AWS Elastic Beanstalk
 configuration templates, configure an 'OptionSetting' with 'Namespace' set to
```

```
 'aws:elasticbeanstalk:healthreporting:system', 'OptionName' set to 'SystemType', and
 'Value' set to 'enhanced'. Omit this setting to adopt the default value of 'enhanced'.
        >>
}

rule elastic_beanstalk_enhanced_health_reporting_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE) {
    check_elastic_beanstalk_environments(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_ENVIRONMENT_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.1]: Require AWS Elastic Beanstalk environments to have
 enhanced health reporting enabled
        [FIX]: For AWS Elastic Beanstalk environments, configure an 'OptionSetting'
 with 'Namespace' set to 'aws:elasticbeanstalk:healthreporting:system', 'OptionName'
 set to 'SystemType', and 'Value' set to 'enhanced'. For AWS Elastic Beanstalk
 configuration templates, configure an 'OptionSetting' with 'Namespace' set to
 'aws:elasticbeanstalk:healthreporting:system', 'OptionName' set to 'SystemType', and
 'Value' set to 'enhanced'. Omit this setting to adopt the default value of 'enhanced'.
        >>
}

rule elastic_beanstalk_enhanced_health_reporting_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE) {
    check_elastic_beanstalk_configuration_templates(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.1]: Require AWS Elastic Beanstalk environments to have
 enhanced health reporting enabled
        [FIX]: For AWS Elastic Beanstalk environments, configure an 'OptionSetting'
 with 'Namespace' set to 'aws:elasticbeanstalk:healthreporting:system', 'OptionName'
 set to 'SystemType', and 'Value' set to 'enhanced'. For AWS Elastic Beanstalk
 configuration templates, configure an 'OptionSetting' with 'Namespace' set to
 'aws:elasticbeanstalk:healthreporting:system', 'OptionName' set to 'SystemType', and
 'Value' set to 'enhanced'. Omit this setting to adopt the default value of 'enhanced'.
        >>
}

#
# Parameterized Rules
#
rule check_elastic_beanstalk_environments(elastic_beanstalk_environments) {
    %elastic_beanstalk_environments {
        # Scenario 2
        check_option_settings_exists_or_is_non_empty_list(this)
```

```
        # Scenario 3, 4, 7
        check_option_settings_enhanced(OptionSettings[*])
    }
}

rule
 check_elastic_beanstalk_configuration_templates(elastic_beanstalk_configuration_templates)
 {
    %elastic_beanstalk_configuration_templates {
        # Scenario 7
        check_option_settings_with_enhanced_health_reporting(this) or
        # Scenario 6
        check_option_settings_without_health_reporting(this) or
        # Scenario 5
        check_option_settings_not_exists_or_is_empty_list(this)
    }
}

rule
 check_option_settings_with_enhanced_health_reporting(elastic_beanstalk_configuration_templates
 {
    %elastic_beanstalk_configuration_templates [
        filter_option_settings_with_health_reporting(this)
    ] {
        check_option_settings_enhanced(OptionSettings[*])
    }
}

rule
 filter_option_settings_with_health_reporting(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_or_is_non_empty_list(this)

        some OptionSettings[*] {
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_ENHANCED_HEALTH_REPORTING_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_SYSTEM_TYPE_OPTION_NAME
        }
    }
}
```

```
rule check_option_settings_enhanced(option_settings) {
    # Scenario 3, 4
    some %option_settings[*] {
        Namespace exists
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_ENHANCED_HEALTH_REPORTING_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_SYSTEM_TYPE_OPTION_NAME
        Value == %ELASTIC_BEANSTALK_ENHANCED_VALUE
    }

    # Scenario 7
    let option_setting_duplicates = OptionSettings [
        Namespace exists
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_ENHANCED_HEALTH_REPORTING_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_SYSTEM_TYPE_OPTION_NAME
        Value != %ELASTIC_BEANSTALK_ENHANCED_VALUE
    ]
    %option_setting_duplicates empty
}


rule
 check_option_settings_without_health_reporting(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_or_is_non_empty_list(this)

        let option_settings_with_health_reporting = OptionSettings [
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_ENHANCED_HEALTH_REPORTING_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_SYSTEM_TYPE_OPTION_NAME
        ]
        %option_settings_with_health_reporting empty
    }
}
```

```
rule check_option_settings_exists_or_is_non_empty_list(elastic_beanstalk_resource) {
    %elastic_beanstalk_resource {
        OptionSettings exists
        OptionSettings is_list
        OptionSettings not empty
    }
}

rule check_option_settings_not_exists_or_is_empty_list(configuration_template) {
    %configuration_template {
        OptionSettings not exists or
        check_is_empty_list(OptionSettings)
    }
}

rule check_is_empty_list(option_settings) {
    %option_settings {
        this is_list
        this empty
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICBEANSTALK.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - 'sts:AssumeRole'
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
  ElasticBeanstalkConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: "64bit Amazon Linux 2 v3.4.0 running Python 3.8"
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  InstanceRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - 'sts:AssumeRole'
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
  ElasticBeanstalkEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      SolutionStackName: "64bit Amazon Linux 2 v3.4.0 running Python 3.8"
      ApplicationName:
        Ref: App
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
      - Namespace: aws:elasticbeanstalk:healthreporting:system
        OptionName: SystemType
        Value: basic
```

[CT.ELASTICBEANSTALK.PR.2] Require an AWS Elastic Beanstalk environment to have managed platform updates configured

This control checks whether managed platform updates in AWS Elastic Beanstalk environments and configuration templates are activated.

- **Control objective:** Manage vulnerabilities

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticBeanstalk::Environment`, `AWS::ElasticBeanstalk::ConfigurationTemplate`
- **CloudFormation guard rule:** CT.ELASTICBEANSTALK.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICBEANSTALK.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICBEANSTALK.PR.2 example templates

Explanation

Managed platform updates ensure that the most recent platform fixes, updates, and features for the environment are installed. Keeping patch installations up to date is an important step in securing systems.

Usage considerations

- When you set up managed actions on AWS Elastic Beanstalk environments and configuration templates, you must provide `PreferredStartTime` and `UpdateLevel` option settings also.
- This control allows you to set up managed actions on AWS Elastic Beanstalk environments only, because environment-level settings take precedence over settings that are defined in configuration templates.
- This control does not allow you to deactivate managed actions on AWS Elastic Beanstalk configuration templates.

Remediation for rule failure

For AWS Elastic Beanstalk environments, create an `OptionSetting` with a `Namespace` value set to `aws:elasticbeanstalk:managedactions`, `OptionName` set to `ManagedActionsEnabled`,

and `Value` set to `true`. For Elastic Beanstalk configuration templates, create an `OptionSetting` with a `Namespace` value set to `aws:elasticbeanstalk:managedactions`, `OptionName` set to `ManagedActionsEnabled`, and `Value` set to `true`, or omit this setting to adopt the default value of `true`.

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Environment - Example

AWS Elastic Beanstalk environment configured with managed platform updates activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkEnvironment": {
        "Type": "AWS::ElasticBeanstalk::Environment",
        "Properties": {
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "ApplicationName": {
                "Ref": "App"
            },
            "OptionSettings": [
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions",
                    "OptionName": "ManagedActionsEnabled",
                    "Value": true
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions",
                    "OptionName": "PreferredStartTime",
                    "Value": "Tue:09:00"
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions",
```

```
                    "OptionName": "ServiceRoleForManagedUpdates",
                    "Value": "AWSServiceRoleForElasticBeanstalkManagedUpdates"
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions:platformupdate",
                    "OptionName": "UpdateLevel",
                    "Value": "patch"
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkEnvironment:
  Type: AWS::ElasticBeanstalk::Environment
  Properties:
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    ApplicationName: !Ref 'App'
    OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: ManagedActionsEnabled
        Value: true
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: PreferredStartTime
        Value: Tue:09:00
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: ServiceRoleForManagedUpdates
        Value: AWSServiceRoleForElasticBeanstalkManagedUpdates
      - Namespace: aws:elasticbeanstalk:managedactions:platformupdate
        OptionName: UpdateLevel
        Value: patch
```

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Configuration Template - Example One

AWS Elastic Beanstalk configuration template configured with managed platform updates enabled, by means of AWS CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkConfigurationTemplate": {
        "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
```

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Configuration Template - Example Two

AWS Elastic Beanstalk configuration template configured with managed platform updates enabled, by means of an entry in the `OptionSettings` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkConfigurationTemplate": {
        "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.0 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions",
                    "OptionName": "ManagedActionsEnabled",
                    "Value": true
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions",
                    "OptionName": "PreferredStartTime",
                    "Value": "Tue:09:00"
                },
                {
                    "Namespace": "aws:elasticbeanstalk:managedactions:platformupdate",
                    "OptionName": "UpdateLevel",
                    "Value": "minor"
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.0 running Python 3.8
    OptionSettings:
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: ManagedActionsEnabled
        Value: true
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: PreferredStartTime
        Value: Tue:09:00
      - Namespace: aws:elasticbeanstalk:managedactions:platformupdate
        OptionName: UpdateLevel
        Value: minor
```

CT.ELASTICBEANSTALK.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elastic_beanstalk_managed_updates_enabled_check
#
# Description:
#   This control checks whether managed platform updates in AWS Elastic Beanstalk
 environments and configuration templates are activated.
#
# Reports on:
#   AWS::ElasticBeanstalk::Environment, AWS::ElasticBeanstalk::ConfigurationTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElasticBeanstalk environment
 resources or ElasticBeanstalk
#            configuration template resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:managedactions' and an 'OptionName' property with a
 value of 'ManagedActionsEnabled'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource or an
 ElasticBeanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: An entry in the 'OptionSettings' list has a 'Namespace' property with a
 value of
#            'aws:elasticbeanstalk:managedactions'
#       And: That same entry has an 'OptionName' property with a value of
 'ManagedActionsEnabled'
#       And: That same entry has a 'Value' property with a value of anything other than
 bool(true), or the 'Value'
#            property is not provided.
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk configuration template
 resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
```

```
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk configuration template
 resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:managedactions' and an 'OptionName' property with a
 value of 'ManagedActionsEnabled'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource or an
 ElasticBeanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: Every entry in the 'OptionSettings' list that has both a 'Namespace'
 property with a value of
#            'aws:elasticbeanstalk:managedactions' and an 'OptionName' property with a
 value of 'ManagedActionsEnabled'
#            also has a 'Value' property with a value of bool(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_BEANSTALK_ENVIRONMENT_TYPE = "AWS::ElasticBeanstalk::Environment"
let ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE =
 "AWS::ElasticBeanstalk::ConfigurationTemplate"
let ELASTIC_BEANSTALK_MANAGED_ACTIONS_NAMESPACE = "aws:elasticbeanstalk:managedactions"
let ELASTIC_BEANSTALK_MANAGED_ACTIONS_OPTION_NAME = "ManagedActionsEnabled"
let ELASTIC_BEANSTALK_MANAGED_ACTIONS_ENABLED_VALUE = ["true", true]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_beanstalk_environments = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE ]
let elastic_beanstalk_configuration_templates = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE ]

#
```

```
# Primary Rules
#
rule elastic_beanstalk_managed_updates_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elastic_beanstalk_environments not empty {
    check_elastic_beanstalk_environments(%elastic_beanstalk_environments.Properties)
        <<
        [CT.ELASTICBEANSTALK.PR.2]: Require an AWS Elastic Beanstalk environment to
 have managed platform updates configured
            [FIX]: For AWS Elastic Beanstalk environments, create an 'OptionSetting'
 with a 'Namespace' value set to  'aws:elasticbeanstalk:managedactions', 'OptionName'
 set to 'ManagedActionsEnabled', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, create an 'OptionSetting' with a 'Namespace' value set to
 'aws:elasticbeanstalk:managedactions', 'OptionName' set to 'ManagedActionsEnabled',
 and 'Value' set to 'true', or omit this setting to adopt the default value of 'true'.
        >>
}

rule elastic_beanstalk_managed_updates_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elastic_beanstalk_configuration_templates not empty {

 check_elastic_beanstalk_configuration_templates(%elastic_beanstalk_configuration_templates.Pro
        <<
        [CT.ELASTICBEANSTALK.PR.2]: Require an AWS Elastic Beanstalk environment to
 have managed platform updates configured
            [FIX]: For AWS Elastic Beanstalk environments, create an 'OptionSetting'
 with a 'Namespace' value set to  'aws:elasticbeanstalk:managedactions', 'OptionName'
 set to 'ManagedActionsEnabled', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, create an 'OptionSetting' with a 'Namespace' value set to
 'aws:elasticbeanstalk:managedactions', 'OptionName' set to 'ManagedActionsEnabled',
 and 'Value' set to 'true', or omit this setting to adopt the default value of 'true'.
        >>
}

rule elastic_beanstalk_managed_updates_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE) {
    check_elastic_beanstalk_environments(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_ENVIRONMENT_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.2]: Require an AWS Elastic Beanstalk environment to
 have managed platform updates configured
```

```
            [FIX]: For AWS Elastic Beanstalk environments, create an 'OptionSetting'
 with a 'Namespace' value set to  'aws:elasticbeanstalk:managedactions', 'OptionName'
 set to 'ManagedActionsEnabled', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, create an 'OptionSetting' with a 'Namespace' value set to
 'aws:elasticbeanstalk:managedactions', 'OptionName' set to 'ManagedActionsEnabled',
 and 'Value' set to 'true', or omit this setting to adopt the default value of 'true'.
        >>
}

rule elastic_beanstalk_managed_updates_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE) {
    check_elastic_beanstalk_configuration_templates(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.2]: Require an AWS Elastic Beanstalk environment to
 have managed platform updates configured
            [FIX]: For AWS Elastic Beanstalk environments, create an 'OptionSetting'
 with a 'Namespace' value set to  'aws:elasticbeanstalk:managedactions', 'OptionName'
 set to 'ManagedActionsEnabled', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, create an 'OptionSetting' with a 'Namespace' value set to
 'aws:elasticbeanstalk:managedactions', 'OptionName' set to 'ManagedActionsEnabled',
 and 'Value' set to 'true', or omit this setting to adopt the default value of 'true'.
        >>
}

#
# Parameterized Rules
#
rule check_elastic_beanstalk_environments(elastic_beanstalk_environments) {
    %elastic_beanstalk_environments {
        # Scenario 2
        check_option_settings_exists_and_is_non_empty_list(this)

        # Scenario 3, 4, 7
        check_option_settings_managed_actions_enabled(OptionSettings[*])
    }
}

rule
 check_elastic_beanstalk_configuration_templates(elastic_beanstalk_configuration_templates)
 {
    %elastic_beanstalk_configuration_templates {
        # Scenario 7
        check_option_settings_with_managed_actions_enabled(this) or
```

```
        # Scenario 6
        check_option_settings_without_managed_actions(this) or
        # Scenario 5
        check_option_settings_not_exists_or_is_empty_list(this)
    }
}

rule
 check_option_settings_with_managed_actions_enabled(elastic_beanstalk_configuration_templates)
 {
    %elastic_beanstalk_configuration_templates [
        filter_option_settings_with_managed_actions(this)
    ] {
        check_option_settings_managed_actions_enabled(OptionSettings[*])
    }
}

rule
 filter_option_settings_with_managed_actions(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_and_is_non_empty_list(this)

        some OptionSettings[*] {
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_OPTION_NAME
        }
    }
}

rule check_option_settings_managed_actions_enabled(option_settings) {
    # Scenario 3, 4
    some %option_settings[*] {
        Namespace exists
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_OPTION_NAME
        Value in %ELASTIC_BEANSTALK_MANAGED_ACTIONS_ENABLED_VALUE
    }
```

```
    # Scenario 7
    let option_setting_duplicates = OptionSettings [
        Namespace exists
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_OPTION_NAME
        Value not in %ELASTIC_BEANSTALK_MANAGED_ACTIONS_ENABLED_VALUE
    ]
    %option_setting_duplicates empty
}


rule
 check_option_settings_without_managed_actions(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_and_is_non_empty_list(this)

        let option_settings_with_managed_actions = OptionSettings [
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_MANAGED_ACTIONS_OPTION_NAME
        ]
        %option_settings_with_managed_actions empty
    }
}

rule check_option_settings_exists_and_is_non_empty_list(elastic_beanstalk_resource) {
    %elastic_beanstalk_resource {
        OptionSettings exists
        OptionSettings is_list
        OptionSettings not empty
    }
}

rule check_option_settings_not_exists_or_is_empty_list(configuration_template) {
    %configuration_template {
        OptionSettings not exists or
        check_is_empty_list(OptionSettings)
```

```
    }
}

rule check_is_empty_list(option_settings) {
    %option_settings {
        this is_list
        this empty
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICBEANSTALK.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
```

```
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - 'sts:AssumeRole'
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
  ElasticBeanstalkConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: "64bit Amazon Linux 2 v3.4.0 running Python 3.8"
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - 'sts:AssumeRole'
```

```
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
  ElasticBeanstalkEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      SolutionStackName: "64bit Amazon Linux 2 v3.4.0 running Python 3.8"
      ApplicationName:
        Ref: App
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
      - Namespace: aws:elasticbeanstalk:managedactions
        OptionName: ManagedActionsEnabled
        Value: false
```

[CT.ELASTICBEANSTALK.PR.3] Require an AWS Elastic Beanstalk environment to have a logging configuration

This control checks whether an AWS Elastic Beanstalk environment is configured to send logs to Amazon CloudWatch Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticBeanstalk::Environment`, `AWS::ElasticBeanstalk::ConfigurationTemplate`
- **CloudFormation guard rule:** CT.ELASTICBEANSTALK.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICBEANSTALK.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICBEANSTALK.PR.3 example templates

Explanation

Monitoring is an important part of maintaining the reliability, availability, and performance of your AWS solutions. We recommend that you collect monitoring data from all of the parts of your AWS solution, so that you can debug a multi-point failure, if one occurs. From a security perspective, logging is an important feature to enable for future forensics efforts in the case of a security incident.

Usage considerations

- This control requires only enabling logging to Amazon CloudWatch Logs on AWS Elastic Beanstalk environments, because environment level settings take precedence over settings defined in configuration templates.
- This control does not allow explicitly disabling logging to Amazon CloudWatch Logs on AWS Elastic Beanstalk configuration templates.

Remediation for rule failure

For AWS Elastic Beanstalk environments, establish an `OptionSetting` with a `Namespace` set to `aws:elasticbeanstalk:cloudwatch:logs`, `OptionName` set to `StreamLogs`, and `Value` set to `true`. For Elastic Beanstalk configuration templates, establish an `OptionSetting` with a `Namespace` set to `aws:elasticbeanstalk:cloudwatch:logs`, `OptionName` set to `StreamLogs`, and `Value` set to `true`, or omit this `OptionSetting`.

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Environment - Example

An AWS Elastic Beanstalk environment configured to stream logs to Amazon CloudWatch Logs by means of an entry in the `OptionSettings` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkEnvironment": {
        "Type": "AWS::ElasticBeanstalk::Environment",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.1 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                },
                {
                    "Namespace": "aws:elasticbeanstalk:cloudwatch:logs",
                    "OptionName": "StreamLogs",
                    "Value": true
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkEnvironment:
  Type: AWS::ElasticBeanstalk::Environment
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
    OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
      - Namespace: aws:elasticbeanstalk:cloudwatch:logs
        OptionName: StreamLogs
        Value: true
```

The examples that follow show how to implement this remediation.

AWS Elastic Beanstalk Configuration Template - Example

AWS Elastic Beanstalk configuration template configured to stream logs to Amazon CloudWatch Logs by means of an entry in the `OptionSettings` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticBeanstalkConfigurationTemplate": {
        "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
        "Properties": {
            "ApplicationName": {
                "Ref": "App"
            },
            "SolutionStackName": "64bit Amazon Linux 2 v3.4.1 running Python 3.8",
            "OptionSettings": [
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "InstanceProfile"
                    }
                },
                {
                    "Namespace": "aws:elasticbeanstalk:cloudwatch:logs",
                    "OptionName": "StreamLogs",
                    "Value": true
                }
            ]
        }
    }
}
```

YAML example

```
ElasticBeanstalkConfigurationTemplate:
  Type: AWS::ElasticBeanstalk::ConfigurationTemplate
  Properties:
    ApplicationName: !Ref 'App'
    SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
    OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref 'InstanceProfile'
      - Namespace: aws:elasticbeanstalk:cloudwatch:logs
        OptionName: StreamLogs
        Value: true
```

CT.ELASTICBEANSTALK.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elastic_beanstalk_logs_to_cloudwatch_check
#
# Description:
#   This control checks whether an AWS Elastic Beanstalk environment is configured to
 send logs to Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::ElasticBeanstalk::Environment, AWS::ElasticBeanstalk::ConfigurationTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any ElasticBeanstalk environment
 resources or ElasticBeanstalk
#            configuration template resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:cloudwatch:logs' and an 'OptionName' property with a
 value of 'StreamLogs'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource or an
 ElasticBeanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: An entry in the 'OptionSettings' list has a 'Namespace' property with a
 value of
#            'aws:elasticbeanstalk:cloudwatch:logs'
#       And: That same entry has an 'OptionName' property with a value of 'StreamLogs'
#       And: That same entry has a 'Value' property with a value of anything other than
 bool(true), or the 'Value'
#            property is not provided.
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk configuration template
 resource
#       And: 'OptionSettings' is not present in the resource properties or is an empty
 list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk configuration template
 resource
```

```
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: No entry in the 'OptionSettings' list has both a 'Namespace' property with
 a value of
#            'aws:elasticbeanstalk:cloudwatch:logs' and an 'OptionName' property with a
 value of 'StreamLogs'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticBeanstalk environment resource or an
 ElasticBeanstalk
#            configuration template resource
#       And: 'OptionSettings' is present in the resource properties as a non-empty list
#       And: Every entry in the 'OptionSettings' list that has both a 'Namespace'
 property with a value of
#            'aws:elasticbeanstalk:cloudwatch:logs' and an 'OptionName' property with a
 value of 'StreamLogs'
#            also has a 'Value' property with a value of bool(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_BEANSTALK_ENVIRONMENT_TYPE = "AWS::ElasticBeanstalk::Environment"
let ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE =
 "AWS::ElasticBeanstalk::ConfigurationTemplate"
let ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_NAMESPACE =
 "aws:elasticbeanstalk:cloudwatch:logs"
let ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_OPTION_NAME = "StreamLogs"
let ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_ENABLED_VALUE = ["true", true]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_beanstalk_environments = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE ]
let elastic_beanstalk_configuration_templates = Resources.*[ Type ==
 %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule elastic_beanstalk_logs_to_cloudwatch_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                    %elastic_beanstalk_environments
 not empty {
    check_elastic_beanstalk_environments(%elastic_beanstalk_environments.Properties)
        <<
        [CT.ELASTICBEANSTALK.PR.3]: Require an AWS Elastic Beanstalk environment to
 have a logging configuration
        [FIX]: For AWS Elastic Beanstalk environments, establish an 'OptionSetting'
 with a 'Namespace' set to 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName'
 set to 'StreamLogs', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, establish an 'OptionSetting' with a 'Namespace' set to
 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName' set to 'StreamLogs', and 'Value'
 set to 'true', or omit this 'OptionSetting'.
        >>
}

rule elastic_beanstalk_logs_to_cloudwatch_check when is_cfn_template(%INPUT_DOCUMENT)

 %elastic_beanstalk_configuration_templates not empty {

 check_elastic_beanstalk_configuration_templates(%elastic_beanstalk_configuration_templates.Pro
        <<
        [CT.ELASTICBEANSTALK.PR.3]: Require an AWS Elastic Beanstalk environment to
 have a logging configuration
        [FIX]: For AWS Elastic Beanstalk environments, establish an 'OptionSetting'
 with a 'Namespace' set to 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName'
 set to 'StreamLogs', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, establish an 'OptionSetting' with a 'Namespace' set to
 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName' set to 'StreamLogs', and 'Value'
 set to 'true', or omit this 'OptionSetting'.
        >>
}

rule elastic_beanstalk_logs_to_cloudwatch_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_BEANSTALK_ENVIRONMENT_TYPE) {
    check_elastic_beanstalk_environments(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_ENVIRONMENT_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.3]: Require an AWS Elastic Beanstalk environment to
 have a logging configuration
        [FIX]: For AWS Elastic Beanstalk environments, establish an 'OptionSetting'
 with a 'Namespace' set to 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName'
 set to 'StreamLogs', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, establish an 'OptionSetting' with a 'Namespace' set to
```

```
 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName' set to 'StreamLogs', and 'Value'
 set to 'true', or omit this 'OptionSetting'.
        >>
}

rule elastic_beanstalk_logs_to_cloudwatch_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE) {
    check_elastic_beanstalk_configuration_templates(%INPUT_DOCUMENT.
%ELASTIC_BEANSTALK_CONFIGURATION_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.ELASTICBEANSTALK.PR.3]: Require an AWS Elastic Beanstalk environment to
 have a logging configuration
        [FIX]: For AWS Elastic Beanstalk environments, establish an 'OptionSetting'
 with a 'Namespace' set to 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName'
 set to 'StreamLogs', and 'Value' set to 'true'. For Elastic Beanstalk
 configuration templates, establish an 'OptionSetting' with a 'Namespace' set to
 'aws:elasticbeanstalk:cloudwatch:logs', 'OptionName' set to 'StreamLogs', and 'Value'
 set to 'true', or omit this 'OptionSetting'.
        >>
}

#
# Parameterized Rules
#
rule check_elastic_beanstalk_environments(elastic_beanstalk_environments) {
    %elastic_beanstalk_environments {
        # Scenario 2
        check_option_settings_exists_and_is_non_empty_list(this)

        # Scenario 3, 4, 7
        check_option_settings_cloudwatch_logs_enabled(OptionSettings[*])
    }
}

rule
 check_elastic_beanstalk_configuration_templates(elastic_beanstalk_configuration_templates)
 {
    %elastic_beanstalk_configuration_templates {
        # Scenario 7
        check_option_settings_with_cloudwatch_logs_enabled(this) or
        # Scenario 6
        check_option_settings_without_cloudwatch_logs(this) or
        # Scenario 5
        check_option_settings_not_exists_or_is_empty_list(this)
```

```
    }
}

rule
 check_option_settings_with_cloudwatch_logs_enabled(elastic_beanstalk_configuration_templates)
 {
    %elastic_beanstalk_configuration_templates [
        filter_option_settings_with_cloudwatch_logs(this)
    ] {
        check_option_settings_cloudwatch_logs_enabled(OptionSettings[*])
    }
}

rule
 filter_option_settings_with_cloudwatch_logs(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_and_is_non_empty_list(this)

        some OptionSettings[*] {
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_OPTION_NAME
        }
    }
}

rule check_option_settings_cloudwatch_logs_enabled(option_settings) {
    # Scenario 3, 4
    some %option_settings[*] {
        Namespace exists
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_OPTION_NAME
        Value in %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_ENABLED_VALUE
    }

    # Scenario 7
    let option_setting_duplicates = OptionSettings [
        Namespace exists
```

```
        OptionName exists
        Value exists

        Namespace == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_NAMESPACE
        OptionName == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_OPTION_NAME
        Value not in %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_ENABLED_VALUE
    ]
    %option_setting_duplicates empty
}


rule
 check_option_settings_without_cloudwatch_logs(elastic_beanstalk_configuration_templates)
 {
    some %elastic_beanstalk_configuration_templates {
        check_option_settings_exists_and_is_non_empty_list(this)

        let option_settings_with_cloudwatch_logs = OptionSettings [
            Namespace exists
            OptionName exists

            Namespace == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_NAMESPACE
            OptionName == %ELASTIC_BEANSTALK_CLOUDWATCH_LOGS_OPTION_NAME
        ]
        %option_settings_with_cloudwatch_logs empty
    }
}

rule check_option_settings_exists_and_is_non_empty_list(elastic_beanstalk_resource) {
    %elastic_beanstalk_resource {
        OptionSettings exists
        OptionSettings is_list
        OptionSettings not empty
    }
}

rule check_option_settings_not_exists_or_is_empty_list(configuration_template) {
    %configuration_template {
        OptionSettings not exists or
        check_is_empty_list(OptionSettings)
    }
}

rule check_is_empty_list(option_settings) {
```

```
    %option_settings {
        this is_list
        this empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICBEANSTALK.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - sts:AssumeRole
```

```
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
    Properties: {}
  ElasticBeanstalkEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
      - Namespace: aws:elasticbeanstalk:cloudwatch:logs
        OptionName: StreamLogs
        Value: true
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - sts:AssumeRole
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
```

```
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
    Properties: {}
  ElasticBeanstalkConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
      - Namespace: aws:elasticbeanstalk:cloudwatch:logs
        OptionName: StreamLogs
        Value: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - sts:AssumeRole
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
```

```
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
    Properties: {}
  ElasticBeanstalkEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  InstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - elasticbeanstalk.amazonaws.com
          Action:
          - sts:AssumeRole
  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
      - Ref: InstanceRole
  App:
    Type: AWS::ElasticBeanstalk::Application
```

```
    Properties: {}
  ElasticBeanstalkConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: App
      SolutionStackName: 64bit Amazon Linux 2 v3.4.1 running Python 3.8
      OptionSettings:
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value:
          Ref: InstanceProfile
      - Namespace: aws:elasticbeanstalk:cloudwatch:logs
        OptionName: StreamLogs
        Value: false
```

Amazon Elastic Compute Cloud (Amazon EC2) controls

Topics

- [CT.EC2.PR.1] Require an Amazon EC2 launch template to have IMDSv2 configured
- [CT.EC2.PR.2] Require that Amazon EC2 launch templates restrict the token hop limit to a maximum of one
- [CT.EC2.PR.3] Require that any Amazon EC2 security group rule does not use the source IP range 0.0.0.0/0 or ::/0 for ports other than 80 and 443
- [CT.EC2.PR.4] Require that any Amazon EC2 security group rule does not use the source IP range 0.0.0.0/0 or ::/0 for specific high-risk ports
- [CT.EC2.PR.5] Require any Amazon EC2 network ACL to prevent ingress from 0.0.0.0/0 to port 22 or port 3389
- [CT.EC2.PR.6] Require that Amazon EC2 transit gateways refuse automatic Amazon VPC attachment requests
- [CT.EC2.PR.7] Require an Amazon EBS volume resource to be encrypted at rest when defined by means of the AWS::EC2::Instance BlockDeviceMappings property or AWS::EC2::Volume resource type
- [CT.EC2.PR.8] Require an Amazon EC2 instance to set AssociatePublicIpAddress to false on a new network interface created by means of the NetworkInterfaces property in the AWS::EC2::Instance resource

- [CT.EC2.PR.9] Require any Amazon EC2 launch template not to auto-assign public IP addresses to network interfaces
- [CT.EC2.PR.10] Require Amazon EC2 launch templates to have Amazon CloudWatch detailed monitoring activated
- [CT.EC2.PR.11] Require that an Amazon EC2 subnet does not automatically assign public IP addresses
- [CT.EC2.PR.12] Require an Amazon EC2 instance to specify at most one network interface by means of the NetworkInterfaces property in the AWS::EC2::Instance resource
- [CT.EC2.PR.13] Require an Amazon EC2 instance to have detailed monitoring enabled
- [CT.EC2.PR.14] Require an Amazon EBS volume configured through an Amazon EC2 launch template to encrypt data at rest
- [CT.EC2.PR.15] Require an Amazon EC2 instance to use an AWS Nitro instance type when creating from the 'AWS::EC2::LaunchTemplate' resource type
- [CT.EC2.PR.16] Require an Amazon EC2 instance to use an AWS Nitro instance type when created using the 'AWS::EC2::Instance' resource type
- [CT.EC2.PR.17] Require an Amazon EC2 dedicated host to use an AWS Nitro instance type
- [CT.EC2.PR.18] Require an Amazon EC2 fleet to override only those launch templates with AWS Nitro instance types
- [CT.EC2.PR.19] Require an Amazon EC2 instance to use an AWS Nitro instance type that supports encryption in-transit between instances when created using the AWS::EC2::Instance resource type
- [CT.EC2.PR.20] Require an Amazon EC2 fleet to override only those launch templates with AWS Nitro instance types that support encryption in transit between instances

[CT.EC2.PR.1] Require an Amazon EC2 launch template to have IMDSv2 configured

This control checks whether your Amazon EC2 launch templates are configured with Instance Metadata Service Version 2 (IMDSv2).

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.1 example templates

Explanation

Instance metadata configures and manages your running instances. The IMDS provides access to temporary, frequently rotated credentials, so you don't need to distribute sensitive credentials to instances, either manually or programmatically. The IMDS is attached locally to every EC2 instance. It runs on a special IP address of 169.254.169.254. This IP address is accessible only to software that runs on the instance.

Version 2 of the IMDS adds protections for vulnerabilities that can be used to gain access to the IMDS: `Open website application firewalls`, `Open reverse proxies`, `Server-side request forgery (SSRF) vulnerabilities` and `'Open Layer 3 firewalls and network address translation (NAT)`.

AWS Control Tower recommends that you configure your EC2 instances with IMDSv2.

Usage considerations

- This control applies only to Amazon EC2 launch templates that allow access to instance metadata.

Remediation for rule failure

Within the `LaunchTemplateData` property, provide a `MetadataOptions` configuration and set the value of `HttpTokens` to `required`.

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example

Amazon EC2 launch template configured with IMDSv2 activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "InstanceType": "t3.micro",
                "ImageId": {
                    "Ref": "LatestAmiId"
                },
                "MetadataOptions": {
                    "HttpTokens": "required"
                }
            }
        }
    }
}
```

YAML example

```
EC2LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      InstanceType: t3.micro
      ImageId: !Ref 'LatestAmiId'
      MetadataOptions:
        HttpTokens: required
```

CT.EC2.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_launch_template_imdsv2_check
#
# Description:
#   This control checks whether your Amazon EC2 launch templates are configured with
 Instance Metadata Service Version 2 (IMDSv2).
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 launch template resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'LaunchTemplateData' has not been provided or
 'LaunchTemplateData.MetadataOptions.HttpEndpoint' has
#             been provided and is equal to 'disabled'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'LaunchTemplateData' has been provided
#       And: 'MetadataOptions.HttpEndpoint' in 'LaunchTemplateData' has not been
 provided or has been provided and
#            is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' in 'LaunchTemplateData' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
```

```
#       And: 'LaunchTemplateData' has been provided
#       And: 'MetadataOptions.HttpEndpoint' in 'LaunchTemplateData' has not been
 provided or has been provided and
#            is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' in 'LaunchTemplateData' has been provided and
 set to a value other than 'required'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'LaunchTemplateData' has been provided
#       And: 'MetadataOptions.HttpEndpoint' in 'LaunchTemplateData' has not been
 provided or has been provided and
#            is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' in 'LaunchTemplateData' has been provided and
 set to 'required'
#      Then: PASS

#
# Constants
#
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_imdsv2_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %ec2_launch_templates not empty {
    check(%ec2_launch_templates.Properties)
        <<
        [CT.EC2.PR.1]: Require an Amazon EC2 launch template to have IMDSv2 configured
            [FIX]: Within the 'LaunchTemplateData' property, provide a
 'MetadataOptions' configuration and set the value of 'HttpTokens' to 'required'.
        >>
}

rule ec2_launch_template_imdsv2_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.1]: Require an Amazon EC2 launch template to have IMDSv2 configured
            [FIX]: Within the 'LaunchTemplateData' property, provide a
 'MetadataOptions' configuration and set the value of 'HttpTokens' to 'required'.
        >>
}

#
# Parameterized Rules
#
rule check(launch_template) {
    %launch_template [
        # Scenario 2
        filter_launch_template_imds_enabled(this)
    ] {
        LaunchTemplateData exists
        LaunchTemplateData is_struct

        LaunchTemplateData {
            # Scenario 3, 4 and 5
            MetadataOptions exists
            MetadataOptions is_struct
            MetadataOptions {
                HttpTokens exists
                HttpTokens == "required"
            }
        }
    }
}

rule filter_launch_template_imds_enabled(launch_template) {
    %launch_template {
        LaunchTemplateData exists
        LaunchTemplateData is_struct
        LaunchTemplateData {
            MetadataOptions not exists or
            filter_metadata_options_imds_enabled(this)
        }
    }
}

rule filter_metadata_options_imds_enabled(metadata_options) {
    %metadata_options {
```

```
        MetadataOptions is_struct
        MetadataOptions {
            HttpEndpoint not exists or
            HttpEndpoint == "enabled"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceType: t3.micro
```

```
        ImageId:
          Ref: LatestAmiId
        MetadataOptions:
          HttpTokens: required
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceType: t3.micro
        ImageId:
          Ref: LatestAmiId
        MetadataOptions:
          HttpTokens: optional
```

[CT.EC2.PR.2] Require that Amazon EC2 launch templates restrict the token hop limit to a maximum of one

This control checks whether an Amazon EC2 launch template has a metadata token hop limit set to 1.

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.2 example templates

Explanation

The Amazon Instance Metadata Service (IMDS) provides metadata information about an Amazon EC2 instance, which is useful for application configuration. Restricting the HTTP PUT response for the metadata service to the EC2 instance protects the IMDS from unauthorized use.

The Time To Live (TTL) field in the IP packet is reduced by one on every hop. This reduction can be used to ensure that the packet does not travel outside EC2. IMDSv2 protects EC2 instances that may have been misconfigured as open routers, layer 3 firewalls, VPNs, tunnels, or NAT devices, which prevents unauthorized users from retrieving metadata. With IMDSv2, the PUT response that contains the secret token cannot travel outside the instance, because the default metadata response hop limit is set to 1. However, if this value is greater than 1, the token can leave the EC2 instance.

Usage considerations

- This control applies only to Amazon EC2 launch templates that allow access to instance metadata.
- This control is incompatible with Amazon EC2 launch templates that require a token hop limit of 2.

Remediation for rule failure

Within the `LaunchTemplateData` property, provide a `MetadataOptions` configuration with the value of `HttpPutResponseLimit` set to 1, or omit the `HttpPutResponseLimit` property to adopt the CloudFormation default value of 1.

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example One

Amazon EC2 launch template configured with access to instance metadata enabled and a token hop limit of 1, set by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "MetadataOptions": {
                    "HttpEndpoint": "enabled"
                }
            }
        }
    }
}
```

YAML example

```
EC2LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      MetadataOptions:
        HttpEndpoint: enabled
```

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example Two

Amazon EC2 launch template configured with access to instance metadata enabled and a token hop limit of 1, set by means of the `MetadataOptions` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "MetadataOptions": {
                    "HttpEndpoint": "enabled",
                    "HttpPutResponseHopLimit": 1
                }
            }
        }
    }
}
```

YAML example

```
EC2LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      MetadataOptions:
        HttpEndpoint: enabled
        HttpPutResponseHopLimit: 1
```

CT.EC2.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_launch_template_token_hop_limit_check
#
# Description:
```

```
#   This control checks whether an Amazon EC2 launch template has a metadata token hop
 limit set to '1'.
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any EC2 launch template resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.MetadataOptions' has been provided
#        And: 'LaunchTemplateData.MetadataOptions.HttpEndpoint' has been provided and
 is equal to 'disabled'
#       Then: SKIP
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.MetadataOptions' has been provided
#        And: 'LaunchTemplateData.MetadataOptions.HttpEndpoint' has not been provided
 or has been provided and is
#             equal to 'enabled'
#        And: 'LaunchTemplateData.MetadataOptions.HttpPutResponseHopLimit' has been
 provided and is not equal to
#             an integer of 1.
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.MetadataOptions' has not been provided
#       Then: PASS
#   Scenario: 5
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.MetadataOptions' has been provided
```

```
#        And: 'LaunchTemplateData.MetadataOptions.HttpEndpoint' has not been provided
 or has been provided and is
#             equal to 'enabled'
#        And: 'LaunchTemplateData.MetadataOptions.HttpPutResponseHopLimit' has not been
 provided
#       Then: PASS
#   Scenario: 6
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.MetadataOptions' has been provided
#        And: 'LaunchTemplateData.MetadataOptions.HttpEndpoint' has not been provided
 or has been provided and is
#             equal to 'enabled'
#        And: 'LaunchTemplateData.MetadataOptions.HttpPutResponseHopLimit' has been
 provided and is equal to an
#             integer of 1.
#       Then: PASS

#
# Constants
#
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_token_hop_limit_check when is_cfn_template(this)
                                                    %ec2_launch_templates not empty {
    check(%ec2_launch_templates.Properties)
        <<
        [CT.EC2.PR.2]: Require that Amazon EC2 launch templates restrict the token hop
 limit to a maximum of one
            [FIX]: Within the 'LaunchTemplateData' property, provide a
 'MetadataOptions' configuration with the value of 'HttpPutResponseLimit' set to '1',
 or omit the 'HttpPutResponseLimit' property to adopt the CloudFormation default value
 of '1'.
        >>
}
```

```
rule ec2_launch_template_token_hop_limit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.2]: Require that Amazon EC2 launch templates restrict the token hop
 limit to a maximum of one
            [FIX]: Within the 'LaunchTemplateData' property, provide a
 'MetadataOptions' configuration with the value of 'HttpPutResponseLimit' set to '1',
 or omit the 'HttpPutResponseLimit' property to adopt the CloudFormation default value
 of '1'.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_launch_template) {
  %ec2_launch_template[
      # Scenario 2, 3 and 4
      filter_launch_template(this)
  ] {
      # Scenario 5 and 6
      LaunchTemplateData {
          MetadataOptions not exists or
          MetadataOptions {
              HttpPutResponseHopLimit not exists or
              HttpPutResponseHopLimit == 1
          }
      }
  }
}

rule filter_launch_template(ec2_launch_template) {
  %ec2_launch_template {
      LaunchTemplateData exists
      LaunchTemplateData is_struct
      LaunchTemplateData {
          MetadataOptions not exists or
          filter_metadata_options_provided(this)
      }
  }
}
```

```
rule filter_metadata_options_provided(options) {
  %options {
      MetadataOptions is_struct
      MetadataOptions {
          HttpEndpoint not exists or
          HttpEndpoint == "enabled"
      }
  }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
  %doc {
    AWSTemplateFormatVersion exists  or
    Resources exists
  }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
```

```
      LaunchTemplateData:
        InstanceType: t3.micro
        ImageId:
          Ref: LatestAmiId
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        MetadataOptions:
          HttpPutResponseHopLimit: 2
```

[CT.EC2.PR.3] Require that any Amazon EC2 security group rule does not use the source IP range `0.0.0.0/0` or `::/0` for ports other than 80 and 443

This control checks whether an Amazon EC2 security group rule contains the string `0.0.0.0/0` or `::/0` as a source IP range. This control is not triggered if a rule allows connection to port 80 or 443 with TCP, UDP, ICMP, or ICMPv6. The use of managed prefix lists is not supported.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::SecurityGroup`, `AWS::EC2::SecurityGroupIngress`
- **CloudFormation guard rule:** CT.EC2.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.3 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.3 example templates

Explanation

Security groups provide stateful filtering of ingress and egress network traffic to AWS resources. Disallowing usage of the strings `0.0.0.0/0` or `::/0` helps protect against this common misconfiguration and encourages users to choose a range aligned with least-privilege principles.

AWS recommends a layered approach, to ensure that network access is provided only as necessary for your business requirements. Security group rules should follow the principle of least privileged access. Unrestricted access increases the opportunity for malicious activity. Unless a port is specifically allowed, the port should deny unrestricted access (any IP address with a `/0` suffix).

Usage considerations

- This control applies only to Amazon EC2 security group and EC2 security group ingress resources with ingress rules that allow inbound traffic from `0.0.0.0/0` or `::/0`
- This control does not allow use of the `SourcePrefixListId` property on Amazon EC2 Security Group and Amazon EC2 Security Group Ingress resources.

Remediation for rule failure

Ensure that security groups with ingress rules that allow TCP or UDP traffic from `0.0.0.0/0` or `::/0` allow traffic from ports `80` or `443` only.

The use of managed prefix lists is not supported.

The examples that follow show how to implement this remediation.

Amazon EC2 Security Group - Example One

Amazon EC2 Security Group allowing inbound TCP traffic from `0.0.0.0/0` on port `80`. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": {
                "Fn::Sub": "${AWS::StackName}-example"
            },
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 80,
                    "ToPort": 80
                }
            ]
        }
    }
}
```

YAML example

```
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: !Sub '${AWS::StackName}-example'
    SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: '0.0.0.0/0'
        FromPort: 80
        ToPort: 80
```

The examples that follow show how to implement this remediation.

Amazon EC2 Security Group - Example Two

Amazon EC2 Security Group allowing inbound TCP traffic from `0.0.0.0/0` on port `443`. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": {
                "Fn::Sub": "${AWS::StackName}-example"
            },
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 443,
                    "ToPort": 443
                }
            ]
        }
    }
}
```

YAML example

```
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: !Sub '${AWS::StackName}-example'
    SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: '0.0.0.0/0'
        FromPort: 443
        ToPort: 443
```

CT.EC2.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   vpc_sg_open_only_to_authorized_ports_check
#
# Description:
#    This control checks whether the Amazon EC2 security group contains the string
 '0.0.0.0/0' or '::/0' as a source IP range.
#    This control is not triggered if a rule allows connection to port 80 or 443 with
 TCP, UDP, ICMP, or ICMPv6.
#
# Reports on:
#   AWS::EC2::SecurityGroup, AWS::EC2::SecurityGroupIngress
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 security group or EC2
 security group ingress resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 security group resource or EC2
 security group ingress resource
#       And: The EC2 security group or EC2 security group ingress resource does not
 allow inbound traffic from a source
#            prefix list and has no rules allowing inbound traffic from source
 '0.0.0.0/0' or '::/0'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 security group resource or EC2
 security group ingress resource
#       And: The EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: The EC2 security group or EC2 security group ingress resource has a rule
 that allows all traffic
#            ('IpProtocol' is set to '-1' or another protocol number)
```

```
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 security group resource or EC2
 security group ingress resource
#       And: The EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: The EC2 security group or EC2 security group ingress resource has no rules
 that allow all traffic
#            ('IpProtocol' is not set to '-1' or another protocol number)
#       And: Ports allowed are not in the list of allowed ports
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 security group resource or EC2
 security group ingress resource
#       And: The EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: The EC2 security group or EC2 security group ingress resource has no rules
 that allow all traffic
#            ('IpProtocol' is not set to '-1' or another protocol number)
#       And: Ports allowed are in the list of allowed ports
#      Then: PASS

#
# Constants
#
let SECURITY_GROUP_TYPE = "AWS::EC2::SecurityGroup"
let SECURITY_GROUP_INGRESS_TYPE = "AWS::EC2::SecurityGroupIngress"
let ALLOWED_PORTS = [80, 443]
let AUTHORIZED_PROTOCOLS = ["tcp", "udp", "icmp", "icmpv6"]
let UNRESTRICTED_IPV4_RANGES = ["0.0.0.0/0"]
let UNRESTRICTED_IPV6_RANGES = ["::/0"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_security_groups = Resources[
    Type == %SECURITY_GROUP_TYPE
]
let ec2_security_group_ingress_rules = Resources[
```

```
    Type == %SECURITY_GROUP_INGRESS_TYPE
]

#
# Primary Rules
#
rule vpc_sg_open_only_to_authorized_ports_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %ec2_security_groups not empty {

    check_security_group(%ec2_security_groups.Properties)
        <<
        [CT.EC2.PR.3]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for ports other than 80 and 443
        [FIX]: Ensure that security groups with ingress rules that allow TCP or UDP
 traffic from '0.0.0.0/0' or '::/0' only allow traffic to ports 80 or 443. The use of
 managed prefix lists is not supported.
        >>
}

rule vpc_sg_open_only_to_authorized_ports_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %ec2_security_group_ingress_rules
 not empty {

    check_ingress_rule(%ec2_security_group_ingress_rules.Properties)
        <<
        [CT.EC2.PR.3]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for ports other than 80 and 443
        [FIX]: Ensure that security groups with ingress rules that allow TCP or UDP
 traffic from '0.0.0.0/0' or '::/0' only allow traffic to ports 80 or 443. The use of
 managed prefix lists is not supported.
        >>
}

rule vpc_sg_open_only_to_authorized_ports_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SECURITY_GROUP_TYPE) {

    check_security_group(%INPUT_DOCUMENT.%SECURITY_GROUP_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.3]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for ports other than 80 and 443
        [FIX]: Ensure that security groups with ingress rules that allow TCP or UDP
 traffic from '0.0.0.0/0' or '::/0' only allow traffic to ports 80 or 443. The use of
 managed prefix lists is not supported.
        >>
```

```
}

rule vpc_sg_open_only_to_authorized_ports_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SECURITY_GROUP_INGRESS_TYPE) {

    check_ingress_rule(%INPUT_DOCUMENT.%SECURITY_GROUP_INGRESS_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.3]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for ports other than 80 and 443
        [FIX]: Ensure that security groups with ingress rules that allow TCP or UDP
 traffic from '0.0.0.0/0' or '::/0' only allow traffic to ports 80 or 443. The use of
 managed prefix lists is not supported.
        >>
}

#
# Parameterized Rules
#
rule check_security_group(security_group) {
    %security_group [
        SecurityGroupIngress exists
        SecurityGroupIngress is_list
        SecurityGroupIngress not empty
    ] {
        SecurityGroupIngress[*] {
            check_ingress_rule(this)
        }
    }
}

rule check_ingress_rule(ingress_rule) {
    %ingress_rule[ CidrIp in %UNRESTRICTED_IPV4_RANGES or
                   CidrIpv6 in %UNRESTRICTED_IPV6_RANGES or
                   SourcePrefixListId exists ] {
        # Scenario 3
        IpProtocol exists
        IpProtocol in %AUTHORIZED_PROTOCOLS

        when IpProtocol in ["tcp", "udp"] {
            FromPort exists
            ToPort exists
            # Scenarios 4 and 5
            check_ports(FromPort, ToPort)
        }
```

```
    }
}

rule check_ports(from_port, to_port) {
    %from_port in %ALLOWED_PORTS
    %to_port in %ALLOWED_PORTS
    %from_port in %to_port
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: 0.0.0.0/0
        FromPort: 80
```

```
        ToPort: 80
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  PrefixList:
    Type: AWS::EC2::PrefixList
    Properties:
      PrefixListName:
        Fn::Sub: ${AWS::StackName}-example
      AddressFamily: IPv4
      MaxEntries: 10
      Entries:
        - Cidr: "0.0.0.0/0"
          Description: Public internet
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId:
        Fn::GetAtt: [ SecurityGroup, GroupId ]
      IpProtocol: -1
      SourcePrefixListId:
        Ref: PrefixList
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
```

```
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId:
        Fn::GetAtt: [ SecurityGroup, GroupId ]
      IpProtocol: udp
      CidrIp: 0.0.0.0/0
      FromPort: 80
      ToPort: 90
```

[CT.EC2.PR.4] Require that any Amazon EC2 security group rule does not use the source IP range `0.0.0.0/0` or `::/0` for specific high-risk ports

This control checks whether an Amazon EC2 security group rule that contains the strings `0.0.0.0/0` or `::/0` as a source IP range does not allow incoming TCP, UDP, ICMP, or ICMPv6 traffic to the following ports: 3389, 20, 23, 110, 143, 3306, 8080, 1433, 9200, 9300, 25, 445, 135, 21, 1434, 4333, 5432, 5500, 5601, 22, 3000, 5000, 8088, 8888. The use of managed prefix lists is not supported.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::SecurityGroup`, `AWS::EC2::SecurityGroupIngress`
- **CloudFormation guard rule:** CT.EC2.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.4 example templates

Explanation

Security groups provide stateful filtering of ingress and egress network traffic to AWS resources. Disallowing usage of the strings `0.0.0.0/0` or `::/0` helps protect against this common misconfiguration and encourages users to choose a range aligned with least-privilege principles.

AWS recommends a layered approach, to ensure that network access is provided only as necessary for your business requirements. No security group should allow unrestricted ingress access to the following ports:

3389, 20, 23, 110, 143, 3306, 8080, 1433, 9200, 9300, 25, 445, 135, 21, 1434, 4333, 5432, 5500, 5601, 22, 3000, 5000, 8088, 8888.

Unrestricted access (0.0.0.0/0) increases opportunities for malicious activity, such as hacking, denial-of-service attacks, and loss of data.

Usage considerations

- This control applies only to Amazon EC2 security group and security group ingress resources with ingress rules that allow inbound traffic from `0.0.0.0/0` or `::/0`.
- This control does not allow use of the `SourcePrefixListId` property on Amazon EC2 Security Group and Amazon EC2 Security Group Ingress resources.

Remediation for rule failure

Remove Amazon EC2 security group ingress rules that allow traffic from `0.0.0.0/0` or `::/0` to high-risk ports: 3389, 20, 23, 110, 143, 3306, 8080, 1433, 9200, 9300, 25, 445, 135, 21, 1434, 4333, 5432, 5500, 5601, 22, 3000, 5000, 8088, 8888.

The use of managed prefix lists is not supported.

The examples that follow show how to implement this remediation.

Amazon EC2 Security Group - Example

Amazon EC2 security group configured to allow traffic from the source IP range `0.0.0.0/0` or `::/0` on a port range that does not include a high-risk port. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "sample-security-group",
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 80,
                    "ToPort": 80
                }
            ]
        }
    }
}
```

YAML example

```
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: sample-security-group
    SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: '0.0.0.0/0'
        FromPort: 80
        ToPort: 80
```

The examples that follow show how to implement this remediation.

Amazon EC2 Security Group Ingress Rule - Example

Amazon EC2 security group ingress rule configured to allow traffic from the source IP range `0.0.0.0/0` or `::/0` on a port range that does not include a high-risk port. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityGroupIngress": {
        "Type": "AWS::EC2::SecurityGroupIngress",
        "Properties": {
            "GroupId": {
                "Fn::GetAtt": [
                    "SecurityGroup",
                    "GroupId"
                ]
            },
            "IpProtocol": "tcp",
            "CidrIp": "0.0.0.0/0",
            "FromPort": 80,
            "ToPort": 90
        }
    }
}
```

YAML example

```
SecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt 'SecurityGroup.GroupId'
    IpProtocol: tcp
    CidrIp: '0.0.0.0/0'
    FromPort: 80
    ToPort: 90
```

CT.EC2.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   vpc_sg_restricted_common_ports_check
#
```

```
# Description:
#   This control checks whether an Amazon EC2 security group rule that contains the
 strings '0.0.0.0/0' or '::/0' as a source IP range
#   does not allow incoming TCP, UDP, ICMP, ICMPv6 traffic to the following ports:
 '3389', '20', '23', '110', '143',
#   '3306', '8080', '1433', '9200', '9300', '25', '445', '135', '21', '1434', '4333',
 '5432', '5500', '5601', '22', '3000', '5000',
#   '8088', '8888'.
#
# Reports on:
#   AWS::EC2::SecurityGroup, AWS::EC2::SecurityGroupIngress
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 security group or EC2 security
 group ingress resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 security group resource or EC2 security
 group ingress resource
#       And: EC2 security group or EC2 security group ingress resource does not allow
 inbound traffic from a source
#            prefix list and has no rules allowing inbound traffic from source
 '0.0.0.0/0' or '::/0'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 security group resource or EC2 security
 group ingress resource
#       And: EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: EC2 security group or EC2 security group ingress resource has a rule that
 allows all traffic
#            ('IpProtocol' is set to '-1' or another protocol number)
#      Then: FAIL
```

```
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 security group resource or EC2 security
 group ingress resource
#       And: EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: EC2 security group or EC2 security group ingress resource has no rules
 that allow all traffic
#            ('IpProtocol' is not set to '-1' or another protocol number)
#       And: Ports allowed are in the list of blocked ports
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 security group resource or EC2 security
 group ingress resource
#       And: EC2 security group or EC2 security group ingress resource has rules
 allowing inbound traffic
#            from a source prefix list, or source '0.0.0.0/0' or '::/0'
#       And: EC2 security group or EC2 security group ingress resource has no rules
 that allow all traffic
#            ('IpProtocol' is not set to '-1' or another protocol number)
#       And: Ports allowed are not in the list of blocked ports
#      Then: PASS

#
# Constants
#
let SECURITY_GROUP_TYPE = "AWS::EC2::SecurityGroup"
let SECURITY_GROUP_INGRESS_TYPE = "AWS::EC2::SecurityGroupIngress"
let BLOCKED_PORTS = [3389, 20, 23, 110, 143, 3306, 8080, 1433, 9200, 9300, 25, 445,
 135, 21, 1434, 4333, 5432, 5500,
                     5601, 22, 3000, 5000, 8088, 8888]
let AUTHORIZED_PROTOCOLS = ["tcp", "udp", "icmp", "icmpv6"]
let UNRESTRICTED_IPV4_RANGES = ["0.0.0.0/0"]
let UNRESTRICTED_IPV6_RANGES = ["::/0"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_security_groups = Resources[
    Type == %SECURITY_GROUP_TYPE
]
```

```
let ec2_security_group_ingress_rules = Resources[
    Type == %SECURITY_GROUP_INGRESS_TYPE
]

#
# Primary Rules
#
rule vpc_sg_restricted_common_ports_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %ec2_security_groups not empty {

    check_security_group(%ec2_security_groups.Properties)
        <<
        [CT.EC2.PR.4]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for specific high-risk ports
            [FIX]: Remove Amazon EC2 security group ingress rules that allow traffic
 from '0.0.0.0/0' or '::/0' to high-risk ports: '3389', '20', '23', '110', '143',
 '3306', '8080', '1433', '9200', '9300', '25', '445', '135', '21', '1434', '4333',
 '5432', '5500', '5601', '22', '3000', '5000', '8088', '8888'. The use of managed
 prefix lists is not supported.
        >>
}

rule vpc_sg_restricted_common_ports_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %ec2_security_group_ingress_rules not
 empty {

    check_ingress_rule(%ec2_security_group_ingress_rules.Properties)
        <<
        [CT.EC2.PR.4]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for specific high-risk ports
            [FIX]: Remove Amazon EC2 security group ingress rules that allow traffic
 from '0.0.0.0/0' or '::/0' to high-risk ports: '3389', '20', '23', '110', '143',
 '3306', '8080', '1433', '9200', '9300', '25', '445', '135', '21', '1434', '4333',
 '5432', '5500', '5601', '22', '3000', '5000', '8088', '8888'. The use of managed
 prefix lists is not supported.
        >>
}

rule vpc_sg_restricted_common_ports_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SECURITY_GROUP_TYPE) {

    check_security_group(%INPUT_DOCUMENT.%SECURITY_GROUP_TYPE.resourceProperties)
        <<
```

```
        [CT.EC2.PR.4]: Require that any Amazon EC2 security group rule does not use the
 source IP range 0.0.0.0/0 or ::/0 for specific high-risk ports
            [FIX]: Remove Amazon EC2 security group ingress rules that allow traffic
 from '0.0.0.0/0' or '::/0' to high-risk ports: '3389', '20', '23', '110', '143',
 '3306', '8080', '1433', '9200', '9300', '25', '445', '135', '21', '1434', '4333',
 '5432', '5500', '5601', '22', '3000', '5000', '8088', '8888'. The use of managed
 prefix lists is not supported.
        >>
}

rule vpc_sg_restricted_common_ports_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SECURITY_GROUP_INGRESS_TYPE) {

    check_ingress_rule(%INPUT_DOCUMENT.%SECURITY_GROUP_INGRESS_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.4]: Require that any Amazon EC2 security group ruledoes not use the
 source IP range 0.0.0.0/0 or ::/0 for specific high-risk ports
            [FIX]: Remove Amazon EC2 security group ingress rules that allow traffic
 from '0.0.0.0/0' or '::/0' to high-risk ports: '3389', '20', '23', '110', '143',
 '3306', '8080', '1433', '9200', '9300', '25', '445', '135', '21', '1434', '4333',
 '5432', '5500', '5601', '22', '3000', '5000', '8088', '8888'. The use of managed
 prefix lists is not supported.
        >>
}

#
# Parameterized Rules
#
rule check_security_group(security_group) {
    %security_group [
        SecurityGroupIngress exists
        SecurityGroupIngress is_list
        SecurityGroupIngress not empty
    ] {
        SecurityGroupIngress[*] {
            check_ingress_rule(this)
        }
    }
}

rule check_ingress_rule(ingress_rule) {
    %ingress_rule[ CidrIp in %UNRESTRICTED_IPV4_RANGES or
                   CidrIpv6 in %UNRESTRICTED_IPV6_RANGES or
                   SourcePrefixListId exists ] {
```

```
        # Scenario 3
        IpProtocol exists
        IpProtocol in %AUTHORIZED_PROTOCOLS

        when IpProtocol in ["tcp", "udp"] {
            FromPort exists
            ToPort exists

            let ingress_block = this

            %BLOCKED_PORTS.* {
                # Scenarios 4 and 5
                check_ports(this, %ingress_block.FromPort, %ingress_block.ToPort)
            }
        }
    }
}
rule check_ports(port, FromPort, ToPort) {
    %FromPort > %port or
    %ToPort < %port
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: 0.0.0.0/0
        FromPort: 80
        ToPort: 80
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  PrefixList:
    Type: AWS::EC2::PrefixList
    Properties:
      PrefixListName:
        Fn::Sub: ${AWS::StackName}-example
      AddressFamily: IPv4
      MaxEntries: 10
      Entries:
        - Cidr: "0.0.0.0/0"
          Description: Public internet
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      SecurityGroupIngress:
      - IpProtocol: -1
        SourcePrefixListId:
          Ref: PrefixList
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      SecurityGroupIngress:
      - IpProtocol: tcp
        CidrIp: 0.0.0.0/0
        FromPort: 22
        ToPort: 22
```

[CT.EC2.PR.5] Require any Amazon EC2 network ACL to prevent ingress from 0.0.0.0/0 to port 22 or port 3389

This control checks whether the Amazon EC2 network ACL inbound entry allows unrestricted incoming traffic (`0.0.0.0/0` or `::/0`) for SSH or RDP.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::NetworkAclEntry`
- **CloudFormation guard rule:** CT.EC2.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.5 example templates

Explanation

Access to remote server administration ports, such as port 22 (SSH) and port 3389 (RDP), should not be publicly accessible, because these ports may allow unintended access to resources within your VPC.

Usage considerations

- This control only applies to Amazon EC2 network ACL entry resources that allow unrestricted inbound traffic.

Remediation for rule failure

For Amazon EC2 network ACL entries that allow inbound connectivity on port 22 or port 3389, provide a CIDR range in `CidrBlock` or `Ipv6CidrBlock` that does not allow traffic from all sources.

The examples that follow show how to implement this remediation.

Amazon EC2 Network ACL Entry - Example One

Amazon EC2 network ACL entry configured to allow unrestricted inbound IPv4 TCP traffic in a port range excluding port 22 (SSH) and port 3389 (RDP). The example is shown in JSON and in YAML.

JSON example

```
{
    "NetworkAclEntry": {
        "Type": "AWS::EC2::NetworkAclEntry",
        "Properties": {
            "CidrBlock": "0.0.0.0/0",
            "Egress": false,
            "NetworkAclId": {
                "Ref": "NACL"
            },
            "Protocol": 6,
            "PortRange": {
                "From": 2000,
                "To": 2005
            },
            "RuleAction": "allow",
            "RuleNumber": 100
```

```
        }
    }
}
```

YAML example

```
NetworkAclEntry:
  Type: AWS::EC2::NetworkAclEntry
  Properties:
    CidrBlock: '0.0.0.0/0'
    Egress: false
    NetworkAclId: !Ref 'NACL'
    Protocol: 6
    PortRange:
      From: 2000
      To: 2005
    RuleAction: allow
    RuleNumber: 100
```

The examples that follow show how to implement this remediation.

Amazon EC2 Network ACL Entry - Example Two

Amazon EC2 network ACL entry configured to allow unrestricted inbound IPv6 UDP traffic in a port range excluding port 3389 (RDP). The example is shown in JSON and in YAML.

JSON example

```
{
    "NetworkAclEntry": {
        "Type": "AWS::EC2::NetworkAclEntry",
        "Properties": {
            "Ipv6CidrBlock": "::/0",
            "Egress": false,
            "NetworkAclId": {
                "Ref": "NACL"
            },
            "Protocol": 17,
```

```
            "PortRange": {
                "From": 100,
                "To": 200
            },
            "RuleAction": "allow",
            "RuleNumber": 100
        }
    }
}
```

YAML example

```
NetworkAclEntry:
  Type: AWS::EC2::NetworkAclEntry
  Properties:
    Ipv6CidrBlock: ::/0
    Egress: false
    NetworkAclId: !Ref 'NACL'
    Protocol: 17
    PortRange:
      From: 100
      To: 200
    RuleAction: allow
    RuleNumber: 100
```

CT.EC2.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   nacl_no_unrestricted_ssh_rdp_check
#
# Description:
#   This control checks whether the Amazon EC2 network ACL inbound entry allows
 unrestricted incoming traffic ('0.0.0.0/0' or '::/0') for SSH or RDP.
#
```

```
# Reports on:
#   AWS::EC2::NetworkAclEntry
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 network ACL entry resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource has no CIDR block allowing inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource allows inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#       And: EC2 network ACL entry resource allows all traffic
#            ('IpProtocol' is set to '-1')
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource allows inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#       And: EC2 network ACL entry resource allows TCP (protocol 6) traffic
#       And: EC2 network ACL entry resource allows traffic from a PortRange that
 includes 22
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource allows inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#       And: EC2 network ACL entry resource allows TCP (protocol 6) or UDP (protocol
 17) traffic
```

```
#       And: EC2 network ACL entry resource allows traffic from a PortRange that
 includes 3389
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource allows inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#       And: EC2 network ACL entry resource allows TCP (protocol 6) traffic
#       And: EC2 network ACL entry resource allows traffic from a PortRange that
 excludes 22
#      Then: PASS
# Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a EC2 network ACL entry resource
#       And: EC2 network ACL entry resource allows inbound traffic
#            from source '0.0.0.0/0' or '::/0'
#       And: EC2 network ACL entry resource allows TCP (protocol 6) or UDP (protocol
 17) traffic
#       And: EC2 network ACL entry resource allows traffic from a PortRange that
 excludes 3389
#      Then: PASS

#
# Constants
#
let NETWORK_ACL_TYPE = "AWS::EC2::NetworkAclEntry"
let INPUT_DOCUMENT = this
let ALL_TRAFFIC_PROTOCOL = [-1, "-1"]
let TCP_PROTOCOL = [6, "6"]
let UDP_PROTOCOL = [17, "17"]
let UNRESTRICTED_IPV4_RANGES = ["0.0.0.0/0"]
let UNRESTRICTED_IPV6_RANGES = ["::/0"]
let SSH_PORT = 22
let RDP_PORT = 3389

#
# Assignments
#
let nacl_entries = Resources.*[ Type == %NETWORK_ACL_TYPE ]

#
# Primary Rules
#
```

```
rule nacl_no_unrestricted_ssh_rdp_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %nacl_entries not empty {
    check(%nacl_entries.Properties)
        <<
        [CT.EC2.PR.5]: Require any Amazon EC2 network ACL to prevent ingress from
 0.0.0.0/0 to port 22 or port 3389
        [FIX]: For Amazon EC2 network ACL entries that allow inbound connectivity on
 port 22 or port 3389, provide a CIDR range in 'CidrBlock' or 'Ipv6CidrBlock' that does
 not allow traffic from all sources.
        >>
}

rule nacl_no_unrestricted_ssh_rdp_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NETWORK_ACL_TYPE) {
    check(%INPUT_DOCUMENT.%NETWORK_ACL_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.5]: Require any Amazon EC2 network ACL to prevent ingress from
 0.0.0.0/0 to port 22 or port 3389
        [FIX]: For Amazon EC2 network ACL entries that allow inbound connectivity on
 port 22 or port 3389, provide a CIDR range in 'CidrBlock' or 'Ipv6CidrBlock' that does
 not allow traffic from all sources.
        >>
}

#
# Parameterized Rules
#
rule check(nacl_entry) {
    %nacl_entry [
        # Scenario 2
        filter_allow_unrestricted_ingress(this)
    ] {
        # Scenario 3
        Protocol exists
        Protocol not in %ALL_TRAFFIC_PROTOCOL

        # Scenario 4, 6
        check_for_open_ssh(this)

        # Scenario 5, 7
        check_for_open_rdp(this)
    }
}
```

```
rule filter_allow_unrestricted_ingress(nacl_entry) {
    Egress not exists or
    Egress != true

    CidrBlock in %UNRESTRICTED_IPV4_RANGES or
    Ipv6CidrBlock in %UNRESTRICTED_IPV6_RANGES

    RuleAction == "allow"
}

rule check_for_open_ssh(nacl_entry) {
    %nacl_entry [
        Protocol in %TCP_PROTOCOL
    ] {
        check_port_range_exists(this)
        check_ports(%SSH_PORT, PortRange.From, PortRange.To)
   }
}

rule check_for_open_rdp(nacl_entry) {
    %nacl_entry [
           Protocol in %TCP_PROTOCOL or
           Protocol in %UDP_PROTOCOL
    ] {
        check_port_range_exists(this)
        check_ports(%RDP_PORT, PortRange.From, PortRange.To)
   }
}

rule check_port_range_exists(nacl_entry) {
    PortRange exists
    PortRange is_struct
    PortRange {
        From exists
        To exists
    }
}

rule check_ports(port, nacl_from_port, nacl_to_port) {
    %nacl_from_port > %port or
    %nacl_to_port < %port
}

#
```

```
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, NETWORK_ACL_TYPE) {
    %doc.%NETWORK_ACL_TYPE.resourceProperties exists
}
```

CT.EC2.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 192.168.0.0/16
  NACL:
    Type: AWS::EC2::NetworkAcl
    Properties:
      VpcId:
        Ref: VPC
  NetworkAclEntry:
    Type: AWS::EC2::NetworkAclEntry
    Properties:
      CidrBlock: 0.0.0.0/0
      Egress: false
      NetworkAclId:
        Ref: NACL
      Protocol: 6
      PortRange:
        From: 2000
        To: 2005
```

```
      RuleAction: allow
      RuleNumber: 100
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 192.168.0.0/16
  NACL:
    Type: AWS::EC2::NetworkAcl
    Properties:
      VpcId:
        Ref: VPC
  NetworkAclEntry:
    Type: AWS::EC2::NetworkAclEntry
    Properties:
      CidrBlock: 0.0.0.0/0
      Egress: false
      NetworkAclId:
        Ref: NACL
      Protocol: 6
      PortRange:
        From: 3000
        To: 3500
      RuleAction: allow
      RuleNumber: 100
```

[CT.EC2.PR.6] Require that Amazon EC2 transit gateways refuse automatic Amazon VPC attachment requests

This control checks whether Amazon EC2 transit gateways are configured to accept Amazon VPC attachment requests automatically.

- **Control objective:** Limit network access

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::TransitGateway`
- **CloudFormation guard rule:** CT.EC2.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.6 example templates

Explanation

Turning on the `AutoAcceptSharedAttachments` property configures a transit gateway to accept cross-account VPC attachment requests automatically, without verifying the request or the account from which the attachment is originating. In alignment with the best practices of authorization and authentication, we recommended turning off this feature, to ensure that only authorized VPC attachment requests are accepted.

Remediation for rule failure

Omit the `AutoAcceptSharedAttachments` property or set the property to `disable`.

The examples that follow show how to implement this remediation.

AWS Transit Gateway - Example

AWS Transit Gateway configured to deactivate auto-acceptance of cross-account Amazon VPC attachments. The example is shown in JSON and in YAML.

JSON example

```
{
    "TransitGateway": {
        "Type": "AWS::EC2::TransitGateway",
        "Properties": {
            "AutoAcceptSharedAttachments": "disable"
        }
```

```
    }
}
```

YAML example

```
TransitGateway:
  Type: AWS::EC2::TransitGateway
  Properties:
    AutoAcceptSharedAttachments: disable
```

CT.EC2.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_transit_gateway_auto_vpc_attach_disabled_check
#
# Description:
#   This control checks whether Amazon EC2 transit gateways are configured to accept
 Amazon VPC attachment requests automatically.
#
# Reports on:
#   AWS::EC2::TransitGateway
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 transit gateway resources
#      Then: SKIP
#   Scenario: 2
```

```
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 transit gateway resource
#        And: 'AutoAcceptSharedAttachments' configuration has been provided and is set
 to a value other than 'disable'
#       Then: FAIL
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 transit gateway resource
#        And: 'AutoAcceptSharedAttachments' configuration has not been provided
#       Then: PASS
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 transit gateway resource
#        And: 'AutoAcceptSharedAttachments' configuration has been provided and set to
 'disable'
#       Then: PASS

#
# Constants
#
let EC2_TRANSIT_GATEWAY_TYPE = "AWS::EC2::TransitGateway"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_transit_gateway = Resources.*[ Type == %EC2_TRANSIT_GATEWAY_TYPE ]

#
# Primary Rules
#
rule ec2_transit_gateway_auto_vpc_attach_disabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %ec2_transit_gateway not
 empty {
    check(%ec2_transit_gateway.Properties)
        <<
        [CT.EC2.PR.6]: Require that Amazon EC2 transit gateways refuse automatic Amazon
 VPC attachment requests
            [FIX]: Omit the 'AutoAcceptSharedAttachments' property or set the property
 to 'disable'.
        >>
}
```

```
rule ec2_transit_gateway_auto_vpc_attach_disabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %EC2_TRANSIT_GATEWAY_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_TRANSIT_GATEWAY_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.6]: Require that Amazon EC2 transit gateways refuse automatic Amazon
 VPC attachment requests
            [FIX]: Omit the 'AutoAcceptSharedAttachments' property or set the property
 to 'disable'.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_transit_gateway) {
    %ec2_transit_gateway {
        # Scenario 3
        AutoAcceptSharedAttachments not exists or
        # Scenario 2 and 4
        AutoAcceptSharedAttachments == "disable"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  TransitGateway:
    Type: AWS::EC2::TransitGateway
    Properties:
      AutoAcceptSharedAttachments: disable
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  TransitGateway:
    Type: AWS::EC2::TransitGateway
    Properties:
      AutoAcceptSharedAttachments: enable
```

[CT.EC2.PR.7] Require an Amazon EBS volume resource to be encrypted at rest when defined by means of the AWS::EC2::Instance BlockDeviceMappings property or AWS::EC2::Volume resource type

This control checks whether your standalone Amazon EC2 EBS volume and Amazon Elastic Block Store (EBS) volume created through EC2 instance Block Device Mappings are encrypted at rest. Specifically, it checks that the **Encrypted** property is set to **true** in either the EBS volume resource definition or an EC2 instance resource definition's **BlockDeviceMappings** property.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`, `AWS::EC2::Volume`
- **CloudFormation guard rule:** CT.EC2.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.7 example templates

Explanation

For an added layer of security of your sensitive data in Amazon EC2 EBS volumes, you should enable EBS encryption at rest. Amazon EBS encryption offers a straightforward encryption solution for your EBS resources that doesn't require you to build, maintain, and secure your own key management infrastructure. It uses KMS keys when creating encrypted volumes and snapshots.

Amazon Elastic Block Store (EBS) volumes can be inherited from:

- The Amazon Machine Image (AMI) specified with the `ImageId` property
- The Launch Template specified with the `LaunchTemplateId` property

Usage considerations

- For Amazon EC2 instance block device mappings, this control does not check any block device mappings created by means of an EC2 launch template or inherited through the AMI from which the instance is launched.

Remediation for rule failure

Set `Encryption` to true on Amazon EC2 EBS Volumes.

The examples that follow show how to implement this remediation.

Amazon EC2 Instance - Example

Amazon EC2 instance with an encrypted EBS volume. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Instance": {
        "Type": "AWS::EC2::Instance",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "t3.micro",
            "NetworkInterfaces": [
                {
                    "DeviceIndex": 0,
                    "SubnetId": {
                        "Ref": "Subnet"
                    },
                    "AssociatePublicIpAddress": false
                }
            ],
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sdm",
                    "Ebs": {
                        "VolumeType": "gp3",
                        "Iops": 200,
                        "Encrypted": true,
                        "DeleteOnTermination": true,
                        "VolumeSize": 20
                    }
                }
            ]
        }
    }
}
```

YAML example

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: t3.micro
    NetworkInterfaces:
```

```
      - DeviceIndex: 0
        SubnetId: !Ref 'Subnet'
        AssociatePublicIpAddress: false
    BlockDeviceMappings:
      - DeviceName: /dev/sdm
        Ebs:
          VolumeType: gp3
          Iops: 200
          Encrypted: true
          DeleteOnTermination: true
          VolumeSize: 20
```

The examples that follow show how to implement this remediation.

Amazon EBS Volume - Example

Amazon EBS Volume with encryption configured. The example is shown in JSON and in YAML.

JSON example

```
{
    "EBSVolume": {
        "Type": "AWS::EC2::Volume",
        "Properties": {
            "Size": 100,
            "AvailabilityZone": {
                "Fn::Select": [
                    0,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
            "Encrypted": true
        }
    }
}
```

YAML example

```
EBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 100
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    Encrypted: true
```

CT.EC2.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ec2_encrypted_volumes_check
#
# Description:
#   Checks whether standalone Amazon EC2 EBS volumes and new EC2 EBS volumes created
 through EC2 instance
#   Block Device Mappings are encrypted at rest.
#
# Reports on:
#   AWS::EC2::Instance, AWS::EC2::Volume
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 volume resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an EC2 instance resource
#       And: 'BlockDeviceMappings' has not been provided or has been provided as an
 empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: 'Ebs' has been provided in a 'BlockDeviceMappings' configuration
#       And: 'Encrypted' has not been provided in the 'Ebs' configuration
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: 'Ebs' has been provided in a 'BlockDeviceMappings' configuration
#       And: 'Encrypted' has been provided in the 'Ebs' configuration and set to
 bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 volume resource
#       And: 'Encrypted' on the EC2 volume has not been provided
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 volume resource
#       And: 'Encrypted' on the EC2 volume has been provided and is set to bool(false)
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: 'Ebs' has been provided in a 'BlockDeviceMappings' configuration
#       And: 'Encrypted' has been provided in the 'Ebs' configuration and set to
 bool(true)
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 volume resource
#       And: 'Encrypted' on the EC2 volume has been provided and is set to bool(true)
#      Then: PASS

#
```

```
# Constants
#
let EC2_VOLUME_TYPE = "AWS::EC2::Volume"
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_volumes = Resources.*[ Type == %EC2_VOLUME_TYPE ]
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
# Primary Rules
#
rule ec2_encrypted_volumes_check when is_cfn_template(%INPUT_DOCUMENT)
                                      %ec2_volumes not empty {
    check_volume(%ec2_volumes.Properties)
        <<
        [CT.EC2.PR.7]: Require that an Amazon EBS volume attached to an Amazon EC2
 instance is encrypted at rest
        [FIX]: Set 'Encryption' to true on EC2 EBS Volumes.
        >>
}

rule ec2_encrypted_volumes_check when is_cfn_hook(%INPUT_DOCUMENT, %EC2_VOLUME_TYPE) {
    check_volume(%INPUT_DOCUMENT.%EC2_VOLUME_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.7]: Require that an Amazon EBS volume attached to an Amazon EC2
 instance is encrypted at rest
        [FIX]: Set 'Encryption' to true on EC2 EBS Volumes.
        >>
}

rule ec2_encrypted_volumes_check when is_cfn_template(%INPUT_DOCUMENT)
                                      %ec2_instances not empty {
    check_instance(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.7]: Require that an Amazon EBS volume attached to an Amazon EC2
 instance is encrypted at rest
        [FIX]: Set 'Encryption' to true on EC2 EBS Volumes.
        >>
}
```

```
rule ec2_encrypted_volumes_check when is_cfn_hook(%INPUT_DOCUMENT, %EC2_INSTANCE_TYPE)
 {
    check_instance(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.7]: Require that an Amazon EBS volume attached to an Amazon EC2
 instance is encrypted at rest
        [FIX]: Set 'Encryption' to true on EC2 EBS Volumes.
        >>
}

#
# Parameterized Rules
#

rule check_instance(ec2_instance) {
    %ec2_instance[
        filter_ec2_instance_block_device_mappings(this)
    ] {
        BlockDeviceMappings[
            Ebs exists
            Ebs is_struct
        ] {
            check_volume(Ebs)
        }
    }
}

rule check_volume(ec2_volume) {
    %ec2_volume {
        # Scenario 2
        Encrypted exists
        # Scenarios 3 and 4
        Encrypted == true
    }
}

rule filter_ec2_instance_block_device_mappings(ec2_instance) {
    %ec2_instance {
        BlockDeviceMappings exists
        BlockDeviceMappings is_list
        BlockDeviceMappings not empty
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
```

```
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId:
          Ref: Subnet
        AssociatePublicIpAddress: false
      BlockDeviceMappings:
      - DeviceName: "/dev/sdm"
        Ebs:
          VolumeType: gp3
          Iops: 200
          Encrypted: true
          DeleteOnTermination: true
          VolumeSize: 20
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId:
          Ref: Subnet
        AssociatePublicIpAddress: false
      BlockDeviceMappings:
      - DeviceName: "/dev/sdm"
        Ebs:
          VolumeType: gp3
          Iops: 200
          Encrypted: false
          DeleteOnTermination: true
          VolumeSize: 20
```

[CT.EC2.PR.8] Require an Amazon EC2 instance to set AssociatePublicIpAddress to false on a new network interface created by means of the NetworkInterfaces property in the AWS::EC2::Instance resource

This control checks whether your Amazon EC2 instance is configured **not** to associate a public IP address by default. In particular, this control requires configuring the **AssociatePublicIpAddress** parameter to **false** on a new network interface created by means of the **NetworkInterfaces** property.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`
- **CloudFormation guard rule:** CT.EC2.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.8 example templates

> **This control is incompatible with AWS Cloud9**
>
> A compatibility issue exists with AWS Cloud9 and this AWS Control Tower proactive control, [CT.EC2.PR.8] Require an Amazon EC2 instance to set **AssociatePublicIpAddress** to **false** on a new network interface created by means of the **NetworkInterfaces** property in the **AWS::EC2::Instance** resource. If this control is enabled, you cannot create an Amazon EC2 environment in AWS Cloud9. For more information, see Troubleshooting AWS Cloud9.

Explanation

A public IPv4 address is an IP address that is reachable from the internet. If you launch your instance with a public IP address, then your EC2 instance is reachable from the internet. A private IPv4 address is an IP address that is not reachable from the internet. You can use private IPv4 addresses for communication between EC2 instances in the same VPC or in your connected private network.

IPv6 addresses are globally unique, and therefore are reachable from the internet. However, by default all subnets have the IPv6 addressing attribute set to false.

The network interface settings can be inherited from the Launch Template specified with the **LaunchTemplateId** property.

Usage considerations

- This control applies only to a new network interface created by means of the **NetworkInterfaces** property, where a **NetworkInterfaceId** has not been specified.
- This control requires subnet information to be specified within a `NetworkInterfaces` configuration instead of the root level `SubnetId` property.
- This control does not check a network interface that may be created in an Amazon EC2 launch template that may be referenced by the **LaunchTemplateId** property.
- A compatibility issue exists with AWS Cloud9 and this AWS Control Tower proactive control. If this control is enabled, you cannot create an Amazon EC2 environment in AWS Cloud9.

Remediation for rule failure

Specify network interfaces using the `NetworkInterfaces` property instead of the root level `SubnetId` property. Set `AssociatePublicIpAddress` to false within each `NetworkInterfaces` configuration.

The examples that follow show how to implement this remediation.

Amazon EC2 Instance - Example

Amazon EC2 instance configured with a new interface that disables public IP address association on creation. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Instance": {
        "Type": "AWS::EC2::Instance",
        "Properties": {
            "InstanceType": "t3.micro",
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "NetworkInterfaces": [
                {
                    "DeviceIndex": 0,
```

```
                    "SubnetId": {
                        "Ref": "Subnet"
                    },
                    "AssociatePublicIpAddress": false
                }
            ]
        }
    }
}
```

YAML example

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    ImageId: !Ref 'LatestAmiId'
    NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId: !Ref 'Subnet'
        AssociatePublicIpAddress: false
```

CT.EC2.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ec2_instance_no_public_ip_check
#
# Description:
#   This control checks whether your Amazon EC2 instance is configured to associate a
 public IP address.
#
# Reports on:
#   AWS::EC2::Instance
#
```

```
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is not present on the EC2 instance resource or is an
 empty list
#       And: 'SubnetId' is not provided as a top-level resource property
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is present on the EC2 instance resource as a non-empty
 list
#       And: 'NetworkInterfaceId' is present for a configuration in 'NetworkInterfaces'
 and is a non-empty string or
#             valid local reference
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is not provided
#       And: 'SubnetId' is provided as a top-level resource property
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is present on the EC2 instance resource with one or
 more configurations
#       And: 'NetworkInterfaceId' is not present or is present and and is an empty
 string or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is not present for a configuration in
 'NetworkInterfaces'
#      Then: FAIL
```

```
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is present on the EC2 instance resource
#       And: 'NetworkInterfaceId' is not present or is present and and is an empty
 string or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is present for a configuration in
 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is set to bool(true)
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'NetworkInterfaces' is present on the EC2 instance resource
#       And: 'NetworkInterfaceId' is not present or is present and and is an empty
 string or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is present for a configuration in
 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is set to bool(false)
#      Then: PASS

#
# Constants
#
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
# Primary Rules
#
rule ec2_instance_no_public_ip_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %ec2_instances not empty {
    check(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.8]: Require any Amazon EC2 instance to have a non-public IP address
```

```
        [FIX]: Specify network interfaces using the 'NetworkInterfaces' property
 instead of the root level 'SubnetId' property. Set 'AssociatePublicIpAddress' to false
 within each 'NetworkInterfaces' configuration.
        >>
}

rule ec2_instance_no_public_ip_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.8]: Require any Amazon EC2 instance to have a non-public IP address
        [FIX]: Specify network interfaces using the 'NetworkInterfaces' property
 instead of the root level 'SubnetId' property. Set 'AssociatePublicIpAddress' to false
 within each 'NetworkInterfaces' configuration.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_instance) {
    %ec2_instance[ SubnetId exists ] {
        # Scenario 5
        SubnetId not exists
    }

    %ec2_instance[
        # Scenario 2
        NetworkInterfaces exists
        NetworkInterfaces is_list
        NetworkInterfaces not empty
    ] {
        NetworkInterfaces[
            # Scenario 3 and 4
            filter_network_interfaces(this)
        ] {
            # Scenario 6
            AssociatePublicIpAddress exists
            # Scenarios 7 and 8
            AssociatePublicIpAddress == false
        }
    }
}
```

```
rule filter_network_interfaces(network_interface) {
    %network_interface {
        NetworkInterfaceId not exists or
        filter_property_is_empty_string(NetworkInterfaceId) or
        filter_exclude_valid_local_reference(%INPUT_DOCUMENT, NetworkInterfaceId,
 "AWS::EC2::NetworkInterface")
     }
}

rule filter_property_is_empty_string(value) {
    %value {
        this is_string
        this == /\A\s*\z/
    }
}

rule filter_exclude_valid_local_reference(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        this not is_string
        this is_struct

        when this.'Fn::GetAtt' exists {
            'Fn::GetAtt' {
                when filter_query_template_resources(%doc, this[0],
 %referenced_resource_type) {
                    this not exists
                }
                this exists
            }
        }
        when this.'Fn::GetAtt' not exists {
            this exists
        }
    }
}

rule filter_query_template_resources(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type in %referenced_resource_type
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t3.micro
      ImageId:
        Ref: LatestAmiId
      NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId:
          Ref: Subnet
        AssociatePublicIpAddress: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
```

```
        - Fn::GetAZs: ''
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t3.micro
      ImageId:
        Ref: LatestAmiId
      SubnetId:
        Ref: Subnet
```

[CT.EC2.PR.9] Require any Amazon EC2 launch template not to auto-assign public IP addresses to network interfaces

This control checks whether your Amazon EC2 launch templates are configured to assign public IP addresses to network interfaces.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.9 example templates

Explanation

A public IP address is an IP address that is reachable from the internet. If you configure your network interfaces with a public IP address, then the resources associated to those network interfaces are reachable from the internet. EC2 resources should not be publicly accessible, because this may allow unintended access to your application servers.

Usage considerations

- This control applies only to new network interfaces created by means of the `NetworkInterfaceId` property in `LaunchTemplateData` (`NetworkInterfaces` configurations where a `NetworkInterfaceId` has not been specified).
- This control requires setting `AssociatePublicIpAddress` to `false` on new network interfaces created by means of the `NetworkInterfaces` property in `LaunchTemplateData`.

Remediation for rule failure

Set `AssociatePublicIpAddress` to `false` within each `NetworkInterfaces` configuration in `LaunchTemplateData`.

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example

Amazon EC2 launch template configured with a network interface that disables public IP address association. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "NetworkInterfaces": [
                    {
                        "DeviceIndex": 0,
                        "SubnetId": {
                            "Ref": "Subnet"
                        },
                        "AssociatePublicIpAddress": false
                    }
                ]
            }
        }
```

```
    }
}
```

YAML example

```
EC2LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      NetworkInterfaces:
        - DeviceIndex: 0
          SubnetId: !Ref 'Subnet'
          AssociatePublicIpAddress: false
```

CT.EC2.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_launch_template_public_ip_disabled_check
#
# Description:
#   This control checks whether your Amazon EC2 launch templates are configured to
 assign public IP addresses to network interfaces.
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 launch template
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 launch template resource
#       And: 'NetworkInterfaces' is not provided in 'LaunchTemplateData'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 launch template resource
#       And: 'LaunchTemplateData.NetworkInterfaces' is present on the Amazon EC2 launch
 template resource as a non empty list
#       And: 'NetworkInterfaceId' is present for a configuration in 'NetworkInterfaces'
 and is a non-empty string or
#             valid local reference
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 launch template resource
#       And: 'LaunchTemplateData.NetworkInterfaces' is present on the Amazon EC2 launch
 template resource
#       And: 'NetworkInterfaceId' is not present or is present and is an empty string
 or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is not present for a configuration in
 'NetworkInterfaces'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 launch template resource
#       And: 'LaunchTemplateData.NetworkInterfaces' is present on the Amazon EC2 launch
 template resource
#       And: 'NetworkInterfaceId' is not present or is present and is an empty string
 or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is present for a configuration in
 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is set to bool(true)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 launch template resource
```

```
#       And: 'LaunchTemplateData.NetworkInterfaces' is present on the Amazon EC2 launch
 template resource
#       And: 'NetworkInterfaceId' is not present or is present and is an empty string
 or invalid local reference for
#            a configuration in 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is present for a configuration in
 'NetworkInterfaces'
#       And: 'AssociatePublicIpAddress' is set to bool(false)
#      Then: PASS

#
# Constants
#
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_public_ip_disabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %ec2_launch_templates not empty
 {
    check(%ec2_launch_templates.Properties)
        <<
        [CT.EC2.PR.9]: Require any Amazon EC2 launch template not to auto-assign public
 IP addresses to network interfaces
        [FIX]: Set 'AssociatePublicIpAddress' to 'false' within each
 'NetworkInterfaces' configuration in 'LaunchTemplateData'.
        >>
}

rule ec2_launch_template_public_ip_disabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.9]: Require any Amazon EC2 launch template not to auto-assign public
 IP addresses to network interfaces
        [FIX]: Set 'AssociatePublicIpAddress' to 'false' within each
 'NetworkInterfaces' configuration in 'LaunchTemplateData'.
```

```
        >>
}

#
# Parameterized Rules
#
rule check(ec2_launch_templates) {
    %ec2_launch_templates[
        # Scenario 2
        filter_launch_template(this)
    ] {
        LaunchTemplateData {
            NetworkInterfaces[
                # Scenario 3 and 4
                filter_network_interfaces(this)
            ] {
                # Scenario 5 and 6
                AssociatePublicIpAddress exists
                AssociatePublicIpAddress == false
            }
        }
    }
}

rule filter_launch_template(ec2_launch_template) {
    %ec2_launch_template {
        LaunchTemplateData exists
        LaunchTemplateData is_struct
        LaunchTemplateData {
            NetworkInterfaces exists
            NetworkInterfaces is_list
            NetworkInterfaces not empty
        }
    }
}

rule filter_network_interfaces(network_interface) {
    %network_interface {
        NetworkInterfaceId not exists or
        filter_property_is_empty_string(NetworkInterfaceId) or
        filter_exclude_valid_local_reference(%INPUT_DOCUMENT, NetworkInterfaceId,
 "AWS::EC2::NetworkInterface")
    }
}
```

```
rule filter_property_is_empty_string(value) {
    %value {
        this is_string
        this == /\A\s*\z/
    }
}

rule filter_exclude_valid_local_reference(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        this not is_string
        this is_struct

        when this.'Fn::GetAtt' exists {
            'Fn::GetAtt' {
                when query_for_resource(%doc, this[0], %referenced_resource_type) {
                    this not exists
                }
                this exists
            }
        }
        when this.'Fn::GetAtt' not exists {
            this exists
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
```

```
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.EC2.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        NetworkInterfaces:
        - DeviceIndex: 0
          SubnetId:
            Ref: Subnet
          AssociatePublicIpAddress: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        NetworkInterfaces:
        - DeviceIndex: 0
          SubnetId:
            Ref: Subnet
          AssociatePublicIpAddress: true
```

[CT.EC2.PR.10] Require Amazon EC2 launch templates to have Amazon CloudWatch detailed monitoring activated

This control checks whether the Amazon EC2 launch template has detailed monitoring enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.10 example templates

Explanation

Monitoring is an important part of maintaining the reliability, availability, and performance of your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. From a security perspective, logging is also an important feature to enable for future forensics efforts in the case of any security incidents.

Remediation for rule failure

In `LaunchTemplateData`, provide a `Monitoring` configuration with `Enabled` set to `true`.

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example

Amazon EC2 launch template configured with detailed monitoring enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "Monitoring": {
                    "Enabled": true
```

```
                }
            }
        }
    }
}
```

YAML example

```
EC2LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      Monitoring:
        Enabled: true
```

CT.EC2.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_launch_template_monitoring_enabled_check
#
# Description:
#   This control checks whether the Amazon EC2 launch template has detailed monitoring
 enabled.
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any EC2 launch template resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.Monitoring.Enabled' has not been provided or has been
 provided and is empty.
#       Then: FAIL
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.Monitoring.Enabled' has been provided
#        And: 'LaunchTemplateData.Monitoring.Enabled' is equal to a value other than
 bool(true)
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an EC2 launch template resource
#        And: 'LaunchTemplateData.Monitoring.Enabled' has been provided
#        And: 'LaunchTemplateData.Monitoring.Enabled' is equal to bool(true)
#       Then: PASS

#
# Constants
#
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_monitoring_enabled_check when is_cfn_template(this)
                                                       %ec2_launch_templates not empty
 {
    check(%ec2_launch_templates.Properties)
        <<
```

```
        [CT.EC2.PR.10]: Require Amazon EC2 launch templates to have Amazon CloudWatch
 detailed monitoring activated
        [FIX]: In 'LaunchTemplateData', provide a 'Monitoring' configuration with
 'Enabled' set to 'true'.
        >>
}

rule ec2_launch_template_monitoring_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.10]: Require Amazon EC2 launch templates to have Amazon CloudWatch
 detailed monitoring activated
        [FIX]: In 'LaunchTemplateData', provide a 'Monitoring' configuration with
 'Enabled' set to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_launch_template) {
    %ec2_launch_template {
        # Scenario 2
        LaunchTemplateData exists
        LaunchTemplateData is_struct

        LaunchTemplateData {
            Monitoring exists
            Monitoring is_struct

            # Scenario 3 and 4
            Monitoring {
                Enabled exists
                Enabled == true
            }

        }
    }
}

#
# Utility Rules
#
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        Monitoring:
          Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        Monitoring:
          Enabled: false
```

[CT.EC2.PR.11] Require that an Amazon EC2 subnet does not automatically assign public IP addresses

This control checks whether your Amazon VPC subnets assign public IP addresses automatically.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Subnet`
- **CloudFormation guard rule:** CT.EC2.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.11 example templates

Explanation

All subnets have an attribute that determines whether a network interface created in the subnet automatically receives a public IPv4 address. When launched into subnets that have this attribute enabled, instances receive a public IP address assigned to their primary network interface.

Usage considerations

- This control deactivates automatic assignment of public IP addresses for new network interfaces in Amazon VPC subnets.
- When this control is in operation, public IP addresses can be assigned to network interfaces by means of resource-level settings. (For example, assignment of a public IP address can be made at EC2 instance launch time.)

Remediation for rule failure

Omit the `MapPublicIpOnLaunch` property to use the default configuration, or set the `MapPublicIpOnLaunch` property to `false`.

The examples that follow show how to implement this remediation.

Amazon VPC Subnet - Example One

Amazon VPC subnet configured to deactivate automatic assignment of public IP addresses by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "Subnet": {
        "Type": "AWS::EC2::Subnet",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "CidrBlock": "10.0.0.0/24",
            "AvailabilityZone": {
                "Fn::Select": [
                    0,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            }
        }
    }
}
```

YAML example

```
Subnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref 'VPC'
```

```
    CidrBlock: 10.0.0.0/24
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
```

The examples that follow show how to implement this remediation.

Amazon VPC Subnet - Example Two

Amazon VPC subnet configured to deactivate automatic assignment of public IP addresses by means of the `MapPublicIpOnLaunch` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "Subnet": {
        "Type": "AWS::EC2::Subnet",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "CidrBlock": "10.0.0.0/24",
            "AvailabilityZone": {
                "Fn::Select": [
                    0,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
            "MapPublicIpOnLaunch": false
        }
    }
}
```

YAML example

```
Subnet:
```

```
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref 'VPC'
    CidrBlock: 10.0.0.0/24
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    MapPublicIpOnLaunch: false
```

CT.EC2.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
#
# Rule Identifier:
#   subnet_auto_assign_public_ip_disabled_check
#
# Description:
#   This control checks whether your Amazon VPC subnets automatically assign public IP
 addresses.
#
# Reports on:
#   AWS::EC2::Subnet
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 subnet resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 subnet resource
```

```
#       And: 'MapPublicIpOnLaunch' is present and set to bool(true)
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 subnet resource
#       And: 'MapPublicIpOnLaunch' is not present
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 subnet resource
#       And: 'MapPublicIpOnLaunch' is present and set to bool(false)
#      Then: PASS

#
# Constants
#
let EC2_SUBNET_TYPE  = "AWS::EC2::Subnet"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_subnets = Resources.*[ Type == %EC2_SUBNET_TYPE ]

#
# Primary Rules
#
rule subnet_auto_assign_public_ip_disabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %ec2_subnets not empty {
    check(%ec2_subnets.Properties)
        <<
        [CT.EC2.PR.11]: Require that an Amazon EC2 subnet does not automatically assign
 public IP addresses
        [FIX]: Omit the 'MapPublicIpOnLaunch' property to use the default
 configuration, or set the 'MapPublicIpOnLaunch' property to 'false'.
        >>
}

rule subnet_auto_assign_public_ip_disabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_SUBNET_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_SUBNET_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.11]: Require that an Amazon EC2 subnet does not automatically assign
 public IP addresses
```

```
        [FIX]: Omit the 'MapPublicIpOnLaunch' property to use the default
 configuration, or set the 'MapPublicIpOnLaunch' property to 'false'.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_subnet) {
    %ec2_subnet {
        # Scenario 3
        MapPublicIpOnLaunch not exists or
        # Scenarios 2 and 4
        MapPublicIpOnLaunch == false
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
```

```
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
      MapPublicIpOnLaunch: true
```

[CT.EC2.PR.12] Require an Amazon EC2 instance to specify at most one network interface by means of the NetworkInterfaces property in the AWS::EC2::Instance resource

This control checks whether your Amazon Elastic Compute Cloud (Amazon EC2) instance uses multiple ENIs (Elastic Network Interfaces). Specifically, it checks whether an **AWS::EC2::Instance** resource specifies multiple ENIs in the **NetworkInterfaces** property.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`
- **CloudFormation guard rule:** CT.EC2.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.EC2.PR.12 example templates

Explanation

Multiple ENIs can cause dual-homed instances, meaning instances that have multiple subnets. This duplication can add network security complexity and introduce unintended network paths and access.

The network interface settings can be inherited from the Launch Template specified with the **LaunchTemplateId** property.

Usage considerations

- This control does not check a network interface that may be specified in an Amazon EC2 launch template and referenced by the **LaunchTemplateId** property.
- This rule is incompatible with scenarios in which the **NetworkInterfaces** property must be used to specify multiple ENIs. For example, this control may fail if an Amazon EC2

instance that belongs to an Amazon EKS cluster specifies more than one ENI by means of the **NetworkInterfaces** property.

Remediation for rule failure

Configure Amazon EC2 instances with only one ENI.

The examples that follow show how to implement this remediation.

Amazon EC2 Instance - Example

EC2 Instance with a single network interface. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Instance": {
        "Type": "AWS::EC2::Instance",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "NetworkInterfaces": [
                {
                    "SubnetId": {
                        "Ref": "TestSubnet"
                    },
                    "DeviceIndex": 0
                }
            ]
        }
    }
}
```

YAML example

```
EC2Instance:
  Type: AWS::EC2::Instance
```

```
  Properties:
    ImageId: !Ref 'LatestAmiId'
    NetworkInterfaces:
      - SubnetId: !Ref 'TestSubnet'
        DeviceIndex: 0
```

CT.EC2.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_instance_multiple_eni_check
#
# Description:
#   Checks whether Amazon Elastic Compute Cloud (Amazon EC2) instances use multiple
 ENIs (Elastic Network Interfaces)
#   or Elastic Fabric Adapters (EFAs).
#
# Reports on:
#   AWS::EC2::Instance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'NetworkInterfaces' is not present or is present and contains 0
 configurations
#      Then: SKIP
```

```
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'NetworkInterfaces' is present and contains >1 configurations
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'NetworkInterfaces' is present
#       And: 'NetworkInterfaces' is present and contains 1 configuration
#      Then: PASS

#
# Constants
#
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
# Primary Rules
#
rule ec2_instance_multiple_eni_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %ec2_instances not empty {
    check(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.12]: Require an Amazon EC2 instance to configure one ENI only
        [FIX]: Configure Amazon EC2 instances with only one ENI.
        >>

}

rule ec2_instance_multiple_eni_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_INSTANCE_TYPE) {

    check(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.12]: Require an Amazon EC2 instance to configure one ENI only
        [FIX]: Configure Amazon EC2 instances with only one ENI.
        >>
```

```
}

#
# Parameterized Rules
#
rule check(ec2_instance) {
    %ec2_instance [
        # Scenario 2
        NetworkInterfaces exists
        NetworkInterfaces is_list
        NetworkInterfaces not empty
    ] {
        # Scenario 3 and 4
        NetworkInterfaces[1] not exists
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
```

```
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - SubnetId:
          Ref: Subnet
        DeviceIndex: 0
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
```

```
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - SubnetId:
          Ref: Subnet
        DeviceIndex: 0
      - SubnetId:
          Ref: Subnet
        DeviceIndex: 1
```

[CT.EC2.PR.13] Require an Amazon EC2 instance to have detailed monitoring enabled

This control checks whether an Amazon EC2 instance has detailed monitoring enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`
- **CloudFormation guard rule:** CT.EC2.PR.13 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.13 rule specification

- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.EC2.PR.13 example templates

Explanation

By default, all Amazon EC2 instances are created with basic monitoring that sends host-level logs to Amazon CloudWatch every five (5) minutes. With detailed monitoring, host-level logs are collected every one (1) minute instead, leading to faster detection of possible malicious or anomalous activity.

Usage considerations

- When you enable detailed monitoring, you are charged per metric that is sent to CloudWatch. You are not charged for data storage. For more information, see the Amazon CloudWatch pricing page.

Remediation for rule failure

Set `Monitoring` to `true`.

The examples that follow show how to implement this remediation.

EC2 Instance - Example

An EC2 Instance with detailed monitoring enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "Parameters": {
        "LatestAmiId": {
            "Description": "Region specific latest AMI ID from the Parameter Store",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
        }
    },
    "Resources": {
        "VPC": {
```

```
            "Type": "AWS::EC2::VPC",
            "Properties": {
                "CidrBlock": "10.0.0.0/16",
                "EnableDnsSupport": "true",
                "EnableDnsHostnames": "true"
            }
        },
        "Subnet": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": "10.0.0.0/24"
            }
        },
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": {
                    "Ref": "LatestAmiId"
                },
                "InstanceType": "t3.micro",
                "NetworkInterfaces": [
                    {
                        "SubnetId": {
                            "Ref": "Subnet"
                        },
                        "DeviceIndex": 0
                    }
                ],
                "Monitoring": true
            }
        }
    }
}
```

YAML example

```
Parameters:
  LatestAmiId:
```

```
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref 'VPC'
      CidrBlock: 10.0.0.0/24
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref 'LatestAmiId'
      InstanceType: t3.micro
      NetworkInterfaces:
        - SubnetId: !Ref 'Subnet'
          DeviceIndex: 0
      Monitoring: true
```

CT.EC2.PR.13 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ec2_instance_detailed_monitoring_enabled_check
#
# Description:
#   This control checks whether an Amazon EC2 instance has detailed monitoring enabled.
#
# Reports on:
#   AWS::EC2::Instance
#
```

```
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'Monitoring' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 instance resource
#       And: 'Monitoring' has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains an EC2 instance resource
#       And: 'Monitoring' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
# Primary Rules
#
rule ec2_instance_detailed_monitoring_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %ec2_instances not empty {
```

```
    check(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.13]: Require an Amazon EC2 instance to have detailed monitoring
 enabled
        [FIX]: Set 'Monitoring' to 'true'.
        >>

}

rule ec2_instance_detailed_monitoring_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.13]: Require an Amazon EC2 instance to have detailed monitoring
 enabled
        [FIX]: Set 'Monitoring' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_instance) {
    %ec2_instance {
        # Scenario 2
        Monitoring exists
        # Scenarios 3 and 4
        Monitoring == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.13 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - SubnetId:
          Ref: Subnet
        DeviceIndex: 0
      Monitoring: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      NetworkInterfaces:
      - SubnetId:
          Ref: Subnet
        DeviceIndex: 0
      Monitoring: false
```

[CT.EC2.PR.14] Require an Amazon EBS volume configured through an Amazon EC2 launch template to encrypt data at rest

This control checks whether an Amazon EC2 launch template with EBS volume block device mappings is configured to enable EBS volume encryption.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.14 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.14 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.14 example templates

Explanation

For an added layer of security of your sensitive data in an EBS volume, you should enable EBS encryption at rest. Amazon EBS encryption offers a straightforward encryption solution for your EBS resources. It doesn't require you to build, maintain, and secure your own key management infrastructure, and it uses KMS keys when creating encrypted volumes and snapshots.

Usage considerations

- This control applies only to an EC2 launch template that specifies EBS block device mappings.
- When you launch an instance using a launch template, you can override parameters that are specified in the launch template. To ensure that encryption is enabled for EBS block device mappings when you launch an instance with a launch template by means of the `AWS::EC2::Instance` resource, use this control in conjunction with `CT.EC2.PR.7`.

Remediation for rule failure

For every entry in the `BlockDeviceMappings` parameter with an `Ebs` configuration, set `Encryption` to true.

The examples that follow show how to implement this remediation.

Amazon EC2 launch template - Example

An Amazon EC2 launch template configured with an EBS block device mapping that has volume encyrption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdc",
                        "Ebs": {
                            "Encrypted": true
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      BlockDeviceMappings:
        - DeviceName: /dev/sdc
          Ebs:
            Encrypted: true
```

CT.EC2.PR.14 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_launch_template_encrypted_volumes_check
#
# Description:
#   This control checks whether an Amazon EC2 launch template with EBS volume block
 device mappings is configured to enable EBS volume encryption.
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 launch template resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'BlockDeviceMappings' in 'LaunchTemplateData' has not been provided or has
#            been provided as an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'BlockDeviceMappings' in 'LaunchTemplateData' been provided as a non-empty
 list
#       And: No entries in 'BlockDeviceMappings' contain 'Ebs' as a struct
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
```

```
#       And: 'BlockDeviceMappings' in 'LaunchTemplateData' been provided as a non-empty
 list
#       And: An entry in 'BlockDeviceMappings' contains 'Ebs' as a struct
#       And: In the same entry, 'Encrypted' in 'Ebs' has not been provided or has been
 provided
#            and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EC2 launch template resource
#       And: 'BlockDeviceMappings' in 'LaunchTemplateData' been provided as a non-empty
 list
#       And: An entry in 'BlockDeviceMappings' contains 'Ebs' as a struct
#       And: In the same entry, 'Encrypted' in 'Ebs' has not been provided or has been
 provided
#            and set to bool(true)
#      Then: PASS

#
# Constants
#
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_encrypted_volumes_check when is_cfn_template(this)
                                                      %ec2_launch_templates not empty {
    check(%ec2_launch_templates.Properties)
        <<
        [CT.EC2.PR.14]: Require an Amazon EBS volume configured through an Amazon EC2
 launch template to encrypt data at rest
        [FIX]: For every entry in the 'BlockDeviceMappings' parameter with an 'Ebs'
 configuration, set 'Encryption' to true.
        >>
}
```

```
rule ec2_launch_template_encrypted_volumes_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.14]: Require an Amazon EBS volume configured through an Amazon EC2
 launch template to encrypt data at rest
        [FIX]: For every entry in the 'BlockDeviceMappings' parameter with an 'Ebs'
 configuration, set 'Encryption' to true.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_launch_template) {
    %ec2_launch_template [
        # Scenarios 2 and 3
        filter_launch_template_contains_ebs_block_device_mappings(this)
    ] {
        LaunchTemplateData {
            BlockDeviceMappings[
                Ebs exists
                Ebs is_struct
            ] {
                Ebs {
                    # Scenarios 4 and 5
                    Encrypted exists
                    Encrypted == true
                }
            }
        }
    }
}

rule filter_launch_template_contains_ebs_block_device_mappings(launch_template) {
    %launch_template {
        LaunchTemplateData exists
        LaunchTemplateData is_struct

        LaunchTemplateData {
            BlockDeviceMappings exists
            BlockDeviceMappings is_list
            BlockDeviceMappings not empty
```

```
            some BlockDeviceMappings[*] {
                Ebs exists
                Ebs is_struct
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.14 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        BlockDeviceMappings:
        - DeviceName: /dev/sdc
          Ebs:
            Encrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        BlockDeviceMappings:
        - DeviceName: /dev/sdc
          Ebs:
            Encrypted: false
```

[CT.EC2.PR.15] Require an Amazon EC2 instance to use an AWS Nitro instance type when creating from the 'AWS::EC2::LaunchTemplate' resource type

This control checks whether Amazon EC2 launch templates that specify an Amazon EC2 instance type or use attribute based instance selection, specify only AWS Nitro instance types.

- **Control objective:** Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::LaunchTemplate`
- **CloudFormation guard rule:** CT.EC2.PR.15 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.15 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.15 example templates

Explanation

The AWS Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security that continuously monitors, protects, and verifies the instance hardware and firmware. AWS Nitro offloads virtualization resources to dedicated hardware and software, which minimizes the attack surface. Finally, the Nitro System has a locked down security model to prohibit administrative access, eliminating the possibility of human error and tampering.

For information about Nitro instance types, see Instances built on the Nitro System in the *Amazon EC2 User Guide for Linux Instances*.

Usage considerations

- This control applies only to launch templates that specify an Amazon EC2 instance type by means of the InstanceType property or use attribute based instance selection by means of the InstanceRequirements property.
- When you launch an instance using a launch template, you can override parameters that are specified in the launch template. To launch instances with a Nitro instance type when using a launch template, use this control in conjunction with related proactive controls.

Remediation for rule failure

When InstanceType in LaunchTemplateData has been provided, set InstanceType to an Amazon EC2 instance type that is based on the AWS Nitro system. When InstanceRequirements in LaunchTemplateData has been provided, set AllowedInstanceTypes to a list of Amazon EC2 instance types based on the AWS Nitro system.

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example One

An Amazon EC2 launch template configured with an instance type based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
```

```
        "Properties": {
            "LaunchTemplateData": {
                "InstanceType": "t3.micro"
            }
        }
    }
}
```

YAML example

```
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      InstanceType: t3.micro
```

The examples that follow show how to implement this remediation.

Amazon EC2 Launch Template - Example Two

An Amazon EC2 launch template configured with an instance requirements configuration that includes allowed instances based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "LaunchTemplate": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateData": {
                "InstanceRequirements": {
                    "AllowedInstanceTypes": [
                        "m5.*",
                        "c5.*"
                    ],
                    "VCpuCount": {
                        "Max": 16,
```

```
                        "Min": 1
                    },
                    "MemoryMiB": {
                        "Min": 1024,
                        "Max": 17000
                    }
                }
            }
        }
    }
}
```

YAML example

```
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateData:
      InstanceRequirements:
        AllowedInstanceTypes:
          - m5.*
          - c5.*
        VCpuCount:
          Max: 16
          Min: 1
        MemoryMiB:
          Min: 1024
          Max: 17000
```

CT.EC2.PR.15 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ec2_launch_template_nitro_instance_type_check
#
```

```
# Description:
#   This control checks whether Amazon EC2 launch templates that specify an Amazon
 EC2 instance type or use attribute based instance selection, specify only AWS Nitro
 instance types.
#
# Reports on:
#   AWS::EC2::LaunchTemplate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' or 'InstanceRequirements' in 'LaunchTemplateData' has not
 been provided
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' in 'LaunchTemplateData' has been provided
#       And: 'InstanceType' has been set to a non-Nitro instance type
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceRequirements' in 'LaunchTemplateData' has been provided as a
 struct
#       And: In 'InstanceRequirements', 'AllowedInstanceTypes' has not been provided or
 provided as
#            an empty list
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
```

```
#       And: 'InstanceRequirements' in 'LaunchTemplateData' has been provided as a
 struct
#       And: In 'InstanceRequirements', 'AllowedInstanceTypes' has been provided as a
 non-empty list
#            that contains one or more non-Nitro instance types
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' in 'LaunchTemplateData' has been provided
#       And: 'InstanceType' has been set to a Nitro instance type
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceRequirements' in 'LaunchTemplateData' has been provided as a
 struct
#       And: In 'InstanceRequirements', 'AllowedInstanceTypes' has been provided as a
 non-empty list
#            that contains only Nitro instance types
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let EC2_LAUNCH_TEMPLATE_TYPE = "AWS::EC2::LaunchTemplate"
let NITRO_INSTANCE_TYPES = [
    /^a1\..+?$/,
    /^c5\..+?$/, /^c5a\..+?$/, /^c5ad\..+?$/, /^c5d\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/,
    /^c6g\..+?$/, /^c6gd\..+?$/, /^c6gn\..+?$/, /^c6i\..+?$/, /^c6id\..+?$/, /^c6in\..
+?$/,
    /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..+?$/, /^c7i-flex\..+?$/,
    /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g5g\..+?$/, /^g6\..+?$/, /^g6e\..+?$/,
    /^g6f\..+?$/, /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3\.metal$/, /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?
$/,
    /^i8g\..+?$/, /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5\..+?$/, /^m5a\..+?$/, /^m5ad\..+?$/, /^m5d\..+?$/, /^m5dn\..+?$/, /^m5n\..+?
$/,
```

```
    /^m5zn\..+?$/, /^m6a\..+?$/, /^m6g\..+?$/, /^m6gd\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/, /^mac1\.metal$/,
    /^mac2-m1ultra\.metal$/, /^mac2-m2\.metal$/, /^mac2-m2pro\.metal$/, /^mac2\.metal
$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5\..+?$/, /^r5a\..+?$/, /^r5ad\..+?$/, /^r5b\..+?$/, /^r5d\..+?$/, /^r5dn\..+?
$/,
    /^r5n\..+?$/, /^r6a\..+?$/, /^r6g\..+?$/, /^r6gd\..+?$/, /^r6i\..+?$/, /^r6id\..+?
$/,
    /^r6idn\..+?$/, /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..
+?$/,
    /^r7iz\..+?$/, /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^t3\..+?$/, /^t3a\..+?$/, /^t4g\..+?$/, /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?
$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2gd\..+?$/, /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/,
    /^z1d\..+?$/
]

#
# Assignments
#
let ec2_launch_templates = Resources.*[ Type == %EC2_LAUNCH_TEMPLATE_TYPE ]

#
# Primary Rules
#
rule ec2_launch_template_nitro_instance_type_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                    %ec2_launch_templates not empty
 {
    check(%ec2_launch_templates.Properties)
        <<
        [CT.EC2.PR.15]: Require an Amazon EC2 instance to use an AWS Nitro instance
 type when creating from the 'AWS::EC2::LaunchTemplate' resource type
```

```
        [FIX]: When InstanceType in LaunchTemplateData has been provided, set
 InstanceType to an Amazon EC2 instance type that is based on the AWS Nitro system.
        When InstanceRequirements in LaunchTemplateData has been provided, set
 AllowedInstanceTypes to a list of Amazon EC2 instance types based on the AWS Nitro
 system.
        >>
}

rule ec2_launch_template_nitro_instance_type_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_LAUNCH_TEMPLATE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_LAUNCH_TEMPLATE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.15]: Require an Amazon EC2 instance to use an AWS Nitro instance
 type when creating from the 'AWS::EC2::LaunchTemplate' resource type
        [FIX]: When InstanceType in LaunchTemplateData has been provided, set
 InstanceType to an Amazon EC2 instance type that is based on the AWS Nitro system.
 When InstanceRequirements in LaunchTemplateData has been provided,
        set AllowedInstanceTypes to a list of Amazon EC2 instance types based on the
 AWS Nitro system.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_launch_template) {
    %ec2_launch_template[
        # Scenario 2
        filter_instance_type_provided(this)
    ] {
        LaunchTemplateData {
            # Scenarios 3 and 6
            InstanceType in %NITRO_INSTANCE_TYPES
        }
    }

    %ec2_launch_template[
        # Scenario 2
        filter_instance_requirements_provided(this)
    ] {
        LaunchTemplateData {
            InstanceRequirements is_struct
            InstanceRequirements {
                # Scenarios 4, 5 and 7
```

```
                AllowedInstanceTypes exists
                AllowedInstanceTypes is_list
                AllowedInstanceTypes not empty
                AllowedInstanceTypes[*] in %NITRO_INSTANCE_TYPES
            }
        }
    }
}

rule filter_instance_type_provided(ec2_launch_template) {
    %ec2_launch_template {
        LaunchTemplateData exists
        LaunchTemplateData is_struct

        LaunchTemplateData {
            InstanceType exists
        }
    }
}

rule filter_instance_requirements_provided(ec2_launch_template) {
    %ec2_launch_template {
        LaunchTemplateData exists
        LaunchTemplateData is_struct

        LaunchTemplateData {
            InstanceRequirements exists
        }
    }
}



#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.EC2.PR.15 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceType: t3.micro
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceRequirements:
          AllowedInstanceTypes:
          - m5.*
          - c5.*
          VCpuCount:
            Max: 16
            Min: 1
          MemoryMiB:
            Min: 1024
            Max: 17000
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceType: t2.micro
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        InstanceRequirements:
          AllowedInstanceTypes:
          - t2.micro
          VCpuCount:
            Max: 16
            Min: 1
          MemoryMiB:
            Min: 1024
            Max: 17000
```

[CT.EC2.PR.16] Require an Amazon EC2 instance to use an AWS Nitro instance type when created using the 'AWS::EC2::Instance' resource type

This control checks whether an Amazon EC2 instance is configured to run using an AWS Nitro instance type.

- **Control objective:** Protect data integrity, Enforce least privilege

- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`
- **CloudFormation guard rule:** CT.EC2.PR.16 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.16 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.16 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, reducing the possibility of human error and tampering.

Usage considerations

- This control requires that the InstanceType property is provided and set to a Nitro instance type. This setting prevents you from inheriting an instance type, by way of an Amazon EC2 launch template.

Remediation for rule failure

Set the value of the InstanceType property to an Amazon EC2 instance type based on the AWS Nitro system.

The examples that follow show how to implement this remediation.

Amazon EC2 Instance - Example

An Amazon EC2 instance configured with an instance type based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Instance": {
        "Type": "AWS::EC2::Instance",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "t3.micro"
        }
    }
}
```

YAML example

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: t3.micro
```

CT.EC2.PR.16 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_instance_nitro_instance_type_check
#
# Description:
```

```
#   This control checks whether an Amazon EC2 instance is configured to run using an
 AWS Nitro instance type.
#
# Reports on:
#   AWS::EC2::Instance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' been provided and set to a non-Nitro instance type
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' been provided and set to a Nitro instance type
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let NITRO_INSTANCE_TYPES = [
    /^a1\..+?$/,
    /^c5\..+?$/, /^c5a\..+?$/, /^c5ad\..+?$/, /^c5d\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/,
    /^c6g\..+?$/, /^c6gd\..+?$/, /^c6gn\..+?$/, /^c6i\..+?$/, /^c6id\..+?$/, /^c6in\..
+?$/,
    /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..+?$/, /^c7i-flex\..+?$/,
```

```
    /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g5g\..+?$/, /^g6\..+?$/, /^g6e\..+?$/,
    /^g6f\..+?$/, /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3\.metal$/, /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?
$/,
    /^i8g\..+?$/, /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5\..+?$/, /^m5a\..+?$/, /^m5ad\..+?$/, /^m5d\..+?$/, /^m5dn\..+?$/, /^m5n\..+?
$/,
    /^m5zn\..+?$/, /^m6a\..+?$/, /^m6g\..+?$/, /^m6gd\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/, /^mac1\.metal$/,
    /^mac2-m1ultra\.metal$/, /^mac2-m2\.metal$/, /^mac2-m2pro\.metal$/, /^mac2\.metal
$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5\..+?$/, /^r5a\..+?$/, /^r5ad\..+?$/, /^r5b\..+?$/, /^r5d\..+?$/, /^r5dn\..+?
$/,
    /^r5n\..+?$/, /^r6a\..+?$/, /^r6g\..+?$/, /^r6gd\..+?$/, /^r6i\..+?$/, /^r6id\..+?
$/,
    /^r6idn\..+?$/, /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..
+?$/,
    /^r7iz\..+?$/, /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^t3\..+?$/, /^t3a\..+?$/, /^t4g\..+?$/, /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?
$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2gd\..+?$/, /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/,
    /^z1d\..+?$/
]

#
# Assignments
#
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
```

```
# Primary Rules
#
rule ec2_instance_nitro_instance_type_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %ec2_instances not empty {
    check(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.16]: Require an Amazon EC2 instance to use an AWS Nitro instance
 type when created using the 'AWS::EC2::Instance' resource type
        [FIX]: Set the value of the InstanceType property to an Amazon EC2 instance
 type based on the AWS Nitro system.
        >>
}

rule ec2_instance_nitro_instance_type_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.16]: Require an Amazon EC2 instance to use an AWS Nitro instance
 type when created using the 'AWS::EC2::Instance' resource type
        [FIX]: Set the value of the InstanceType property to an Amazon EC2 instance
 type based on the AWS Nitro system.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_instance) {
    %ec2_instance {
        # Scenario 2
        InstanceType exists
        # Scenarios 3 and 4
        InstanceType in %NITRO_INSTANCE_TYPES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
```

```
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.16 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2Instance:
```

```
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t2.micro
```

[CT.EC2.PR.17] Require an Amazon EC2 dedicated host to use an AWS Nitro instance type

This control checks whether an Amazon EC2 dedicated host is configured to run using an AWS Nitro instance type or family.

- **Control objective:** Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Host`
- **CloudFormation guard rule:** CT.EC2.PR.17 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.17 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.17 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, reducing the possibility of human error and tampering.

Usage considerations

- When you allocate a dedicated host in your account, you can choose a configuration that supports either a single instance type, or multiple instance types within the same instance family. The number of instances that you can run on a host depends on the configuration you choose. See Instance capacity configurations in the *Amazon EC2 User Guide for Linux Instances* for information about support for single instance types and multiple instance types.

Remediation for rule failure

Set the value of the InstanceType property to an Amazon EC2 instance type that is based on the AWS Nitro system, and that supports dedicated hosts, or set the value of the InstanceFamily property to an Amazon EC2 instance family that is based on the AWS Nitro system, and that supports dedicated hosts and multiple instance types.

The examples that follow show how to implement this remediation.

Amazon EC2 Host - Example One

An Amazon EC2 dedicated host configured with an instance family that is based on the AWS Nitro system, and that supports dedicated hosts and multiple instance types. The example is shown in JSON and in YAML.

JSON example

```
{
    "DedicatedHost": {
        "Type": "AWS::EC2::Host",
        "Properties": {
            "AutoPlacement": "on",
            "AvailabilityZone": {
                "Fn::Select": [
                    0,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
```

```
            "InstanceFamily": "m5"
        }
    }
}
```

YAML example

```
DedicatedHost:
  Type: AWS::EC2::Host
  Properties:
    AutoPlacement: 'on'
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    InstanceFamily: m5
```

The examples that follow show how to implement this remediation.

Amazon EC2 Host - Example Two

An Amazon EC2 dedicated host configured with an instance type that is based on the AWS Nitro system, and that supports dedicated hosts. The example is shown in JSON and in YAML.

JSON example

```
{
    "DedicatedHost": {
        "Type": "AWS::EC2::Host",
        "Properties": {
            "AutoPlacement": "on",
            "AvailabilityZone": {
                "Fn::Select": [
                    0,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
```

```
            "InstanceType": "m6a.large"
        }
    }
}
```

YAML example

```
DedicatedHost:
  Type: AWS::EC2::Host
  Properties:
    AutoPlacement: 'on'
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    InstanceType: m6a.large
```

CT.EC2.PR.17 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_host_nitro_check
#
# Description:
#   This control checks whether an Amazon EC2 dedicated host is configured to run using
 an AWS Nitro instance type or family.
#
# Reports on:
#   AWS::EC2::Host
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 dedicated host
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon EC2 dedicated host resource
#       And: 'InstanceFamily' or 'InstanceType' have not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon EC2 dedicated host resource
#       And: 'InstanceType' has not been provided
#       And: 'InstanceFamily' has been provided and set to an instance family other
 than
#            a Nitro instance family with support for both dedicated hosts and multiple
#            instance types
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon EC2 dedicated host resource
#       And: 'InstanceFamily' has not been provided
#       And: 'InstanceType' has been provided and set to an instance type other than
#            a Nitro instance type with dedicated host support
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon EC2 dedicated host resource
#       And: 'InstanceType' has not been provided
#       And: 'InstanceFamily' has been provided and set to a Nitro instance family with
#            support for both dedicated hosts and multiple instance types
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon EC2 dedicated host resource
#       And: 'InstanceFamily' has not been provided
#       And: 'InstanceType' has been provided and set to a Nitro instance type with
#            dedicated host support
#      Then: PASS

#
# Constants
```

```
#
let INPUT_DOCUMENT = this
let EC2_DEDICATED_HOST_TYPE = "AWS::EC2::Host"
let NITRO_INSTANCE_FAMILIES_WITH_DEDICATED_HOST_SUPPORT = [
    /^a1$/,
    /^c5$/, /^c5d$/, /^c5n$/, /^c6a$/, /^c6g$/, /^c6gd$/, /^c6gn$/, /^c6i$/, /^c6id$/,
    /^c6in$/, /^c7a$/, /^c7g$/, /^c7gd$/, /^c7gn$/, /^c7i$/, /^c8g$/, /^c8gd$/, /^c8gn
$/,
    /^dl1$/, /^dl2q$/,
    /^f2$/,
    /^g4ad$/, /^g4dn$/, /^g5$/, /^g5g$/, /^g6$/, /^g6e$/,
    /^i3en$/, /^i4g$/, /^i4i$/, /^i7i$/, /^i7ie$/, /^i8g$/, /^im4gn$/, /^inf1$/, /
^inf2$/,
    /^m5$/, /^m5a$/, /^m5ad$/, /^m5d$/, /^m5dn$/, /^m5n$/, /^m5zn$/, /^m6a$/, /^m6g$/,
    /^m6gd$/, /^m6i$/, /^m6id$/, /^m6idn$/, /^m6in$/, /^m7a$/, /^m7g$/, /^m7gd$/, /^m7i
$/,
    /^m8g$/, /^m8gd$/,
    /^p3dn$/, /^p4d$/,
    /^r5$/, /^r5a$/, /^r5ad$/, /^r5b$/, /^r5d$/, /^r5dn$/, /^r5n$/, /^r6a$/, /^r6g$/,
    /^r6gd$/, /^r6i$/, /^r6id$/, /^r6idn$/, /^r6in$/, /^r7a$/, /^r7g$/, /^r7gd$/, /^r7i
$/,
    /^r7iz$/, /^r8g$/, /^r8gd$/, /^r8i$/,
    /^t3$/, /^trn1$/,
    /^u-12tb1$/, /^u-18tb1$/, /^u-24tb1$/, /^u-6tb1$/, /^u-9tb1$/, /^u7i-12tb$/,
    /^u7i-6tb$/, /^u7i-8tb$/, /^u7in-16tb$/, /^u7in-24tb$/, /^u7in-32tb$/,
    /^vt1$/,
    /^x2gd$/, /^x2idn$/, /^x2iedn$/, /^x2iezn$/, /^x8g$/,
    /^z1d$/
]
let NITRO_INSTANCE_TYPES_WITH_DEDICATED_HOST_SUPPORT = [
    /^a1\..+?$/,
    /^c5\..+?$/, /^c5d\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/, /^c6g\..+?$/, /^c6gd\..+?$/,
    /^c6gn\..+?$/, /^c6i\..+?$/, /^c6id\..+?$/, /^c6in\..+?$/, /^c7a\..+?$/, /^c7g\..+?
$/,
    /^c7gd\..+?$/, /^c7gn\..+?$/, /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..
+?$/,
    /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g5g\..+?$/, /^g6\..+?$/, /^g6e\..+?$/,
    /^i3\.metal$/, /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?
$/,
    /^i8g\..+?$/, /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/,
    /^m5\..+?$/, /^m5a\..+?$/, /^m5ad\..+?$/, /^m5d\..+?$/, /^m5dn\..+?$/, /^m5n\..+?
$/,
```

```
    /^m5zn\..+?$/, /^m6a\..+?$/, /^m6g\..+?$/, /^m6gd\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/, /^m7i\..
+?$/,
    /^m8g\..+?$/, /^m8gd\..+?$/, /^mac1\.metal$/, /^mac2-m1ultra\.metal$/,
    /^mac2-m2\.metal$/, /^mac2-m2pro\.metal$/, /^mac2\.metal$/,
    /^p3dn\..+?$/, /^p4d\..+?$/,
    /^r5\..+?$/, /^r5a\..+?$/, /^r5ad\..+?$/, /^r5b\..+?$/, /^r5d\..+?$/, /^r5dn\..+?
$/,
    /^r5n\..+?$/, /^r6a\..+?$/, /^r6g\..+?$/, /^r6gd\..+?$/, /^r6i\..+?$/, /^r6id\..+?
$/,
    /^r6idn\..+?$/, /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..
+?$/,
    /^r7iz\..+?$/, /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i\..+?$/,
    /^t3\..+?$/, /^trn1\..+?$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-6tb1\..+?$/, /^u-9tb1\..
+?$/,
    /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/, /^u7in-16tb\..+?$/,
    /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2gd\..+?$/, /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/,
    /^z1d\..+?$/
]

#
# Assignments
#
let ec2_dedicated_hosts = Resources.*[ Type == %EC2_DEDICATED_HOST_TYPE ]

#
# Primary Rules
#
rule ec2_host_nitro_check when is_cfn_template(%INPUT_DOCUMENT)
                              %ec2_dedicated_hosts not empty {
    check(%ec2_dedicated_hosts.Properties)
        <<
        [CT.EC2.PR.17]: Require an Amazon EC2 dedicated host to use an AWS Nitro
 instance type
        [FIX]: Set the value of the InstanceType property to an Amazon EC2 instance
 type that is based on the AWS Nitro system, and
        that supports dedicated hosts, or set the value of the InstanceFamily property
 to an Amazon EC2 instance family that is
        based on the AWS Nitro system, and that supports dedicated hosts and multiple
 instance types.
```

```
        >>
}

rule ec2_host_nitro_check when is_cfn_hook(%INPUT_DOCUMENT, %EC2_DEDICATED_HOST_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_DEDICATED_HOST_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.17]: Require an Amazon EC2 dedicated host to use an AWS Nitro
 instance type
        [FIX]: Set the value of the InstanceType property to an Amazon EC2 instance
 type that is based on the AWS Nitro system, and
        that supports dedicated hosts, or set the value of the InstanceFamily property
 to an Amazon EC2 instance family that is based
        on the AWS Nitro system, and that supports dedicated hosts and multiple
 instance types.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_dedicated_host) {
    %ec2_dedicated_host {
        # Scenario 2
        InstanceFamily exists or
        InstanceType exists
    }

    %ec2_dedicated_host[
        InstanceFamily exists
    ] {
        # Scenario 3 and 5
        InstanceFamily in %NITRO_INSTANCE_FAMILIES_WITH_DEDICATED_HOST_SUPPORT
    }
    %ec2_dedicated_host[
        InstanceType exists
    ] {
        # Scenario 4 and 6
        InstanceType in %NITRO_INSTANCE_TYPES_WITH_DEDICATED_HOST_SUPPORT
    }
}

#
# Utility Rules
#
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.17 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DedicatedHost:
    Type: AWS::EC2::Host
    Properties:
      AutoPlacement: 'on'
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
      InstanceFamily: m5
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DedicatedHost:
    Type: AWS::EC2::Host
    Properties:
      AutoPlacement: 'on'
```

```
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
      InstanceType: m6a.large
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DedicatedHost:
    Type: AWS::EC2::Host
    Properties:
      AutoPlacement: 'on'
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
      InstanceType: c4.large
```

[CT.EC2.PR.18] Require an Amazon EC2 fleet to override only those launch templates with AWS Nitro instance types

This control checks that Amazon EC2 fleets only override launch templates with AWS Nitro instance types.

- **Control objective:** Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::EC2Fleet`
- **CloudFormation guard rule:** CT.EC2.PR.18 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.18 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.18 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, reducing the possibility of human error and tampering.

Usage considerations

- This control applies only when launch template overrides have been provided, specifically, entries in `LaunchTemplateConfigs` specifying one or more `Overrides` that also include values for `InstanceType` or `InstanceRequirements` properties.
- This control does not check the instance type configured on a launch template. To ensure that launch templates use Nitro instances types, use this control in conjunction with related controls that check launch templates for Nitro instance types.

Remediation for rule failure

For any entry in the LaunchTemplateConfigs parameter, if it has one or more Overrides properties that also include `InstanceType` or `InstanceRequirements` fields, set the value of the `InstanceType` field to an EC2 instance type based on the AWS Nitro system, or set the value of the `AllowedInstanceTypes` field in the InstanceRequirements property to one or more EC2 instance types that are based on the AWS Nitro system.

The examples that follow show how to implement this remediation.

Amazon EC2 Fleet - Example One

An Amazon EC2 fleet configured with a launch template override and instance type based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Fleet": {
        "Type": "AWS::EC2::EC2Fleet",
        "Properties": {
            "TargetCapacitySpecification": {
                "TotalTargetCapacity": 1,
                "DefaultTargetCapacityType": "on-demand"
            },
            "LaunchTemplateConfigs": [
                {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceType": "t3.micro"
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
EC2Fleet:
  Type: AWS::EC2::EC2Fleet
  Properties:
    TargetCapacitySpecification:
```

```
      TotalTargetCapacity: 1
      DefaultTargetCapacityType: on-demand
    LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceType: t3.micro
```

The examples that follow show how to implement this remediation.

Amazon EC2 Fleet - Example Two

An Amazon EC2 fleet configured with a launch template override and instance requirements that specify a list of allowed instances based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Fleet": {
        "Type": "AWS::EC2::EC2Fleet",
        "Properties": {
            "TargetCapacitySpecification": {
                "TotalTargetCapacity": 1,
                "DefaultTargetCapacityType": "on-demand"
            },
            "LaunchTemplateConfigs": [
                {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    },
```

```
                    "Overrides": [
                        {
                            "InstanceRequirements": {
                                "VCpuCount": {
                                    "Min": 2,
                                    "Max": 4
                                },
                                "MemoryMiB": {
                                    "Min": 4000,
                                    "Max": 8000
                                },
                                "AllowedInstanceTypes": [
                                    "m5.*",
                                    "c5.*"
                                ]
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
EC2Fleet:
  Type: AWS::EC2::EC2Fleet
  Properties:
    TargetCapacitySpecification:
      TotalTargetCapacity: 1
      DefaultTargetCapacityType: on-demand
    LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceRequirements:
              VCpuCount:
                Min: 2
                Max: 4
```

```
              MemoryMiB:
                Min: 4000
                Max: 8000
              AllowedInstanceTypes:
                - m5.*
                - c5.*
```

CT.EC2.PR.18 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_fleet_nitro_instance_override_check
#
# Description:
#   This control checks that Amazon EC2 fleets only override launch templates with AWS
 Nitro instance types.
#
# Reports on:
#   AWS::EC2::EC2Fleet
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 fleet resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: For every entry in 'LaunchTemplateConfigs', 'Overrides' has not been
 provided
#            or has been provided as an empty list
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: For an entry in 'LaunchTemplateConfigs', 'Overrides' has been provided as
 a non-empty list
#       And: For the same entry in 'LaunchTemplateConfigs', no entries in 'Overrides'
 include
#            'InstanceType' or 'InstanceRequirements'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to
 an instance type
#            other than a Nitro instance type
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided or has been
#            provided as an empty list
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided as a non-empty list
#       And: An entry in 'AllowedInstanceTypes' is set to an instance type other than a
 Nitro instance type
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
```

```
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to a
 Nitro instance type
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided as a non-empty list
#       And: Every entry in 'AllowedInstanceTypes' is set to a Nitro instance type
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let EC2_FLEET_TYPE = "AWS::EC2::EC2Fleet"
let NITRO_INSTANCE_TYPES = [
    /^a1\..+?$/,
    /^c5\..+?$/, /^c5a\..+?$/, /^c5ad\..+?$/, /^c5d\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/,
    /^c6g\..+?$/, /^c6gd\..+?$/, /^c6gn\..+?$/, /^c6i\..+?$/, /^c6id\..+?$/, /^c6in\..
+?$/,
    /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..+?$/, /^c7i-flex\..+?$/,
    /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g5g\..+?$/, /^g6\..+?$/, /^g6e\..+?$/,
    /^g6f\..+?$/, /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3\.metal$/, /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?
$/,
    /^i8g\..+?$/, /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5\..+?$/, /^m5a\..+?$/, /^m5ad\..+?$/, /^m5d\..+?$/, /^m5dn\..+?$/, /^m5n\..+?
$/,
    /^m5zn\..+?$/, /^m6a\..+?$/, /^m6g\..+?$/, /^m6gd\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/, /^mac1\.metal$/,
    /^mac2-m1ultra\.metal$/, /^mac2-m2\.metal$/, /^mac2-m2pro\.metal$/, /^mac2\.metal
$/,
```

```
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5\..+?$/, /^r5a\..+?$/, /^r5ad\..+?$/, /^r5b\..+?$/, /^r5d\..+?$/, /^r5dn\..+?
$/,
    /^r5n\..+?$/, /^r6a\..+?$/, /^r6g\..+?$/, /^r6gd\..+?$/, /^r6i\..+?$/, /^r6id\..+?
$/,
    /^r6idn\..+?$/, /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..
+?$/,
    /^r7iz\..+?$/, /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^t3\..+?$/, /^t3a\..+?$/, /^t4g\..+?$/, /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?
$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2gd\..+?$/, /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/,
    /^z1d\..+?$/
]

#
# Assignments
#
let ec2_fleets = Resources.*[ Type == %EC2_FLEET_TYPE ]

#
# Primary Rules
#
rule ec2_fleet_nitro_instance_override_check when is_cfn_template(%INPUT_DOCUMENT)
                                                 %ec2_fleets not empty {
    check(%ec2_fleets.Properties)
        <<
        [CT.EC2.PR.18]: Require an Amazon EC2 fleet to override only those launch
 templates with AWS Nitro instance types
        [FIX]: For any entry in the LaunchTemplateConfigs parameter, if it has one or
 more Overrides properties that also include 'InstanceType' or 'InstanceRequirements'
 fields, set the value of the 'InstanceType' field to an Amazon EC2 instance type based
 on the AWS Nitro system, or set the value of the 'AllowedInstanceTypes' field in the
 InstanceRequirements property to one or more Amazon EC2 instance types that are based
 on the AWS Nitro system.
        >>
}
```

```
rule ec2_fleet_nitro_instance_override_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_FLEET_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_FLEET_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.18]: Require an Amazon EC2 fleet to override only those launch
 templates with AWS Nitro instance types
        [FIX]: For any entry in the LaunchTemplateConfigs parameter, if it has one or
 more Overrides properties that also include 'InstanceType' or 'InstanceRequirements'
 fields, set the value of the 'InstanceType' field to an Amazon EC2 instance type based
 on the AWS Nitro system, or set the value of the 'AllowedInstanceTypes' field in the
 InstanceRequirements property to one or more Amazon EC2 instance types that are based
 on the AWS Nitro system.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_fleet) {
    %ec2_fleet [
        # Scenarios 2 and 3
        filter_launch_template_overrides(this)
    ] {
        LaunchTemplateConfigs[*] {
            Overrides[ InstanceType exists ] {
                # Scenarios 4 and 7
                InstanceType in %NITRO_INSTANCE_TYPES
            }
            Overrides[ InstanceRequirements exists ] {
                InstanceRequirements {
                    # Scenarios 5, 6 and 8
                    AllowedInstanceTypes exists
                    AllowedInstanceTypes is_list
                    AllowedInstanceTypes not empty
                    AllowedInstanceTypes[*] in %NITRO_INSTANCE_TYPES
                }
            }
        }
    }
}

rule filter_launch_template_overrides(ec2_fleet) {
    %ec2_fleet {
        LaunchTemplateConfigs exists
```

```
        LaunchTemplateConfigs is_list
        LaunchTemplateConfigs not empty

        some LaunchTemplateConfigs[*] {
            Overrides exists
            Overrides is_list
            Overrides not empty

            some Overrides[*] {
                InstanceType exists or
                InstanceRequirements exists
            }
        }
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.18 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
```

```
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceType: t3.micro
```

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
```

```
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceRequirements:
            VCpuCount:
              Min: 2
              Max: 4
            MemoryMiB:
              Min: 4000
              Max: 8000
            AllowedInstanceTypes:
            - m5.*
            - c5.*
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
```

```
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceType: t2.micro
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
```

```
        Overrides:
        - InstanceRequirements:
            VCpuCount:
              Min: 0
              Max: 4
            MemoryMiB:
              Min: 0
              Max: 4000
            AllowedInstanceTypes:
            - c4.large
```

[CT.EC2.PR.19] Require an Amazon EC2 instance to use an AWS Nitro instance type that supports encryption in-transit between instances when created using the AWS::EC2::Instance resource type

This control checks whether an Amazon EC2 instance has been configured to run using a Nitro instance type that supports encryption in-transit between instances.

- **Control objective:** Encrypt data in transit, Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::Instance`
- **CloudFormation guard rule:** CT.EC2.PR.19 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.19 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.19 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware.

Virtualization resources are offloaded to dedicated hardware and software, minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, reducing the possibility of human error and tampering.

AWS provides secure and private connectivity between Amazon EC2 instances of all types. In addition, some instance types utilize the offload capabilities of the underlying Nitro System hardware to encrypt in-transit traffic between instances, automatically. This encryption process uses Authenticated Encryption with Associated Data (AEAD) algorithms, with 256-bit encryption. It has no impact on network performance.

Usage considerations

- This control requires that the InstanceType property is provided and set to a Nitro instance type that supports encryption in transit between instances. This setting prevents you from inheriting an instance type by way of an Amazon EC2 launch template.
- To support in-transit traffic encryption between instances, in addition to using one of the Amazon EC2 instance types required by this control, the instances must be in the same Region, and they must be in the same VPC or group of peered VPCs, in which traffic does not pass through a virtual network device or service, such as a load balancer or a transit gateway.

Remediation for rule failure

Set `InstanceType` to an Amazon EC2 instance type based on the AWS Nitro system that supports encryption in-transit between instances.

The examples that follow show how to implement this remediation.

Amazon EC2 Instance - Example

An Amazon EC2 instance configured with an instance type based on the AWS Nitro system, and that supports encryption in transit between instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Instance": {
        "Type": "AWS::EC2::Instance",
```

```
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "c5a.large"
        }
    }
}
```

YAML example

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: c5a.large
```

CT.EC2.PR.19 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_instance_nitro_encryption_in_transit_check
#
# Description:
#   This control checks whether an Amazon EC2 instance has been configured to run using
 a Nitro instance type that supports encryption in-transit between instances.
#
# Reports on:
#   AWS::EC2::Instance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' been provided and set to an instance type other than a
 Nitro
#            instance type that supports encryption in-transit between instances
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 instance resource
#       And: 'InstanceType' been provided and set to a Nitro instance type that
 supports
#            encryption in-transit between instances
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let EC2_INSTANCE_TYPE = "AWS::EC2::Instance"
let NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES = [
    /^c5a\..+?$/, /^c5ad\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/, /^c6gn\..+?$/, /^c6i\..+?
$/,
    /^c6id\..+?$/, /^c6in\..+?$/, /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..
+?$/,
    /^c7i-flex\..+?$/, /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g6\..+?$/, /^g6e\..+?$/, /^g6f\..+?$/,
    /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?$/, /^i8g\..+?
$/,
```

```
    /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5dn\..+?$/, /^m5n\..+?$/, /^m5zn\..+?$/, /^m6a\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5dn\..+?$/, /^r5n\..+?$/, /^r6a\..+?$/, /^r6i\..+?$/, /^r6id\..+?$/, /^r6idn\..
+?$/,
    /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..+?$/, /^r7iz\..+?
$/,
    /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/
]

#
# Assignments
#
let ec2_instances = Resources.*[ Type == %EC2_INSTANCE_TYPE ]

#
# Primary Rules
#
rule ec2_instance_nitro_encryption_in_transit_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %ec2_instances not empty {
    check(%ec2_instances.Properties)
        <<
        [CT.EC2.PR.19]: Require an Amazon EC2 instance to use a nitro instance
 type that supports encryption in-transit between instances when created using the
 AWS::EC2::Instance resource type
        [FIX]: Set 'InstanceType' to an Amazon EC2 instance type based on the AWS Nitro
 system that supports encryption in-transit between instances.
        >>
}
```

```
rule ec2_instance_nitro_encryption_in_transit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EC2_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.19]: Require an Amazon EC2 instance to use a nitro instance
 type that supports encryption in-transit between instances when created using the
 AWS::EC2::Instance resource type
        [FIX]: Set 'InstanceType' to an Amazon EC2 instance type based on the AWS Nitro
 system that supports encryption in-transit between instances.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_instance) {
    %ec2_instance {
        # Scenario 2
        InstanceType exists
        # Scenarios 3 and 4
        InstanceType in %NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.19 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: c5a.large
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t2.micro
```

[CT.EC2.PR.20] Require an Amazon EC2 fleet to override only those launch templates with AWS Nitro instance types that support encryption in transit between instances

This control checks whether an Amazon EC2 fleet overrides only the launch templates based upon AWS Nitro instance types that support encryption in transit between instances.

- **Control objective:** Encrypt data in transit, Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EC2::EC2Fleet`
- **CloudFormation guard rule:** CT.EC2.PR.20 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EC2.PR.20 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EC2.PR.20 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, reducing the possibility of human error and tampering.

AWS provides secure and private connectivity between Amazon EC2 instances of all types. In addition, some instance types utilize the offload capabilities of the underlying Nitro System hardware to encrypt in-transit traffic between instances, automatically. This encryption process uses Authenticated Encryption with Associated Data (AEAD) algorithms, with 256-bit encryption. It has no impact on network performance.

Usage considerations

- This control applies only when launch template overrides have been provided, specifically, entries in `LaunchTemplateConfigs` specifying one or more `Overrides` that also include values for `InstanceType` or `InstanceRequirements` properties.
- This control does not check the instance type configured on a launch template. To ensure that launch templates use Nitro instances types that support encryption in-transit between instances, use this control in conjunction with related controls that check launch templates for Nitro instance types, and that the Nitro instance types support encryption in transit between instances.
- To support in-transit traffic encryption between instances, in addition to using one of the EC2 instance types required by this control, the instances must be in the same AWS Region, and they must be in the same VPC or group of peered VPCs, in which traffic does not pass through a virtual network device or service, such as a load balancer or a transit gateway.

Remediation for rule failure

For any entry in the LaunchTemplateConfigs parameter, if it has one or more Overrides properties that also include `InstanceType` or `InstanceRequirements` fields, set the value of the InstanceType field to an Amazon EC2 instance type that's based on the AWS Nitro system, and that supports encryption in transit between instances, or set the value of the AllowedInstanceTypes field in the InstanceRequirements property to one or more Amazon EC2 instance types that are based on the AWS Nitro system, and that support encryption in transit between instances.

The examples that follow show how to implement this remediation.

Amazon EC2 Fleet - Example One

An Amazon EC2 fleet configured with a launch template override and instance type that is based on the AWS Nitro system, and that supports encryption in transit between instances. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "EC2Fleet": {
        "Type": "AWS::EC2::EC2Fleet",
        "Properties": {
            "TargetCapacitySpecification": {
                "TotalTargetCapacity": 1,
                "DefaultTargetCapacityType": "on-demand"
            },
            "LaunchTemplateConfigs": [
                {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceType": "c5a.large"
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
EC2Fleet:
  Type: AWS::EC2::EC2Fleet
  Properties:
    TargetCapacitySpecification:
      TotalTargetCapacity: 1
      DefaultTargetCapacityType: on-demand
    LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
```

```
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceType: c5a.large
```

The examples that follow show how to implement this remediation.

Amazon EC2 Fleet - Example Two

An Amazon EC2 fleet configured with a launch template override and instance requirements that specify a list of allowed instances, that are based on the AWS Nitro system, and that support encryption in transit between instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "EC2Fleet": {
        "Type": "AWS::EC2::EC2Fleet",
        "Properties": {
            "TargetCapacitySpecification": {
                "TotalTargetCapacity": 1,
                "DefaultTargetCapacityType": "on-demand"
            },
            "LaunchTemplateConfigs": [
                {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceRequirements": {
                                "VCpuCount": {
```

```
                                    "Min": 2,
                                    "Max": 4
                                },
                                "MemoryMiB": {
                                    "Min": 4000,
                                    "Max": 8000
                                },
                                "AllowedInstanceTypes": [
                                    "m6a.*",
                                    "c5a.*"
                                ]
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
EC2Fleet:
  Type: AWS::EC2::EC2Fleet
  Properties:
    TargetCapacitySpecification:
      TotalTargetCapacity: 1
      DefaultTargetCapacityType: on-demand
    LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceRequirements:
              VCpuCount:
                Min: 2
                Max: 4
              MemoryMiB:
                Min: 4000
                Max: 8000
              AllowedInstanceTypes:
```

```
                - m6a.*
                - c5a.*
```

CT.EC2.PR.20 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ec2_fleet_nitro_encryption_in_transit_override_check
#
# Description:
#   This control checks whether an Amazon EC2 fleet overrides only the launch templates
 based upon AWS Nitro instance types that support encryption in transit between
 instances.
#
# Reports on:
#   AWS::EC2::EC2Fleet
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EC2 fleet resources
#     Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: For every entry in 'LaunchTemplateConfigs', 'Overrides' has not been
 provided
#            or has been provided as an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an Amazon EC2 fleet resource
#       And: For an entry in 'LaunchTemplateConfigs', 'Overrides' has been provided as
 a non-empty list
#       And: For the same entry in 'LaunchTemplateConfigs', no entries in 'Overrides'
 include
#            'InstanceType' or 'InstanceRequirements'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to
 an instance type
#            other than a Nitro instance type that supports encryption in-transit
 between instances
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided or has been
#            provided as an empty list
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: An entry in 'AllowedInstanceTypes' is set to an instance type other than a
 Nitro instance
#            type that supports encryption in-transit between instances
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
```

```
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to a
 Nitro instance type that
#            supports encryption in-transit between instances
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 fleet resource
#       And: 'Overrides' in 'LaunchTemplateConfigs' has been provided as a non-empty
 list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: Every entry in 'AllowedInstanceTypes' is set to a Nitro instance type that
#            supports encryption in-transit between instances
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let EC2_FLEET_TYPE = "AWS::EC2::EC2Fleet"
let NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES = [
    /^c5a\..+?$/, /^c5ad\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/, /^c6gn\..+?$/, /^c6i\..+?
$/,
    /^c6id\..+?$/, /^c6in\..+?$/, /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..
+?$/,
    /^c7i-flex\..+?$/, /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g6\..+?$/, /^g6e\..+?$/, /^g6f\..+?$/,
    /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?$/, /^i8g\..+?
$/,
    /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5dn\..+?$/, /^m5n\..+?$/, /^m5zn\..+?$/, /^m6a\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5dn\..+?$/, /^r5n\..+?$/, /^r6a\..+?$/, /^r6i\..+?$/, /^r6id\..+?$/, /^r6idn\..
+?$/,
```

```
    /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..+?$/, /^r7iz\..+?
$/,
    /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/
]
#
# Assignments
#
let ec2_fleets = Resources.*[ Type == %EC2_FLEET_TYPE ]

#
# Primary Rules
#
rule ec2_fleet_nitro_encryption_in_transit_override_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                              %ec2_fleets not empty {
    check(%ec2_fleets.Properties)
        <<
        [CT.EC2.PR.20]: Require an Amazon Amazon EC2 fleet to override only those
 launch templates with AWS Nitro instance types that support encryption in transit
 between instances
        [FIX]: For any entry in the LaunchTemplateConfigs parameter, if it has one or
 more Overrides properties that also include 'InstanceType' or 'InstanceRequirements'
 fields, set the value of the InstanceType field to an EC2 instance type that's based
 on the AWS Nitro system, and that supports encryption in transit between instances, or
 set the value of the AllowedInstanceTypes field in the InstanceRequirements property
 to one or more EC2 instance types that are based on the AWS Nitro system, and that
 support encryption in transit between instances.
        >>
}

rule ec2_fleet_nitro_encryption_in_transit_override_check when
 is_cfn_hook(%INPUT_DOCUMENT, %EC2_FLEET_TYPE) {
    check(%INPUT_DOCUMENT.%EC2_FLEET_TYPE.resourceProperties)
        <<
        [CT.EC2.PR.20]: Require an Amazon EC2 fleet to override only those launch
 templates with AWS Nitro instance types that support encryption in transit between
 instances
```

```
        [FIX]: For any entry in the LaunchTemplateConfigs parameter, if it has one or
 more Overrides properties that also include 'InstanceType' or 'InstanceRequirements'
 fields,
        set the value of the InstanceType field to an Amazon EC2 instance type that's
 based on the AWS Nitro system, and that supports encryption in transit between
 instances, or
        set the value of the AllowedInstanceTypes field in the InstanceRequirements
 property to one or more Amazon EC2 instance types that are based on the AWS Nitro
 system, and that support encryption in transit between instances.
        >>
}

#
# Parameterized Rules
#
rule check(ec2_fleet) {
    %ec2_fleet [
        # Scenarios 2 and 3
        filter_launch_template_overrides(this)
    ] {
        LaunchTemplateConfigs[*] {
            Overrides[ InstanceType exists ] {
                # Scenarios 4 and 7
                InstanceType in %NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES
            }
            Overrides[ InstanceRequirements exists ] {
                InstanceRequirements {
                    # Scenarios 5, 6 and 8
                    AllowedInstanceTypes exists
                    AllowedInstanceTypes is_list
                    AllowedInstanceTypes not empty
                    AllowedInstanceTypes[*] in
 %NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES
                }
            }
        }
    }
}

rule filter_launch_template_overrides(ec2_fleet) {
    %ec2_fleet {
        LaunchTemplateConfigs exists
        LaunchTemplateConfigs is_list
        LaunchTemplateConfigs not empty
```

```
        some LaunchTemplateConfigs[*] {
            Overrides exists
            Overrides is_list
            Overrides not empty

            some Overrides[*] {
                InstanceType exists or
                InstanceRequirements exists
            }
        }
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EC2.PR.20 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
```

```
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceType: c5a.large
```

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
```

```
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceRequirements:
            VCpuCount:
              Min: 2
              Max: 4
            MemoryMiB:
              Min: 4000
              Max: 8000
            AllowedInstanceTypes:
            - m6a.*
            - c5a.*
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
```

```
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceType: t2.micro
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  EC2Fleet:
    Type: AWS::EC2::EC2Fleet
    Properties:
      TargetCapacitySpecification:
        TotalTargetCapacity: 1
        DefaultTargetCapacityType: on-demand
      LaunchTemplateConfigs:
      - LaunchTemplateSpecification:
          LaunchTemplateId:
            Ref: LaunchTemplate
          Version:
            Fn::GetAtt: [LaunchTemplate, LatestVersionNumber]
        Overrides:
        - InstanceRequirements:
```

```
            VCpuCount:
              Min: 0
              Max: 4
            MemoryMiB:
              Min: 0
              Max: 4000
            AllowedInstanceTypes:
            - c4.large
```

Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling controls

Topics

- [CT.AUTOSCALING.PR.1] Require an Amazon EC2 Auto Scaling group to have multiple Availability Zones
- [CT.AUTOSCALING.PR.2] Require an Amazon EC2 Auto Scaling group launch configuration to configure Amazon EC2 instances for IMDSv2
- [CT.AUTOSCALING.PR.4] Require an Amazon EC2 Auto Scaling group associated with an AWS Elastic Load Balancing (ELB) to have ELB health checks activated
- [CT.AUTOSCALING.PR.5] Require that an Amazon EC2 Auto Scaling group launch configuration does not have Amazon EC2 instances with public IP addresses
- [CT.AUTOSCALING.PR.6] Require any Amazon EC2 Auto Scaling groups to use multiple instance types
- [CT.AUTOSCALING.PR.8] Require an Amazon EC2 Auto Scaling group to have EC2 launch templates configured
- [CT.AUTOSCALING.PR.9] Require an Amazon EBS volume configured through an Amazon EC2 Auto Scaling launch configuration to encrypt data at rest
- [CT.AUTOSCALING.PR.10] Require an Amazon EC2 Auto Scaling group to use only AWS Nitro instance types when overriding a launch template
- [CT.AUTOSCALING.PR.11] Require only AWS Nitro instance types that support network traffic encryption between instances to be added to an Amazon EC2 Auto Scaling group, when overriding a launch template

[CT.AUTOSCALING.PR.1] Require an Amazon EC2 Auto Scaling group to have multiple Availability Zones

This control checks whether your Amazon EC2 Auto Scaling group spans multiple Availability Zones.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.1 example templates

Explanation

Amazon EC2 Auto Scaling groups can be configured to use multiple Availability Zones. An Auto Scaling group with a single Availability Zone is preferred in some use cases, such as batch-jobs or when inter-AZ transfer costs need to be kept to a minimum. However, an Auto Scaling group that does not span multiple Availability Zones will not launch instances in another Availability Zone to compensate if the configured single Availability Zone becomes unavailable.

Remediation for rule failure

Configure Auto Scaling groups with multiple Availability Zones.

The examples that follow show how to implement this remediation.

Auto Scaling group - Example

Auto Scaling group configured with multiple Availability Zones. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "LaunchTemplate": {
                "LaunchTemplateId": {
                    "Ref": "LaunchTemplate"
                },
                "Version": {
                    "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                }
            },
            "MaxSize": "1",
            "MinSize": "0",
            "DesiredCapacity": "1",
            "AvailabilityZones": [
                {
                    "Fn::Select": [
                        0,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                {
                    "Fn::Select": [
                        1,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
```

```
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref 'LaunchTemplate'
      Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
    MaxSize: '1'
    MinSize: '0'
    DesiredCapacity: '1'
    AvailabilityZones:
      - !Select
        - 0
        - !GetAZs ''
      - !Select
        - 1
        - !GetAZs ''
```

CT.AUTOSCALING.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_multiple_az_check
#
# Description:
#   This control checks whether your Amazon EC2 Auto Scaling group spans multiple
 Availability Zones.
#
# Reports on:
#    AWS::AutoScaling::AutoScalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
# Scenario: 1
```

```
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document does not contain any Auto Scaling groups
#    Then: SKIP
# Scenario: 2
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an Auto Scaling group resource
#     And: 'AvailabilityZones' is not present on the Auto Scaling group resource
#    Then: FAIL
# Scenario: 3
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an Auto Scaling group resource
#     And: 'AvailabilityZones' is present on the Auto Scaling group resource
#     And: The number of 'AvailabilityZones' present is less than 2 (< 2) or the number
 of
#          unique 'AvailabilityZones' provided is less than 2 (< 2)
#    Then: FAIL
# Scenario: 4
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an Auto Scaling group resource
#     And: 'AvailabilityZones' is present on the Auto Scaling group resource
#     And: The number of 'AvailabilityZones' present is greater than or equal to 2 (>=
 2)
#     And: At least two unique 'AvailabilityZones' have been provided
#    Then: PASS

#
# Constants
#
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]

#
# Primary Rules
#
rule autoscaling_multiple_az_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %autoscaling_groups not empty {
    check(%autoscaling_groups.Properties)
        <<
```

```
        [CT.AUTOSCALING.PR.1]: Require an Amazon Elastic Compute Cloud Auto Scaling
 group to have multiple Availability Zones
        [FIX]: Configure Auto Scaling groups with multiple Availability Zones.
        >>
}

rule autoscaling_multiple_az_check when is_cfn_hook(%INPUT_DOCUMENT,
 %AUTOSCALING_GROUP_TYPE) {
    check(this.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.1]: Require an Amazon Elastic Compute Cloud Auto Scaling
 group to have multiple Availability Zones
        [FIX]: Configure Auto Scaling groups with multiple Availability Zones.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_group) {
    %autoscaling_group {
       # Scenario 2
       AvailabilityZones exists
       # Scenario 3 and 4
       AvailabilityZones is_list
       AvailabilityZones not empty
       AvailabilityZones[0] exists
       AvailabilityZones[1] exists
       AvailabilityZones[0] not in AvailabilityZones[1]
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.AUTOSCALING.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
        InstanceType: t3.micro
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId:
          Ref: LaunchTemplate
        Version:
          Fn::GetAtt: LaunchTemplate.LatestVersionNumber
      MaxSize: '1'
      MinSize: '0'
      DesiredCapacity: '1'
      AvailabilityZones:
      - Fn::Select:
        - 0
        - Fn::GetAZs: ""
      - Fn::Select:
        - 1
        - Fn::GetAZs: ""
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
        InstanceType: t3.micro
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId:
          Ref: LaunchTemplate
        Version:
          Fn::GetAtt: LaunchTemplate.LatestVersionNumber
      MaxSize: '1'
      MinSize: '0'
      DesiredCapacity: '1'
      AvailabilityZones:
      - Fn::Select:
        - 0
        - Fn::GetAZs: ""
```

[CT.AUTOSCALING.PR.2] Require an Amazon EC2 Auto Scaling group launch configuration to configure Amazon EC2 instances for IMDSv2

This control checks whether an Amazon EC2 Auto Scaling launch configuration is configured to require the use of Instance Metadata Service Version 2 (IMDSv2).

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::LaunchConfiguration`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.2 rule specification
- For examples of PASS and FAIL CloudFront Templates related to this control, see: CT.AUTOSCALING.PR.2 example templates

Explanation

IMDS provides data about your instance, which you can use to configure or manage the running instance.

Version 2 of the IMDS adds protections that weren't available in IMDSv1, to safeguard your EC2 instances further.

Usage considerations

- This control applies only to Amazon EC2 Auto Scaling launch configurations that allow access to instance metadata.

Remediation for rule failure

Provide a `MetadataOptions` configuration and set the value of `HttpTokens` to `required`.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Launch Configuration - Example

Amazon EC2 Auto Scaling launch configuration with IMDSv2 enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingLaunchConfiguration": {
        "Type": "AWS::AutoScaling::LaunchConfiguration",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "t3.micro",
            "MetadataOptions": {
                "HttpTokens": "required"
            }
        }
    }
}
```

YAML example

```
AutoScalingLaunchConfiguration:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: t3.micro
    MetadataOptions:
      HttpTokens: required
```

CT.AUTOSCALING.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_launch_config_requires_imdsv2_check
#
# Description:
```

```
#   This control checks whether an Amazon EC2 Auto Scaling launch configuration is
 configured to require the use of Instance Metadata Service Version 2 (IMDSv2).
#
# Reports on:
#   AWS::AutoScaling::LaunchConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#  Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Autoscaling launch configuration
 resources
#      Then: SKIP
#  Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Autoscaling launch configuration resource
#       And: 'MetadataOptions' has been provided.
#       And: 'MetadataOptions.HttpEndpoint' has been provided is equal to 'disabled'
#      Then: SKIP
#  Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Autoscaling launch configuration resource
#       And: 'MetadataOptions.HttpEndpoint' has not been provided or has been provided
 and is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' has not been provided
#      Then: FAIL
#  Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Autoscaling launch configuration resource
#       And: 'MetadataOptions' has been provided.
#       And: 'MetadataOptions.HttpEndpoint' has not been provided or has been provided
 and is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' has been provided and set to a value other
 than 'required'
#      Then: FAIL
#  Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Autoscaling launch configuration resource
#       And: 'MetadataOptions' has been provided.
```

```
#       And: 'MetadataOptions.HttpEndpoint' has not been provided or has been provided
 and is equal to 'enabled'
#       And: 'MetadataOptions.HttpTokens' has been provided and set to 'required'
#      Then: PASS

#
# Constants
#
let AUTOSCALING_LAUNCH_CONFIGURATION_TYPE = "AWS::AutoScaling::LaunchConfiguration"
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_launch_configurations = Resources.*[ Type ==
 %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule autoscaling_launch_config_requires_imdsv2_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %autoscaling_launch_configurations not empty {
    check(%autoscaling_launch_configurations.Properties)
        <<
        [CT.AUTOSCALING.PR.2]: Require an Amazon EC2 Auto Scaling group launch
 configuration to configure Amazon EC2 instances for IMDSv2
            [FIX]: Provide a 'MetadataOptions' configuration and set the value of
 'HttpTokens' to 'required'.
        >>
}

rule autoscaling_launch_config_requires_imdsv2_check when is_cfn_hook(%INPUT_DOCUMENT,
 %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_LAUNCH_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.2]: Require an Amazon EC2 Auto Scaling group launch
 configuration to configure Amazon EC2 instances for IMDSv2
            [FIX]: Provide a 'MetadataOptions' configuration and set the value of
 'HttpTokens' to 'required'.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(autoscaling_launch_configuration) {
    %autoscaling_launch_configuration [
        # Scenario 2
        filter_autoscaling_launch_configurations(this)
    ] {
        # Scenario 3, 4 and 5
        MetadataOptions exists
        MetadataOptions is_struct

        MetadataOptions {
            HttpTokens exists
            HttpTokens == "required"
        }
    }
}

rule filter_autoscaling_launch_configurations(autoscaling_launch_configurations) {
    %autoscaling_launch_configurations {
        MetadataOptions not exists or
        filter_metadata_options(this)
    }
}

rule filter_metadata_options(metadata_options) {
    %metadata_options {
        MetadataOptions is_struct
        MetadataOptions {
            HttpEndpoint not exists or
            HttpEndpoint == "enabled"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
```

```
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.AUTOSCALING.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  AutoScalingLaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      MetadataOptions:
        HttpTokens: required
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
```

```
Resources:
  AutoScalingLaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      MetadataOptions:
        HttpTokens: optional
```

[CT.AUTOSCALING.PR.4] Require an Amazon EC2 Auto Scaling group associated with an AWS Elastic Load Balancing (ELB) to have ELB health checks activated

This control checks whether your Amazon EC2 Auto Scaling groups that are associated with a load balancer are using Elastic Load Balancing health checks.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.4 example templates

Explanation

This configuration requirement ensures that the group can determine an instance's health based on additional tests provided by the load balancer. Using Elastic Load Balancing health checks can help support the availability of applications that use EC2 Auto Scaling groups.

Usage considerations

- This control only applies to Auto Scaling groups associated with a Classic Load Balancer or Target Group

Remediation for rule failure

Configure Amazon EC2 Auto Scaling groups associated with an Elastic Load Balancing to use Elastic Load Balancing health checks.

The examples that follow show how to implement this remediation.

Auto Scaling group - Example One

Auto Scaling group with a Classic Load Balancer association and Elastic Load Balancing health checks. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "LaunchTemplate": {
                "LaunchTemplateId": {
                    "Ref": "LaunchTemplate"
                },
                "Version": {
                    "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                }
            },
            "MaxSize": "1",
            "MinSize": "0",
            "DesiredCapacity": "1",
            "LoadBalancerNames": [
                {
                    "Ref": "ElasticLoadBalancer"
                }
            ],
            "HealthCheckType": "ELB",
```

```
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref 'LaunchTemplate'
      Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
    MaxSize: '1'
    MinSize: '0'
    DesiredCapacity: '1'
    LoadBalancerNames:
      - !Ref 'ElasticLoadBalancer'
    HealthCheckType: ELB
    VPCZoneIdentifier:
      - !Ref 'Subnet'
```

The examples that follow show how to implement this remediation.

Auto Scaling group - Example Two

Auto Scaling group with a Target Group association and Elastic Load Balancing health checks. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
```

```
        "Properties": {
            "LaunchTemplate": {
                "LaunchTemplateId": {
                    "Ref": "LaunchTemplate"
                },
                "Version": {
                    "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                }
            },
            "MaxSize": "1",
            "MinSize": "0",
            "DesiredCapacity": "1",
            "TargetGroupARNs": [
                {
                    "Ref": "ELBv2TargetGroup"
                }
            ],
            "HealthCheckType": "ELB",
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref 'LaunchTemplate'
      Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
    MaxSize: '1'
    MinSize: '0'
    DesiredCapacity: '1'
    TargetGroupARNs:
      - !Ref 'ELBv2TargetGroup'
    HealthCheckType: ELB
```

```
    VPCZoneIdentifier:
      - !Ref 'Subnet'
```

CT.AUTOSCALING.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: example-launch-template-id
        Version: 1
      MaxSize: '1'
      MinSize: '0'
      DesiredCapacity: '1'
      LoadBalancerNames:
        - example-load-balancer
      HealthCheckType: ELB
      VPCZoneIdentifier:
        - example-subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: example-launch-template-id
        Version: 1
      MaxSize: '1'
```

```
      MinSize: '0'
      DesiredCapacity: '1'
      TargetGroupARNs:
        - example-target-group-arn
      VPCZoneIdentifier:
        - example-subnet
      HealthCheckType: EC2
```

CT.AUTOSCALING.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_group_elb_healthcheck_required_check
#
# Description:
#   This control checks whether your Amazon EC2 Auto Scaling groups that are associated
 with a load balancer are using
#   Elastic Load Balancing health checks.
#
# Reports on:
#   AWS::AutoScaling::AutoScalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Auto Scaling groups
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Auto Scaling group resource
#       And: 'LoadBalancerNames' or 'TargetGroupARNs' are not present on the Auto
 Scaling group resource or empty lists
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Auto Scaling group resource
#       And: 'LoadBalancerNames' or 'TargetGroupARNs' are present on the Auto Scaling
 group with at least
#            one configuration
#       And: 'HealthCheckType' is not present
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Auto Scaling group resource
#       And: 'LoadBalancerNames' or 'TargetGroupARNs' are present on the Auto Scaling
 group with at least
#            one configuration
#       And: 'HealthCheckType' is present and set to a value other than 'ELB' (e.g.
 'EC2')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Auto Scaling group resource
#       And: 'LoadBalancerNames' or 'TargetGroupARNs' are present on the Auto Scaling
 group resource with at least
#            one configuration
#       And: 'HealthCheckType' is present and set to 'ELB'
#      Then: PASS

#
# Constants
#
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]

#
# Primary Rules
#
rule autoscaling_group_elb_healthcheck_required_check when
 is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                                %autoscaling_groups not
 empty {
    check(%autoscaling_groups.Properties)
        <<
        [CT.AUTOSCALING.PR.4]: Require an Amazon Elastic Compute Cloud Auto Scaling
 group associated with an Elastic Load Balancing to have ELB health checks activated
        [FIX]: Configure Amazon Elastic Compute Cloud Auto Scaling groups associated
 with an Elastic Load Balancing load balancer to use Elastic Load Balancing health
 checks.
        >>
}

rule autoscaling_group_elb_healthcheck_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %AUTOSCALING_GROUP_TYPE) {
    check(this.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.4]: Require an Amazon Elastic Compute Cloud Auto Scaling
 group associated with an Elastic Load Balancing to have ELB health checks activated
        [FIX]: Configure Amazon Elastic Compute Cloud Auto Scaling groups associated
 with an Elastic Load Balancing load balancer to use Elastic Load Balancing health
 checks.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_group) {
    %autoscaling_group [
        filter_list_exists_and_not_empty(LoadBalancerNames) or
        filter_list_exists_and_not_empty(TargetGroupARNs)
    ] {
       check_healthcheck_type_on_asg(this)
    }
}

rule filter_list_exists_and_not_empty(property) {
    %property {
        # Scenario 2
        this exists
        this is_list
        this not empty
    }
}
```

```
rule check_healthcheck_type_on_asg(asg) {
    %asg {
        # Scenario 3
        HealthCheckType exists
        # Scenario 4 and 5
        HealthCheckType == "ELB"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

[CT.AUTOSCALING.PR.5] Require that an Amazon EC2 Auto Scaling group launch configuration does not have Amazon EC2 instances with public IP addresses

This control checks whether Amazon EC2 Auto Scaling groups have public IP addresses configured through Launch Configurations.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::LaunchConfiguration`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.5 example templates

Explanation

Amazon EC2 instances in an Auto Scaling group launch configuration should not have an associated public IP address, except for in limited edge cases. Amazon EC2 instances should only be accessible from behind a load balancer instead of being directly exposed to the internet.

Remediation for rule failure

Set `AssociatePublicIpAddress` to false on Auto Scaling Launch Configurations.

The examples that follow show how to implement this remediation.

Auto Scaling Launch Configuration - Example

Auto Scaling Launch Configuration configured to disable public IP address association. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingLaunchConfiguration": {
        "Type": "AWS::AutoScaling::LaunchConfiguration",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "t3.micro",
            "AssociatePublicIpAddress": false
        }
    }
}
```

YAML example

```
AutoScalingLaunchConfiguration:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: t3.micro
    AssociatePublicIpAddress: false
```

CT.AUTOSCALING.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_launch_config_public_ip_disabled_check
#
# Description:
#   Checks if Auto Scaling Launch Configurations have been configured to disable public
 IP address association.
#
# Reports on:
#   AWS::Auto Scaling::LaunchConfiguration
#
# Evaluates:
#   AWS CloudFormation, AWS CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document does not contain any Auto Scaling Launch Configuration
 Resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
```

```
#       And: The input document contains an Auto Scaling Launch Configuration Resource
#       And: 'AssociatePublicIpAddress' is not present on the Auto Scaling Launch
 Configuration Resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an Auto Scaling Launch Configuration Resource
#       And: 'AssociatePublicIpAddress' is present on the Auto Scaling Launch
 Configuration Resource
#            and is set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an Auto Scaling Launch Configuration Resource
#       And: 'AssociatePublicIpAddress' is present on the Auto Scaling Launch
 Configuration Resource
#            and is set to bool(false)
#      Then: PASS

#
# Constants
#
let AUTOSCALING_LAUNCH_CONFIGURATION_TYPE = 'AWS::AutoScaling::LaunchConfiguration'
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_launch_configurations = Resources.*[ Type ==
 %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule autoscaling_launch_config_public_ip_disabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %autoscaling_launch_configurations not empty {

    check(%autoscaling_launch_configurations.Properties)
        <<
```

```
        [CT.AUTOSCALING.PR.5]: Require than an Amazon EC2 Auto Scaling group launch
 configuration does not have EC2 instances with public IP addresses
        [FIX]: Set 'AssociatePublicIpAddress' to false on Auto Scaling Launch
 Configurations.
        >>

}

rule autoscaling_launch_config_public_ip_disabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE) {

    check(%INPUT_DOCUMENT.%AUTOSCALING_LAUNCH_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.5]: Require than an Amazon EC2 Auto Scaling group launch
 configuration does not have EC2 instances with public IP addresses
        [FIX]: Set 'AssociatePublicIpAddress' to false on Auto Scaling Launch
 Configurations.
        >>
}

#
# Parameterized Rules
#
rule check(launch_configuration) {
    %launch_configuration {
        # Scenario 2
        AssociatePublicIpAddress exists
        # Scenarios 3 and 4
        AssociatePublicIpAddress == false
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.AUTOSCALING.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  AutoScalingLaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      AssociatePublicIpAddress: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  AutoScalingLaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
```

```
      InstanceType: t3.micro
      AssociatePublicIpAddress: true
```

[CT.AUTOSCALING.PR.6] Require any Amazon EC2 Auto Scaling groups to use multiple instance types

This control checks whether an Amazon EC2 Auto Scaling group uses multiple instance types through a mixed instance policy and explicit instance type overrides.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.6 example templates

Explanation

You can enhance availability by deploying your application across multiple instance types running in multiple Availability Zones. AWS Control Tower recommends using multiple instance types so that the Auto Scaling group can launch another instance type if there is insufficient instance capacity in your chosen Availability Zones.

Usage considerations

- This control applies only to Amazon EC2 Auto Scaling groups that do not use attribute-based instance type selection within a mixed instances policy (configured by means of the `InstanceRequirements` property within mixed instances policy `Overrides`).

Remediation for rule failure

Within a `MixedInstancePolicy` configuration, provide a `LaunchTemplate` configuration with two entries in the `Overrides` property. Within each override, set the `InstanceType` property to a different Amazon EC2 instance type.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example

Amazon EC2 Auto Scaling group configured with multiple instance types. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ],
            "MaxSize": "2",
            "MinSize": "1",
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "EC2LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "EC2LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceType": "t3.micro"
                        },
```

```
                        {
                            "InstanceType": "m5.large"
                        }
                    ]
                }
            }
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref 'Subnet'
    MaxSize: '2'
    MinSize: '1'
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'EC2LaunchTemplate'
          Version: !GetAtt 'EC2LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceType: t3.micro
          - InstanceType: m5.large
```

CT.AUTOSCALING.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_mixed_instances_policy_multiple_instance_types_check
#
# Description:
```

```
#   This control checks whether an Amazon EC2 Auto Scaling group uses multiple instance
 types through a mixed instance policy and explicit instance type overrides.
#
# Reports on:
#   AWS::AutoScaling::AutoscalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#  Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Autoscaling Group resources
#      Then: SKIP
#  Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has been provided as a
 list
#       And: There exists any 'Overrides' entry where 'InstanceRequirements' is present
#       And: There exists no 'Overrides' entry where 'InstanceType' is present
#      Then: SKIP
#  Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has not been provided
#      Then: FAIL
#  Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has been provided as a
 list
#       And: 'InstanceType' is not present or is present as a empty string in
 'MixedInstancesPolicy.LaunchTemplate.Overrides'
#      Then: FAIL
#  Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has been provided as a
 list
#       And: There exists any 'Overrides' entry where 'InstanceRequirements' is present
#       And: There exists any 'Overrides' entry where 'InstanceType' is present
#      Then: FAIL
#  Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has been provided as a
 list
#       And: 'InstanceType' is present in
 'MixedInstancesPolicy.LaunchTemplate.Overrides' as a non empty string
#       And: Length of 'MixedInstancesPolicy.LaunchTemplate.Overrides' is less than or
 equal to 1
#      Then: FAIL
#  Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Autoscaling Group resources
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.Overrides' has been provided as a
 list
#       And: 'InstanceType' is present in
 'MixedInstancesPolicy.LaunchTemplate.Overrides' as a non empty string
#       And: Length of 'MixedInstancesPolicy.LaunchTemplate.Overrides' is greater than
 1
#      Then: PASS

#
# Constants
#
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]
```

```
#
# Primary Rules
#
rule autoscaling_mixed_instances_policy_multiple_instance_types_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %autoscaling_groups not empty {
    check(%autoscaling_groups.Properties)
        <<
        [CT.AUTOSCALING.PR.6]: Require any Amazon EC2 Auto Scaling groups to use
 multiple instance types
            [FIX]: Within a 'MixedInstancePolicy' configuration, provide a
 'LaunchTemplate' configuration with two entries in the 'Overrides' property. Within
 each override, set the 'InstanceType' property to a different Amazon EC2 instance
 type.
        >>
}

rule autoscaling_mixed_instances_policy_multiple_instance_types_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AUTOSCALING_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.6]: Require any Amazon EC2 Auto Scaling groups to use
 multiple instance types
            [FIX]: Within a 'MixedInstancePolicy' configuration, provide a
 'LaunchTemplate' configuration with two entries in the 'Overrides' property. Within
 each override, set the 'InstanceType' property to a different Amazon EC2 instance
 type.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_group) {
    %autoscaling_group [
        # Scenario 2
        filter_asg_no_instance_requirement_overrides(this)
    ] {
        # Scenario 4, 5, 6
        MixedInstancesPolicy exists
        MixedInstancesPolicy is_struct

        MixedInstancesPolicy {
```

```
            LaunchTemplate exists
            LaunchTemplate is_struct

            LaunchTemplate {
                LaunchTemplateSpecification exists
                LaunchTemplateSpecification is_struct

                Overrides exists
                Overrides is_list
                Overrides not empty

                Overrides[0] exists
                Overrides[1] exists

                Overrides[*] {
                    InstanceType exists
                    check_is_string_and_not_empty(InstanceType)
                }

                Overrides[0].InstanceType not in Overrides[1].InstanceType
            }
        }
    }

    %autoscaling_group [
        # Scenario 2
        filter_asg_conflicting_overrides(this)
    ] {
        MixedInstancesPolicy {
            LaunchTemplate {
                Overrides[*] {
                    check_mutually_exclusive_property_combination(InstanceType,
 InstanceRequirements) or
                    check_mutually_exclusive_property_combination(InstanceRequirements,
 InstanceType)
                }
            }
        }
    }
}

rule filter_asg_no_instance_requirement_overrides(autoscaling_group) {
    %autoscaling_group {
        MixedInstancesPolicy not exists or
```

```
        filter_mixed_instances_policy_no_instance_requirement_overrides(this)
    }
}

rule filter_mixed_instances_policy_no_instance_requirement_overrides(autoscaling_group)
 {
    %autoscaling_group {
        MixedInstancesPolicy is_struct
        MixedInstancesPolicy {
            LaunchTemplate not exists or
            filter_launch_templates_no_instance_requirement_overrides(this)
        }
    }
}

rule filter_launch_templates_no_instance_requirement_overrides(launch_template) {
    %launch_template {
        LaunchTemplate is_struct
        LaunchTemplate {
            Overrides not exists or
            filter_overrides_no_instance_requirement_overrides(this)
        }
    }
}

rule filter_overrides_no_instance_requirement_overrides(overrides) {
    %overrides {
        Overrides is_list
        Overrides empty or
        Overrides[*] {
            InstanceRequirements not exists
        }
    }
}

rule filter_asg_conflicting_overrides(autoscaling_group) {
    %autoscaling_group {
        MixedInstancesPolicy not exists or
        filter_mixed_instances_policy_conflicting_overrides(this)
    }
}

rule filter_mixed_instances_policy_conflicting_overrides(autoscaling_group) {
    %autoscaling_group {
```

```
        MixedInstancesPolicy is_struct
        MixedInstancesPolicy {
            LaunchTemplate not exists or
            filter_launch_templates_conflicting_overrides(this)
        }
    }
}

rule filter_launch_templates_conflicting_overrides(launch_template) {
    %launch_template {
        LaunchTemplate is_struct
        LaunchTemplate {
            Overrides not exists or
            filter_overrides_conflicting_overrides(this)
        }
    }
}

rule filter_overrides_conflicting_overrides(overrides) {
    %overrides {
        Overrides is_list
        Overrides empty or
        some Overrides[*] {
            InstanceRequirements exists
            InstanceType exists
        }
    }
}

rule check_mutually_exclusive_property_combination(property1, property2) {
    %property1 not exists
    %property2 exists
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.AUTOSCALING.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName:
        Fn::Sub: ${AWS::StackName}-example
      LaunchTemplateData:
        InstanceType: t3.micro
        ImageId:
          Ref: LatestAmiId
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
```

```
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
      MaxSize: '2'
      MinSize: '1'
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: EC2LaunchTemplate
            Version:
              Fn::GetAtt:
              - EC2LaunchTemplate
              - LatestVersionNumber
          Overrides:
          - InstanceType: t3.micro
          - InstanceType: m5.large
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
```

```
    Properties:
      LaunchTemplateName:
        Fn::Sub: ${AWS::StackName}-example
      LaunchTemplateData:
        InstanceType: t3.micro
        ImageId:
          Ref: LatestAmiId
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
      MaxSize: '2'
      MinSize: '1'
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: EC2LaunchTemplate
            Version:
              Fn::GetAtt:
              - EC2LaunchTemplate
              - LatestVersionNumber
          Overrides:
          - InstanceType: t3.micro
```

[CT.AUTOSCALING.PR.8] Require an Amazon EC2 Auto Scaling group to have EC2 launch templates configured

This control checks whether an Amazon EC2 Auto Scaling group is configured to use an EC2 launch template.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.8 example templates

Explanation

An Auto Scaling group can be created from an EC2 launch template or from a launch configuration. If you use a launch template to create an Auto Scaling group, you have access to the latest features and improvements.

Remediation for rule failure

Provide a `LaunchTemplate` or `MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification` configuration with a valid `Version` and a `LaunchTemplateId` or `LaunchTemplateName`.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example One

Amazon EC2 Auto Scaling group configured with an EC2 launch template. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ],
            "MaxSize": "2",
            "MinSize": "1",
            "LaunchTemplate": {
                "LaunchTemplateName": "SampleLaunchTemplate",
                "Version": {
                    "Fn::GetAtt": [
                        "EC2LaunchTemplate",
                        "LatestVersionNumber"
                    ]
                }
            }
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref 'Subnet'
    MaxSize: '2'
    MinSize: '1'
    LaunchTemplate:
      LaunchTemplateName: SampleLaunchTemplate
      Version: !GetAtt 'EC2LaunchTemplate.LatestVersionNumber'
```

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example Two

Amazon EC2 Auto Scaling group configured with a mixed instances policy and EC2 launch template. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ],
            "MaxSize": "2",
            "MinSize": "1",
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "EC2LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": [
                                "EC2LaunchTemplate",
                                "LatestVersionNumber"
                            ]
                        }
                    }
                }
            }
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref 'Subnet'
    MaxSize: '2'
    MinSize: '1'
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'EC2LaunchTemplate'
          Version: !GetAtt 'EC2LaunchTemplate.LatestVersionNumber'
```

CT.AUTOSCALING.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_launch_template_check
#
# Description:
#   This control checks whether an Amazon EC2 Auto Scaling group is configured to use
 an EC2 launch template.
#
# Reports on:
#   AWS::AutoScaling::AutoscalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#  Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contains any Amazon EC2 Auto Scaling group
 resources
#      Then: SKIP
#  Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling group resource
#       And: 'LaunchTemplate' has not been provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has not
 been provided
#      Then: FAIL
#  Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling group resource
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has not
 been provided
#       And: 'LaunchTemplate' has been provided
#       And: 'LaunchTemplate' has an invalid configuration ('Version' has not been
 provided and one of 'LaunchTemplateId'
#            or 'LaunchTemplateName' has not been provided or 'Version' has been
 provided as an empty string or invalid local
#            reference and one of 'LaunchTemplateId' or 'LaunchTemplateName' has been
 provided as an empty string or an
#            invalid local reference)
#      Then: FAIL
#  Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling group resource
#       And: 'LaunchTemplate' has not been provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has been
 provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has an
 invalid configuration ('Version' has
#            not been provided and one of 'LaunchTemplateId' or 'LaunchTemplateName'
 has not been provided or 'Version' has
#            been provided as an empty string or invalid local reference and one of
 'LaunchTemplateId' or 'LaunchTemplateName'
#            has been provided as an empty string or an invalid local reference)
#      Then: FAIL
#  Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling group resource
#       And: 'LaunchTemplate' has been provided
#       And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has
#            been provided.
```

```
#       Then: FAIL
#   Scenario: 6
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an Amazon EC2 Auto Scaling group resource
#        And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has not
 been provided
#        And: 'LaunchTemplate' has been provided
#        And: 'LaunchTemplate' has a valid configuration ('Version' has been provided
 and one of 'LaunchTemplateId' or
#             'LaunchTemplateName' has been provided as a non-empty string or valid
 local reference)
#       Then: PASS
#   Scenario: 7
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an Amazon EC2 Auto Scaling group resource
#        And: 'LaunchTemplate' has not been provided
#        And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has
 been provided
#        And: 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' has a
 valid configuration ('Version' has
#             been provided and one of 'LaunchTemplateId' or 'LaunchTemplateName' has
 been provided as a non-empty string
#             or valid local reference)
#       Then: PASS

#
# Constants
#
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]

#
# Primary Rules
#
rule autoscaling_launch_template_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %autoscaling_groups not empty {
    check(%autoscaling_groups.Properties)
        <<
```

```
        [CT.AUTOSCALING.PR.8]: Require an Amazon EC2 Auto Scaling group to have EC2
 launch templates configured
            [FIX]: Provide a 'LaunchTemplate' or
 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' configuration with a
 valid 'Version' and a 'LaunchTemplateId' or 'LaunchTemplateName'.
        >>
}

rule autoscaling_launch_template_check when is_cfn_hook(%INPUT_DOCUMENT,
 %AUTOSCALING_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.8]: Require an Amazon EC2 Auto Scaling group to have EC2
 launch templates configured
            [FIX]: Provide a 'LaunchTemplate' or
 'MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification' configuration with a
 valid 'Version' and a 'LaunchTemplateId' or 'LaunchTemplateName'.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_groups) {
    %autoscaling_groups {
        # Scenario 3 and 6
        check_launch_template(this) or

        # Scenario 4 and 7
        check_mixed_instances_policy(this)
    }
}

rule check_launch_template(autoscaling_groups) {
    %autoscaling_groups {
        check_mutually_exclusive_property_combination(LaunchTemplate,
 MixedInstancesPolicy)
        LaunchTemplate is_struct
        LaunchTemplate {
            check_valid_launch_template_config(this)
        }
    }
}
```

```
rule check_mixed_instances_policy(autoscaling_groups) {
    %autoscaling_groups {
        check_mutually_exclusive_property_combination(MixedInstancesPolicy,
 LaunchTemplate)
        MixedInstancesPolicy is_struct
        MixedInstancesPolicy {
            LaunchTemplate exists
            LaunchTemplate is_struct
            LaunchTemplate {
                LaunchTemplateSpecification exists
                LaunchTemplateSpecification is_struct
                check_valid_launch_template_config(LaunchTemplateSpecification)
            }
        }
    }
}

rule check_valid_launch_template_config(launch_template_specification) {
    %launch_template_specification {
        check_valid_launch_template_property(Version)
        check_valid_prop_combination(LaunchTemplateId, LaunchTemplateName) or
        check_valid_prop_combination(LaunchTemplateName, LaunchTemplateId)
    }
}

rule check_valid_prop_combination(valid_property, invalid_property) {
    check_mutually_exclusive_property_combination(%valid_property, %invalid_property)
    check_valid_launch_template_property(%valid_property)
}

rule check_mutually_exclusive_property_combination(valid_property, invalid_property) {
    %invalid_property not exists
    %valid_property exists
}

rule check_valid_launch_template_property(property) {
    %property {
        check_is_string_and_not_empty(this) or
        check_local_references(%INPUT_DOCUMENT, this, "AWS::EC2::LaunchTemplate")
    }
}

rule check_is_string_and_not_empty(value) {
    %value {
```

```
        this is_string
        this != /\A\s*\z/
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.AUTOSCALING.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  EC2LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName:
        Fn::Sub: ${AWS::StackName}-example
      LaunchTemplateData:
        InstanceType: t3.micro
        ImageId:
          Ref: LatestAmiId
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
```

```
      MaxSize: '2'
      MinSize: '1'
      LaunchTemplate:
        LaunchTemplateId:
          Ref: EC2LaunchTemplate
        Version:
          Fn::GetAtt:
          - EC2LaunchTemplate
          - LatestVersionNumber
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  AutoScalingLaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
```

```
      InstanceType: t3.micro
      LaunchConfigurationName: "AutoScalingLaunchConfiguration"
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
      MaxSize: '2'
      MinSize: '1'
      LaunchConfigurationName: "AutoScalingLaunchConfiguration"
```

[CT.AUTOSCALING.PR.9] Require an Amazon EBS volume configured through an Amazon EC2 Auto Scaling launch configuration to encrypt data at rest

This control checks whether Auto Scaling launch configurations with Amazon EBS volume block device mappings enable Amazon EBS volume encryption.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::LaunchConfiguration`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.9 example templates

Explanation

Enable Amazon EBS encryption at rest, to provide an added layer of security for your sensitive data in Amazon EBS volumes. Amazon EBS encryption offers a straightforward encryption solution for your Amazon EBS resources. It doesn't require you to build, maintain, and secure your own key management infrastructure. It uses KMS keys when creating encrypted volumes and snapshots.

Usage considerations

- This control applies only to Amazon EC2 Auto Scaling launch configurations that specify Amazon EBS block device mappings.

Remediation for rule failure

For every entry in the BlockDeviceMappings parameter with an `Ebs` configuration, set the value of `Encryption` to true.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Launch Configuration - Example

An Amazon EC2 Auto Scaling launch configuration configured with an Amazon EBS block device mapping that has volume encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "LaunchConfiguration": {
        "Type": "AWS::AutoScaling::LaunchConfiguration",
        "Properties": {
            "ImageId": {
                "Ref": "LatestAmiId"
            },
            "InstanceType": "t3.micro",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sdc",
                    "Ebs": {
                        "Encrypted": true,
                        "VolumeSize": 100,
                        "VolumeType": "gp3"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
LaunchConfiguration:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    ImageId: !Ref 'LatestAmiId'
    InstanceType: t3.micro
    BlockDeviceMappings:
      - DeviceName: /dev/sdc
        Ebs:
          Encrypted: true
          VolumeSize: 100
          VolumeType: gp3
```

CT.AUTOSCALING.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   autoscaling_launch_config_encrypted_volumes_check
#
# Description:
#   This control checks whether Auto Scaling launch configurations with Amazon EBS
 volume block device mappings enable Amazon EBS volume encryption.
#
# Reports on:
#   AWS::AutoScaling::LaunchConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 Auto Scaling launch
 configuration resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling launch
 configuration resource
#       And: 'BlockDeviceMappings' has not been provided or has been provided as an
 empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling launch
 configuration resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: No entries in 'BlockDeviceMappings' contain 'Ebs' as a struct
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling launch
 configuration resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: An entry in 'BlockDeviceMappings' contains 'Ebs' as a struct
#       And: In the same entry, 'Encrypted' in 'Ebs' has not been provided or has been
 provided
#            and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 Auto Scaling launch
 configuration resource
#       And: 'BlockDeviceMappings' has been provided as a non-empty list
#       And: An entry in 'BlockDeviceMappings' contains 'Ebs' as a struct
#       And: In the same entry, 'Encrypted' in 'Ebs' has been provided and set to
 bool(true)
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let AUTOSCALING_LAUNCH_CONFIGURATION_TYPE = "AWS::AutoScaling::LaunchConfiguration"
```

```
#
# Assignments
#
let autoscaling_launch_configurations = Resources.*[ Type ==
 %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule autoscaling_launch_config_encrypted_volumes_check when is_cfn_template(this)

 %autoscaling_launch_configurations not empty {
    check(%autoscaling_launch_configurations.Properties)
        <<
        [CT.AUTOSCALING.PR.9]: Require an Amazon EBS volume configured through an
 Amazon EC2 Auto Scaling launch configuration to encrypt data at rest
        [FIX]: For every entry in the BlockDeviceMappings parameter with an 'Ebs'
 configuration, set the value of 'Encryption' to true.
        >>
}

rule autoscaling_launch_config_encrypted_volumes_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AUTOSCALING_LAUNCH_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_LAUNCH_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.9]: Require an Amazon EBS volume configured through an
 Amazon EC2 Auto Scaling launch configuration to encrypt data at rest
        [FIX]: For every entry in the BlockDeviceMappings parameter with an 'Ebs'
 configuration, set the value of 'Encryption' to true.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_launch_configuration) {
    %autoscaling_launch_configuration [
        # Scenarios 2 and 3
        filter_launch_configuration_contains_ebs_block_device_mappings(this)
    ] {
        BlockDeviceMappings[
            Ebs exists
            Ebs is_struct
```

```
        ] {
            Ebs {
                # Scenarios 4 and 5
                Encrypted exists
                Encrypted == true
            }
        }
    }
}

rule
 filter_launch_configuration_contains_ebs_block_device_mappings(launch_configuration) {
    %launch_configuration {
        BlockDeviceMappings exists
        BlockDeviceMappings is_list
        BlockDeviceMappings not empty

        some BlockDeviceMappings[*] {
            Ebs exists
            Ebs is_struct
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.AUTOSCALING.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
        Ref: LatestAmiId
      InstanceType: t3.micro
      BlockDeviceMappings:
      - DeviceName: /dev/sdc
        Ebs:
          Encrypted: true
          VolumeSize: 100
          VolumeType: gp3
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value "AWS::EC2::Image::Id"
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  LaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId:
```

```
        Ref: LatestAmiId
      InstanceType: t3.micro
      BlockDeviceMappings:
      - DeviceName: /dev/sdc
        Ebs:
          Encrypted: false
          VolumeSize: 100
          VolumeType: gp3
```

[CT.AUTOSCALING.PR.10] Require an Amazon EC2 Auto Scaling group to use only AWS Nitro instance types when overriding a launch template

This control checks whether, when using a `MixedInstancesPolicy` resource parameter override, an Amazon EC2 Auto Scaling group overrides launch templates by specifying AWS Nitro instance types only.

- **Control objective:** Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.10 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, thereby minimizing the

attack surface. The Nitro System security model is locked down to prohibit administrative access, greatly reducing the possibility of human error and tampering.

Usage considerations

- This control applies only to Amazon EC2 Auto Scaling groups that are configured with launch template overrides that specify an instance type or instance attributes. A `LaunchTemplate.LaunchTemplateSpecification` configuration specifies one or more `Overrides` that also include `InstanceType` or `InstanceRequirements`.
- This control does not check the instance type configured on a launch template. To ensure that launch templates use Nitro instances types, use this control in conjunction with related controls that check launch templates for Nitro instance types.

Remediation for rule failure

In the MixedInstancesPolicy.LaunchTemplate property, if it has one or more `Overrides` fields that include `InstanceType` or `InstanceRequirements`, set the value of `InstanceType` to an Amazon EC2 instance type that is based on the AWS Nitro system, or set the value of `AllowedInstanceTypes` in `InstanceRequirements` to one or more Amazon EC2 instance types that are based on the AWS Nitro system.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example One

An Amazon EC2 Auto Scaling group configured with a launch template override and instance type based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
```

```
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceType": "t3.micro"
                        }
                    ]
                }
            },
            "MaxSize": 1,
            "MinSize": 0,
            "DesiredCapacity": 1,
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceType: t3.micro
    MaxSize: 1
    MinSize: 0
    DesiredCapacity: 1
    VPCZoneIdentifier:
```

```
      - !Ref 'Subnet'
```

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example Two

An Amazon EC2 Auto Scaling group configured with a launch template override and instance requirements that specify a list of allowed instances based on the AWS Nitro system. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceRequirements": {
                                "AllowedInstanceTypes": [
                                    "m5.*",
                                    "c5.*"
                                ],
                                "VCpuCount": {
                                    "Min": 2,
                                    "Max": 4
                                },
                                "MemoryMiB": {
                                    "Min": 4000,
                                    "Max": 8000
```

```
                                }
                            }
                        }
                    ]
                }
            },
            "MaxSize": 1,
            "MinSize": 0,
            "DesiredCapacity": 1,
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
                - m5.*
                - c5.*
              VCpuCount:
                Min: 2
                Max: 4
              MemoryMiB:
                Min: 4000
                Max: 8000
    MaxSize: 1
    MinSize: 0
```

```
    DesiredCapacity: 1
    VPCZoneIdentifier:
      - !Ref 'Subnet'
```

CT.AUTOSCALING.PR.10 rule specification

```
# ###################################
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   autoscaling_group_nitro_instance_override_check
#
# Description:
#   This control checks whether, when using a MixedInstancesPolicy resource parameter
 override, an Amazon EC2 Auto Scaling group overrides launch templates with AWS Nitro
 instance types only.
#
# Reports on:
#   AWS::AutoScaling::AutoScalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 auto scaling group
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has not been provided
 or
#            has been provided as an empty list
#      Then: SKIP
```

```
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: No entries in 'Overrides' include 'InstanceType' or 'InstanceRequirements'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to
 an instance type
#            other than a Nitro instance type
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided or has been
#            provided as an empty list
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: An entry in 'AllowedInstanceTypes' is set to an instance type other than a
 Nitro instance type
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to a
 Nitro instance type
```

```
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: Every entry in 'AllowedInstanceTypes' is set to a Nitro instance type
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let NITRO_INSTANCE_TYPES = [
    /^a1\..+?$/,
    /^c5\..+?$/, /^c5a\..+?$/, /^c5ad\..+?$/, /^c5d\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/,
    /^c6g\..+?$/, /^c6gd\..+?$/, /^c6gn\..+?$/, /^c6i\..+?$/, /^c6id\..+?$/, /^c6in\..
+?$/,
    /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..+?$/, /^c7i-flex\..+?$/,
    /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g5g\..+?$/, /^g6\..+?$/, /^g6e\..+?$/,
    /^g6f\..+?$/, /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3\.metal$/, /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?
$/,
    /^i8g\..+?$/, /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5\..+?$/, /^m5a\..+?$/, /^m5ad\..+?$/, /^m5d\..+?$/, /^m5dn\..+?$/, /^m5n\..+?
$/,
    /^m5zn\..+?$/, /^m6a\..+?$/, /^m6g\..+?$/, /^m6gd\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/, /^mac1\.metal$/,
    /^mac2-m1ultra\.metal$/, /^mac2-m2\.metal$/, /^mac2-m2pro\.metal$/, /^mac2\.metal
$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
```

```
    /^r5\..+?$/, /^r5a\..+?$/, /^r5ad\..+?$/, /^r5b\..+?$/, /^r5d\..+?$/, /^r5dn\..+?
$/,
    /^r5n\..+?$/, /^r6a\..+?$/, /^r6g\..+?$/, /^r6gd\..+?$/, /^r6i\..+?$/, /^r6id\..+?
$/,
    /^r6idn\..+?$/, /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..
+?$/,
    /^r7iz\..+?$/, /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^t3\..+?$/, /^t3a\..+?$/, /^t4g\..+?$/, /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?
$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2gd\..+?$/, /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/,
    /^z1d\..+?$/
]

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]

#
# Primary Rules
#
rule autoscaling_group_nitro_instance_override_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %autoscaling_groups not empty
 {
    check(%autoscaling_groups.Properties)
        <<
        [CT.AUTOSCALING.PR.10]: Require an Amazon EC2 Auto Scaling group to override
 only those launch templates with AWS Nitro instance types
        [FIX]: In the MixedInstancesPolicy.LaunchTemplate property, if it has one or
 more 'Overrides' fields that include 'InstanceType' or 'InstanceRequirements',
        set the value of 'InstanceType' to an Amazon EC2 instance type that is
 based on the AWS Nitro system, or set the value of 'AllowedInstanceTypes' in
 'InstanceRequirements' to one or more Amazon EC2 instance
        types that are based on the AWS Nitro system.
        >>
}
```

```
rule autoscaling_group_nitro_instance_override_check when is_cfn_hook(%INPUT_DOCUMENT,
 %AUTOSCALING_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.10]: Require an Amazon EC2 Auto Scaling group to override
 only those launch templates with AWS Nitro instance types
        [FIX]: In the MixedInstancesPolicy.LaunchTemplate property, if it has one or
 more 'Overrides' fields that include 'InstanceType' or 'InstanceRequirements',
        set the value of 'InstanceType' to an Amazon EC2 instance type that is
 based on the AWS Nitro system, or set the value of 'AllowedInstanceTypes' in
 'InstanceRequirements' to one or more Amazon EC2
        instance types that are based on the AWS Nitro system.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_group) {
    %autoscaling_group [
        # Scenarios 2 and 3
        filter_launch_template_overrides(this)
    ] {
        MixedInstancesPolicy {
            LaunchTemplate {
                Overrides[ InstanceType exists ] {
                    # Scenarios 4 and 7
                    InstanceType in %NITRO_INSTANCE_TYPES
                }
                Overrides[ InstanceRequirements exists ] {
                    InstanceRequirements {
                        # Scenarios 5, 6 and 8
                        AllowedInstanceTypes exists
                        AllowedInstanceTypes is_list
                        AllowedInstanceTypes not empty
                        AllowedInstanceTypes[*] in %NITRO_INSTANCE_TYPES
                    }
                }
            }
        }
    }
}

rule filter_launch_template_overrides(autoscaling_group) {
```

```
    %autoscaling_group {
        MixedInstancesPolicy exists
        MixedInstancesPolicy is_struct

        MixedInstancesPolicy {
            LaunchTemplate exists
            LaunchTemplate is_struct

            LaunchTemplate {
                Overrides exists
                Overrides is_list
                Overrides not empty

                some Overrides[*] {
                    InstanceType exists or
                    InstanceRequirements exists
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.AUTOSCALING.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceType: t3.micro
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
              - m5.*
```

```
              - c5.*
              VCpuCount:
                Min: 2
                Max: 4
              MemoryMiB:
                Min: 4000
                Max: 8000
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
```

```
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceType: t2.micro
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
```

```
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
              - c4.large
              VCpuCount:
                Max: 16
                Min: 1
              MemoryMiB:
                Min: 1000
                Max: 17000
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

[CT.AUTOSCALING.PR.11] Require only AWS Nitro instance types that support network traffic encryption between instances to be added to an Amazon EC2 Auto Scaling group, when overriding a launch template

This control checks whether an Amazon EC2 Auto Scaling group uses AWS Nitro instance types that support network traffic encryption between instances, when overriding a launch template. The Auto Scaling group creates this override in the `AWS::Autoscaling::AutoscalingGroup.MixedInstancesPolicy.LaunchTemplate` parameter.

- **Control objective:** Encrypt data in transit, Protect data integrity, Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AutoScaling::AutoScalingGroup`
- **CloudFormation guard rule:** CT.AUTOSCALING.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.AUTOSCALING.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.AUTOSCALING.PR.11 example templates

Explanation

The Nitro System is a collection of hardware and software components built by AWS to enable high performance, high availability, and high security. The Nitro System provides enhanced security because it continuously monitors, protects, and verifies the instance's hardware and firmware. Virtualization resources are offloaded to dedicated hardware and software, thereby minimizing the attack surface. The Nitro System security model is locked down to prohibit administrative access, greatly reducing the possibility of human error and tampering.

AWS provides secure and private connectivity between Amazon EC2 instances of all types. In addition, some instance types use the offload capabilities of the underlying Nitro System hardware to encrypt in-transit traffic between instances, automatically. This encryption uses Authenticated Encryption with Associated Data (AEAD) algorithms, and 256-bit encryption. It has no impact on network performance.

Usage considerations

- This control applies only to Amazon EC2 Auto Scaling groups that are configured with launch template overrides that specify an instance type or instance attributes. A `LaunchTemplate.LaunchTemplateSpecification` configuration specifies one or more `Overrides` that also include `InstanceType` or `InstanceRequirements`.
- This control does not check the instance type configured on a launch template. To ensure that launch templates use Nitro instances types that support encryption in-transit

between instances, use this control in conjunction with related controls that check launch templates for Nitro instance types that support encryption in-transit between instances.

- To support in-transit traffic encryption between instances, the Amazon EC2 instances must be one of the types required by this control, the instances must be in the same AWS Region, and they must be in the same VPC or group of peered VPCs, in which traffic does not pass through a virtual network device or service, such as a load balancer or a transit gateway.

Remediation for rule failure

In `MixedInstancesPolicy.LaunchTemplate` with one or more `Overrides` that include `InstanceType` or `InstanceRequirements`, set either `InstanceType` to an EC2 instance type based on the AWS Nitro system that supports encryption in-transit between instances, or set `AllowedInstanceTypes` in `InstanceRequirements` to one or more EC2 instance types based on the AWS Nitro system that supports encryption in-transit between instances.

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example One

An Amazon EC2 Auto Scaling group configured with a launch template override and an instance type that is based on the AWS Nitro system. It supports encryption in transit between instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                        }
```

```
                    },
                    "Overrides": [
                        {
                            "InstanceType": "c5a.large"
                        }
                    ]
                }
            },
            "MaxSize": 1,
            "MinSize": 0,
            "DesiredCapacity": 1,
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceType: c5a.large
    MaxSize: 1
    MinSize: 0
    DesiredCapacity: 1
    VPCZoneIdentifier:
      - !Ref 'Subnet'
```

The examples that follow show how to implement this remediation.

Amazon EC2 Auto Scaling Group - Example Two

An Amazon EC2 Auto Scaling group configured with a launch template override and its instance requirements, which specify a list of allowed instances that are based on the AWS Nitro system. It supports encryption in transit between instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "AutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "Properties": {
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": {
                            "Ref": "LaunchTemplate"
                        },
                        "Version": {
                            "Fn::GetAtt": "LaunchTemplate.LatestVersionNumber"
                        }
                    },
                    "Overrides": [
                        {
                            "InstanceRequirements": {
                                "AllowedInstanceTypes": [
                                    "c5a.*",
                                    "m6a.*"
                                ],
                                "VCpuCount": {
                                    "Min": 2,
                                    "Max": 4
                                },
                                "MemoryMiB": {
                                    "Min": 4000,
                                    "Max": 8000
                                }
                            }
                        }
                    ]
                }
            },
```

```
            "MaxSize": 1,
            "MinSize": 0,
            "DesiredCapacity": 1,
            "VPCZoneIdentifier": [
                {
                    "Ref": "Subnet"
                }
            ]
        }
    }
}
```

YAML example

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    MixedInstancesPolicy:
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref 'LaunchTemplate'
          Version: !GetAtt 'LaunchTemplate.LatestVersionNumber'
        Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
                - c5a.*
                - m6a.*
              VCpuCount:
                Min: 2
                Max: 4
              MemoryMiB:
                Min: 4000
                Max: 8000
    MaxSize: 1
    MinSize: 0
    DesiredCapacity: 1
    VPCZoneIdentifier:
      - !Ref 'Subnet'
```

CT.AUTOSCALING.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   autoscaling_group_nitro_encryption_in_transit_override_check
#
# Description:
#   This control checks whether an Auto Scaling group, when using a mixed instance
 policy, overrides only those launch templates with AWS Nitro
#   instance types that support encryption in transit between instances.
#
# Reports on:
#   AWS::AutoScaling::AutoScalingGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EC2 auto scaling group
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has not been provided
 or
#            has been provided as an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: No entries in 'Overrides' include 'InstanceType' or 'InstanceRequirements'
#      Then: SKIP
```

```
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to
 an instance type
#            other than a Nitro instance type that supports encryption in-transit
 between instances
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has not been
 provided or has been
#            provided as an empty list
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: An entry in 'AllowedInstanceTypes' is set to an instance type other than a
 Nitro instance type
#            that supports encryption in-transit between instances
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceType' has been provided and set to a
 Nitro instance type
#            that supports encryption in-transit between instances
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EC2 auto scaling group resource
```

```
#       And: 'Overrides' in 'MixedInstancesPolicy.LaunchTemplate' has been provided as
 a non-empty list
#       And: For an entry in 'Overrides', 'InstanceRequirements' has been provided
#       And: For the same entry in 'Overrides', 'AllowedInstanceTypes' has been
 provided as a non-empty list
#       And: Every entry in 'AllowedInstanceTypes' is set to a Nitro instance type that
#            supports encryption in-transit between instances
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let AUTOSCALING_GROUP_TYPE = "AWS::AutoScaling::AutoScalingGroup"
let NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES = [
    /^c5a\..+?$/, /^c5ad\..+?$/, /^c5n\..+?$/, /^c6a\..+?$/, /^c6gn\..+?$/, /^c6i\..+?
$/,
    /^c6id\..+?$/, /^c6in\..+?$/, /^c7a\..+?$/, /^c7g\..+?$/, /^c7gd\..+?$/, /^c7gn\..
+?$/,
    /^c7i-flex\..+?$/, /^c7i\..+?$/, /^c8g\..+?$/, /^c8gd\..+?$/, /^c8gn\..+?$/,
    /^d3\..+?$/, /^d3en\..+?$/, /^dl1\..+?$/, /^dl2q\..+?$/,
    /^f2\..+?$/,
    /^g4ad\..+?$/, /^g4dn\..+?$/, /^g5\..+?$/, /^g6\..+?$/, /^g6e\..+?$/, /^g6f\..+?$/,
    /^gr6\..+?$/, /^gr6f\..+?$/,
    /^hpc6a\..+?$/, /^hpc6id\..+?$/, /^hpc7a\..+?$/, /^hpc7g\..+?$/,
    /^i3en\..+?$/, /^i4g\..+?$/, /^i4i\..+?$/, /^i7i\..+?$/, /^i7ie\..+?$/, /^i8g\..+?
$/,
    /^im4gn\..+?$/, /^inf1\..+?$/, /^inf2\..+?$/, /^is4gen\..+?$/,
    /^m5dn\..+?$/, /^m5n\..+?$/, /^m5zn\..+?$/, /^m6a\..+?$/, /^m6i\..+?$/, /^m6id\..+?
$/,
    /^m6idn\..+?$/, /^m6in\..+?$/, /^m7a\..+?$/, /^m7g\..+?$/, /^m7gd\..+?$/,
    /^m7i-flex\..+?$/, /^m7i\..+?$/, /^m8g\..+?$/, /^m8gd\..+?$/,
    /^p3dn\..+?$/, /^p4d\..+?$/, /^p4de\..+?$/, /^p5\..+?$/, /^p5e\..+?$/, /^p5en\..+?
$/,
    /^p6-b200\..+?$/,
    /^r5dn\..+?$/, /^r5n\..+?$/, /^r6a\..+?$/, /^r6i\..+?$/, /^r6id\..+?$/, /^r6idn\..
+?$/,
    /^r6in\..+?$/, /^r7a\..+?$/, /^r7g\..+?$/, /^r7gd\..+?$/, /^r7i\..+?$/, /^r7iz\..+?
$/,
    /^r8g\..+?$/, /^r8gd\..+?$/, /^r8i-flex\..+?$/, /^r8i\..+?$/,
    /^trn1\..+?$/, /^trn1n\..+?$/, /^trn2\..+?$/,
    /^u-12tb1\..+?$/, /^u-18tb1\..+?$/, /^u-24tb1\..+?$/, /^u-3tb1\..+?$/, /^u-6tb1\..
+?$/,
    /^u-9tb1\..+?$/, /^u7i-12tb\..+?$/, /^u7i-6tb\..+?$/, /^u7i-8tb\..+?$/,
```

```
    /^u7in-16tb\..+?$/, /^u7in-24tb\..+?$/, /^u7in-32tb\..+?$/,
    /^vt1\..+?$/,
    /^x2idn\..+?$/, /^x2iedn\..+?$/, /^x2iezn\..+?$/, /^x8g\..+?$/
]

#
# Assignments
#
let autoscaling_groups = Resources.*[ Type == %AUTOSCALING_GROUP_TYPE ]

#
# Primary Rules
#
rule autoscaling_group_nitro_encryption_in_transit_override_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %autoscaling_groups not empty {
    check(%autoscaling_groups.Properties)
        <<
        [CT.AUTOSCALING.PR.11]: Require an Amazon EC2 instance to use a Nitro instance
 type that supports encryption in transit between instances when created using the
        'AWS::AutoScaling::AutoScalingGroup' resource type
        [FIX]: In 'MixedInstancesPolicy.LaunchTemplate' with one or more 'Overrides'
 that include 'InstanceType' or 'InstanceRequirements', set either 'InstanceType' to an
 Amazon EC2 instance type based on
        the AWS Nitro system that supports encryption in-transit between instances,
 or set 'AllowedInstanceTypes' in 'InstanceRequirements' to one or more Amazon EC2
 instance types based on the AWS Nitro system
        that supports encryption in-transit between instances.
        >>
}

rule autoscaling_group_nitro_encryption_in_transit_override_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AUTOSCALING_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%AUTOSCALING_GROUP_TYPE.resourceProperties)
        <<
        [CT.AUTOSCALING.PR.11]: Require an Amazon EC2 instance to use a Nitro instance
 type that supports encryption in transit between instances when created using the
        'AWS::AutoScaling::AutoScalingGroup' resource type
        [FIX]: In 'MixedInstancesPolicy.LaunchTemplate' with one or more 'Overrides'
 that include 'InstanceType' or 'InstanceRequirements', set either 'InstanceType' to an
 Amazon EC2 instance type based on the AWS Nitro
```

```
        system that supports encryption in-transit between instances, or set
 'AllowedInstanceTypes' in 'InstanceRequirements' to one or more Amazon EC2 instance
 types based on the AWS Nitro system
        that supports encryption in-transit between instances.
        >>
}

#
# Parameterized Rules
#
rule check(autoscaling_group) {
    %autoscaling_group [
        # Scenarios 2 and 3
        filter_launch_template_overrides(this)
    ] {
        MixedInstancesPolicy {
            LaunchTemplate {
                Overrides[ InstanceType exists ] {
                    # Scenarios 4 and 7
                    InstanceType in %NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES
                }
                Overrides[ InstanceRequirements exists ] {
                    InstanceRequirements {
                        # Scenarios 5, 6 and 8
                        AllowedInstanceTypes exists
                        AllowedInstanceTypes is_list
                        AllowedInstanceTypes not empty
                        AllowedInstanceTypes[*] in
 %NITRO_ENCRYPTION_IN_TRANSIT_INSTANCE_TYPES
                    }
                }
            }
        }
    }
}

rule filter_launch_template_overrides(autoscaling_group) {
    %autoscaling_group {
        MixedInstancesPolicy exists
        MixedInstancesPolicy is_struct

        MixedInstancesPolicy {
            LaunchTemplate exists
            LaunchTemplate is_struct
```

```
            LaunchTemplate {
                Overrides exists
                Overrides is_list
                Overrides not empty

                some Overrides[*] {
                    InstanceType exists or
                    InstanceRequirements exists
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.AUTOSCALING.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
```

```
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceType: c5a.large
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

PASS Example - Use this template to verify a compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
              - m6a.*
              - c5a.*
              VCpuCount:
                Min: 2
                Max: 4
              MemoryMiB:
                Min: 4000
```

```
                Max: 8000
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
```

```
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceType: t2.micro
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Parameters:
  LatestAmiId:
    Description: Region specific latest AMI ID from the Parameter Store
    Type: AWS::SSM::Parameter::Value 'AWS::EC2::Image::Id'
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId:
          Ref: LatestAmiId
  AutoScalingGroup:
```

```
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId:
              Ref: LaunchTemplate
            Version:
              Fn::GetAtt: LaunchTemplate.LatestVersionNumber
          Overrides:
          - InstanceRequirements:
              AllowedInstanceTypes:
              - c4.large
              VCpuCount:
                Max: 16
                Min: 1
              MemoryMiB:
                Min: 1000
                Max: 17000
      MaxSize: 1
      MinSize: 0
      DesiredCapacity: 1
      VPCZoneIdentifier:
      - Ref: Subnet
```

Amazon ElastiCache controls

Topics

- [CT.ELASTICACHE.PR.1] Require an Amazon ElastiCache (Redis OSS) cluster to have automatic backups activated
- [CT.ELASTICACHE.PR.2] Require an Amazon ElastiCache (Redis OSS) cluster to have automatic minor version upgrades activated
- [CT.ELASTICACHE.PR.3] Require an Amazon ElastiCache (Redis OSS) replication group to have automatic failover activated
- [CT.ELASTICACHE.PR.4] Require an Amazon ElastiCache (Redis OSS) replication group to have encryption at rest activated
- [CT.ELASTICACHE.PR.5] Require an Amazon ElastiCache (Redis OSS) replication group to have encryption in transit activated

- [CT.ELASTICACHE.PR.6] Require an Amazon ElastiCache cache cluster to use a custom subnet group
- [CT.ELASTICACHE.PR.7] Require an Amazon ElastiCache replication group of earlier Redis OSS versions to have Redis OSS AUTH activated
- [CT.ELASTICACHE.PR.8] Require an Amazon ElastiCache replication group of later Redis OSS versions to have RBAC authentication activated

[CT.ELASTICACHE.PR.1] Require an Amazon ElastiCache (Redis OSS) cluster to have automatic backups activated

This control checks whether an Amazon ElastiCache (Redis OSS) cluster has automatic backups enabled.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::CacheCluster`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.1 example templates

Explanation

When automatic backups are enabled, Amazon ElastiCache creates a backup of the cluster on a daily basis. There is no impact on the cluster, and the change is immediate. Automatic backups can help guard against data loss. In the event of a failure, you can create a new cluster, and restore your data from the most recent backup.

Usage considerations

- This control applies only to Amazon ElastiCache cache clusters with an engine type of `redis`

Remediation for rule failure

Set the value of the `SnapshotRetentionLimit` parameter to an integer value greater than 0.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Cache Cluster - Example

An Amazon ElastiCache cache cluster configured with automatic backups enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "CacheCluster": {
        "Type": "AWS::ElastiCache::CacheCluster",
        "Properties": {
            "Engine": "redis",
            "NumCacheNodes": 1,
            "CacheNodeType": "cache.t3.micro",
            "VpcSecurityGroupIds": [
                {
                    "Ref": "SecurityGroup"
                }
            ],
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "SnapshotRetentionLimit": 5
        }
    }
}
```

YAML example

```
CacheCluster:
  Type: AWS::ElastiCache::CacheCluster
  Properties:
    Engine: redis
    NumCacheNodes: 1
    CacheNodeType: cache.t3.micro
    VpcSecurityGroupIds:
      - !Ref 'SecurityGroup'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    SnapshotRetentionLimit: 5
```

CT.ELASTICACHE.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticache_redis_cluster_auto_backup_check
#
# Description:
#   This control checks whether an Amazon ElastiCache (Redis OSS) cluster has automatic
 backups enabled.
#
# Reports on:
#   AWS::ElastiCache::CacheCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache cache cluster
 resources
#      Then: SKIP
```

```
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache CacheCluster resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cache cluster resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'SnapshotRetentionLimit' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cache cluster resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'SnapshotRetentionLimit' has been provided and set to a non-integer value
 or an integer value of 0
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cache cluster resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'SnapshotRetentionLimit' has been provided and set to an integer value
 greater than 0
#      Then: PASS

#
# Constants
#
let ELASTICACHE_CACHE_CLUSTER_TYPE = "AWS::ElastiCache::CacheCluster"
let REDIS_ENGINE_TYPE = "redis"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_clusters = Resources.*[ Type == %ELASTICACHE_CACHE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule elasticache_redis_cluster_auto_backup_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %elasticache_clusters not empty {
```

```
    check(%elasticache_clusters.Properties)
        <<
        [CT.ELASTICACHE.PR.1]: Require an Amazon ElastiCache (Redis OSS) cluster to
 have automatic backups activated
        [FIX]: Set the value of the 'SnapshotRetentionLimit' parameter to an integer
 value greater than 0.
        >>
}

rule elasticache_redis_cluster_auto_backup_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_CACHE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_CACHE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.1]: Require an Amazon ElastiCache (Redis OSS) cluster to
 have automatic backups activated
        [FIX]: Set the value of the 'SnapshotRetentionLimit' parameter to an integer
 value greater than 0.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_cache_cluster) {
    %elasticache_cache_cluster [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE
    ] {
        # Scenarios 3, 4 and 5
        SnapshotRetentionLimit exists
        SnapshotRetentionLimit > 0
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICACHE.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: test
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
```

```
    Properties:
      Description: Cache Subnet Group
      SubnetIds:
      - Ref: Subnet
  CacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      Engine: redis
      NumCacheNodes: 1
      CacheNodeType: cache.t3.micro
      VpcSecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      SnapshotRetentionLimit: 5
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: test
      VpcId:
        Ref: VPC
```

```
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Cache Subnet Group
      SubnetIds:
      - Ref: Subnet
  CacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      Engine: redis
      NumCacheNodes: 1
      CacheNodeType: cache.t3.micro
      VpcSecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      SnapshotRetentionLimit: 0
```

[CT.ELASTICACHE.PR.2] Require an Amazon ElastiCache (Redis OSS) cluster to have automatic minor version upgrades activated

This control checks whether an Amazon ElastiCache (Redis OSS) cluster has automatic minor version upgrades enabled.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::CacheCluster`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.2 example templates

Explanation

By enabling automatic minor version upgrades, you ensure that the latest minor version updates to Amazon ElastiCache cache clusters are installed. These upgrades may include security patches and bug fixes. Keeping up to date with patch installation is an important step in securing systems.

Usage considerations

- This control applies only to Amazon ElastiCache cache clusters with an engine type of `redis` and an engine version of 6.0 or later.

Remediation for rule failure

Set the value of the `AutoMinorVersionUpgrade` parameter to true.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Cache Cluster - Example

An Amazon ElastiCache cache cluster configured with automatic minor version upgrades enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElastCacheCacheCluster": {
        "Type": "AWS::ElastiCache::CacheCluster",
        "Properties": {
            "CacheNodeType": "cache.t3.micro",
            "NumCacheNodes": "1",
            "VpcSecurityGroupIds": [
                {
                    "Fn::GetAtt": [
                        "SecurityGroup",
```

```
                        "GroupId"
                    ]
                }
            ],
            "Engine": "redis",
            "EngineVersion": 6.0,
            "AutoMinorVersionUpgrade": true
        }
    }
}
```

YAML example

```
ElastCacheCacheCluster:
  Type: AWS::ElastiCache::CacheCluster
  Properties:
    CacheNodeType: cache.t3.micro
    NumCacheNodes: '1'
    VpcSecurityGroupIds:
      - !GetAtt 'SecurityGroup.GroupId'
    Engine: redis
    EngineVersion: 6.0
    AutoMinorVersionUpgrade: true
```

CT.ELASTICACHE.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticache_auto_minor_version_upgrade_check
#
# Description:
#   This control checks whether an Amazon ElastiCache (Redis OSS) cluster has automatic
 minor version upgrades enabled.
#
# Reports on:
```

```
#   AWS::ElastiCache::CacheCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has been provided and set to a version less than 6
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has not been provided or 'EngineVersion' has been provided
 and set
#            to a version greater than or equal to 6
#       And: 'AutoMinorVersionUpgrade' has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has not been provided or 'EngineVersion' has been provided
 and set
#            to a version greater than or equal to 6
#       And: 'AutoMinorVersionUpgrade' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
```

```
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has not been provided or 'EngineVersion' has been provided
 and set
#            to a version greater than or equal to 6
#       And: 'AutoMinorVersionUpgrade' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICACHE_CLUSTER_TYPE = "AWS::ElastiCache::CacheCluster"
let INPUT_DOCUMENT = this
let REDIS_ENGINE_TYPE = "redis"
let UNSUPPORTED_REDIS_ENGINE_VERSIONS_FOR_AUTO_UPGRADE = [
    /^2\./,
    /^3\./,
    /^4\./,
    /^5\./
]

#
# Assignments
#
let elasticache_clusters = Resources.*[ Type == %ELASTICACHE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule elasticache_auto_minor_version_upgrade_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %elasticache_clusters not empty
 {
    check(%elasticache_clusters.Properties)
        <<
        [CT.ELASTICACHE.PR.2]: Require an Amazon ElastiCache (Redis OSS) cluster to
 have automatic minor version upgrades activated
        [FIX]: Set the value of the 'AutoMinorVersionUpgrade' parameter to true.
        >>
}

rule elasticache_auto_minor_version_upgrade_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_CLUSTER_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%ELASTICACHE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.2]: Require an Amazon ElastiCache (Redis OSS) cluster to
 have automatic minor version upgrades activated
        [FIX]: Set the value of the 'AutoMinorVersionUpgrade' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_clusters) {
    %elasticache_clusters [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE

        # Scenario 3
        EngineVersion not exists or
        EngineVersion not in %UNSUPPORTED_REDIS_ENGINE_VERSIONS_FOR_AUTO_UPGRADE
    ] {
        # Scenario 4, 5 and 6
        AutoMinorVersionUpgrade exists
        AutoMinorVersionUpgrade == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICACHE.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 11211
        ToPort: 11211
        CidrIp: 10.0.0.0/24
  ElastCacheCacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType: cache.t3.micro
      NumCacheNodes: '1'
      VpcSecurityGroupIds:
      - Fn::GetAtt:
        - SecurityGroup
        - GroupId
      Engine: redis
      AutoMinorVersionUpgrade: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      SecurityGroupIngress:
      - IpProtocol: tcp
```

```
        FromPort: 11211
        ToPort: 11211
        CidrIp: 10.0.0.0/24
  ElastCacheCacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType: cache.t3.micro
      NumCacheNodes: '1'
      VpcSecurityGroupIds:
      - Fn::GetAtt:
        - SecurityGroup
        - GroupId
      Engine: redis
      AutoMinorVersionUpgrade: false
```

[CT.ELASTICACHE.PR.3] Require an Amazon ElastiCache (Redis OSS) replication group to have automatic failover activated

This control checks whether an Amazon ElastiCache (Redis OSS) replication group has automatic failover enabled.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::ReplicationGroup`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.3 example templates

Explanation

When automatic failover is enabled for a replication group, the role of primary node will fail over to one of the read replicas, automatically. This failover and replica promotion ensure that you can resume writing to the new primary as soon as promotion is complete, thereby reducing overall downtime in case of failure.

Remediation for rule failure

Set the value of the `AutomaticFailoverEnabled` parameter to true.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Replication Group - Example

An Amazon ElastiCache replication group configured with automatic failover enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ReplicationGroup": {
        "Type": "AWS::ElastiCache::ReplicationGroup",
        "Properties": {
            "ReplicationGroupDescription": "Sample replication group",
            "CacheNodeType": "cache.t3.micro",
            "SecurityGroupIds": [
                {
                    "Ref": "SecurityGroup"
                }
            ],
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "NumCacheClusters": 2,
            "Engine": "redis",
            "AutomaticFailoverEnabled": true
        }
    }
}
```

YAML example

```
ReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    ReplicationGroupDescription: Sample replication group
    CacheNodeType: cache.t3.micro
    SecurityGroupIds:
      - !Ref 'SecurityGroup'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    NumCacheClusters: 2
    Engine: redis
    AutomaticFailoverEnabled: true
```

CT.ELASTICACHE.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticache_repl_grp_backup_enabled_check
#
# Description:
#   This control checks whether an Amazon ElastiCache (Redis OSS) replication group has
 automatic failover enabled.
#
# Reports on:
#   AWS::ElastiCache::ReplicationGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache replication group
 resources
```

```
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AutomaticFailoverEnabled' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AutomaticFailoverEnabled' has been provided and is set to a value other
 than bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AutomaticFailoverEnabled' has been provided and is set to a value of
 bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICACHE_REPLICATION_GROUP_TYPE = "AWS::ElastiCache::ReplicationGroup"
let REDIS_ENGINE_TYPE = "redis"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_replication_groups = Resources.*[ Type ==
 %ELASTICACHE_REPLICATION_GROUP_TYPE ]

#
# Primary Rules
#
```

```
rule elasticache_repl_grp_auto_failover_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elasticache_replication_groups not empty {
    check(%elasticache_replication_groups.Properties)
        <<
        [CT.ELASTICACHE.PR.3]: Require an Amazon ElastiCache (Redis OSS) replication
 group to have automatic failover activated
        [FIX]: Set the value of the 'AutomaticFailoverEnabled' parameter to true.
        >>
}

rule elasticache_repl_grp_auto_failover_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_REPLICATION_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_REPLICATION_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.3]: Require an Amazon ElastiCache (Redis OSS) replication
 group to have automatic failover activated
        [FIX]: Set the value of the 'AutomaticFailoverEnabled' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_replication_group) {
    %elasticache_replication_group [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE
    ] {
        # Scenarios 3, 4 and 5
        AutomaticFailoverEnabled exists
        AutomaticFailoverEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICACHE.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
```

```
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      AutomaticFailoverEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
```

```
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      AutomaticFailoverEnabled: false
```

[CT.ELASTICACHE.PR.4] Require an Amazon ElastiCache (Redis OSS) replication group to have encryption at rest activated

This control checks whether an Amazon ElastiCache (Redis OSS) replication group has the encryption-at-rest setting enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::ReplicationGroup`

- **CloudFormation guard rule:** CT.ELASTICACHE.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.4 example templates

Explanation

Encryption of data at rest is a recommended best practice that adds a layer of access management around your data. In case of any compromise to your Amazon ElastiCache replica nodes, this encryption-at-rest setting ensures that your data is protected from unintended access.

Usage considerations

- This control requires the use of encryption at rest, which is supported only for replication groups with Redis OSS engine versions of 3.2.6 or above.

Remediation for rule failure

Set the value of the `AtRestEncryptionEnabled` parameter to true.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Replication Group - Example

Amazon ElastiCache replication group configured with encryption at rest enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElastiCacheReplicationGroup": {
        "Type": "AWS::ElastiCache::ReplicationGroup",
        "Properties": {
            "CacheSubnetGroupName": {
```

```
                "Ref": "SubnetGroup"
            },
            "CacheNodeType": "cache.t3.medium",
            "NumCacheClusters": 2,
            "Engine": "redis",
            "ReplicationGroupDescription": "Sample replication group",
            "AtRestEncryptionEnabled": true
        }
    }
}
```

YAML example

```
ElastiCacheReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    CacheNodeType: cache.t3.medium
    NumCacheClusters: 2
    Engine: redis
    ReplicationGroupDescription: Sample replication group
    AtRestEncryptionEnabled: true
```

CT.ELASTICACHE.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticache_repl_grp_encrypted_at_rest_check
#
# Description:
#   This control checks whether an Amazon ElastiCache replication group has the
 encryption-at-rest setting enabled.
#
# Reports on:
#   AWS::ElastiCache::ReplicationGroup
```

```
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#  Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache ReplicationGroup
 resources
#      Then: SKIP
#  Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#  Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AtRestEncryptionEnabled' has not been provided
#      Then: FAIL
#  Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AtRestEncryptionEnabled' has been provided and is set to a value other
 than bool(true)
#      Then: FAIL
#  Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'AtRestEncryptionEnabled' has been provided and is set to a value of
 bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICACHE_REPLICATION_GROUP_TYPE = "AWS::ElastiCache::ReplicationGroup"
```

```
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_replication_groups = Resources.*[ Type ==
 %ELASTICACHE_REPLICATION_GROUP_TYPE ]

#
# Primary Rules
#
rule elasticache_repl_grp_encrypted_at_rest_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %elasticache_replication_groups
 not empty {
    check(%elasticache_replication_groups.Properties)
        <<
        [CT.ELASTICACHE.PR.4]: Require an Amazon ElastiCache replication group to have
 encryption at rest activated
        [FIX]: Set the value of the 'AtRestEncryptionEnabled' parameter to true.
        >>
}

rule elasticache_repl_grp_encrypted_at_rest_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_REPLICATION_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_REPLICATION_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.4]: Require an Amazon ElastiCache replication group to have
 encryption at rest activated
        [FIX]: Set the value of the 'AtRestEncryptionEnabled' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_replication_group) {
    %elasticache_replication_group [
        # Scenario 2
        filter_elasticache_replication_group(this)
    ] {
        # Scenario 3
        AtRestEncryptionEnabled exists

        # Scenarios 4 and 5
```

```
        AtRestEncryptionEnabled == true
    }
}

rule filter_elasticache_replication_group(elasticache_replication_group) {
    %elasticache_replication_group {
        Engine exists
        Engine == "redis"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICACHE.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
```

```
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ElastiCacheReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      CacheSubnetGroupName:
        Ref: SubnetGroup
      CacheNodeType: cache.t3.medium
      NumCacheClusters: 2
      Engine: redis
      ReplicationGroupDescription: Example replication group
      AtRestEncryptionEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
```

```
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ElastiCacheReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      CacheSubnetGroupName:
        Ref: SubnetGroup
      CacheNodeType: cache.t3.medium
      NumCacheClusters: 2
      Engine: redis
      ReplicationGroupDescription: Example replication group
      AtRestEncryptionEnabled: false
```

[CT.ELASTICACHE.PR.5] Require an Amazon ElastiCache (Redis OSS) replication group to have encryption in transit activated

This control checks whether an Amazon ElastiCache replication group has encryption-in-transit enabled.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::ReplicationGroup`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.5 example templates

Explanation

TLS can be used to help prevent potential attackers from eavesdropping on or manipulating network traffic using person-in-the-middle or similar attacks. Amazon ElastiCache in-transit encryption is an optional feature that you can use to help protect your data when it is moving from one location to another.

Usage considerations

- Encryption-in-transit is supported on Amazon ElastiCache (Redis OSS) replication groups running versions 3.2.6, 4.0.10 and later.
- Because of the processing required to encrypt and decrypt the data at the endpoints, implementing in-transit encryption can reduce performance. We recommend that you benchmark in-transit encryption, compared to no encryption, on your own data, to determine the impact of encryption-in-transit on performance for your implementation.

Remediation for rule failure

Set the value of the `TransitEncryptionEnabled` parameter to true.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Replication Group - Example

Amazon ElastiCache replication group configured with encryption-in-transit enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ReplicationGroup": {
        "Type": "AWS::ElastiCache::ReplicationGroup",
        "Properties": {
```

```
            "ReplicationGroupDescription": "Sample replication group",
            "CacheNodeType": "cache.t3.micro",
            "SecurityGroupIds": [
                {
                    "Ref": "SecurityGroup"
                }
            ],
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "NumCacheClusters": 2,
            "Engine": "redis",
            "TransitEncryptionEnabled": true
        }
    }
}
```

YAML example

```
ReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    ReplicationGroupDescription: Sample replication group
    CacheNodeType: cache.t3.micro
    SecurityGroupIds:
      - !Ref 'SecurityGroup'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    NumCacheClusters: 2
    Engine: redis
    TransitEncryptionEnabled: true
```

CT.ELASTICACHE.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
```

```
#   elasticache_repl_grp_encrypted_in_transit_check
#
# Description:
#   This control checks whether an Amazon ElastiCache replication group has encryption-
in-transit enabled.
#
# Reports on:
#   AWS::ElastiCache::ReplicationGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache ReplicationGroup
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'TransitEncryptionEnabled' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'TransitEncryptionEnabled' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache ReplicationGroup resource
#       And: 'Engine' has been provided and set to 'redis'
```

```
#       And: 'TransitEncryptionEnabled' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICACHE_REPLICATION_GROUP_TYPE = "AWS::ElastiCache::ReplicationGroup"
let REDIS_ENGINE_TYPE = "redis"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_replication_groups = Resources.*[ Type ==
 %ELASTICACHE_REPLICATION_GROUP_TYPE ]

#
# Primary Rules
#
rule elasticache_repl_grp_encrypted_in_transit_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %elasticache_replication_groups not empty {
    check(%elasticache_replication_groups.Properties)
        <<
        [CT.ELASTICACHE.PR.5]: Require an Amazon ElastiCache (Redis OSS) replication
 group to have encryption in transit activated
        [FIX]: Set the value of the 'TransitEncryptionEnabled' parameter to true.
        >>
}

rule elasticache_repl_grp_encrypted_in_transit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_REPLICATION_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_REPLICATION_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.5]: Require an Amazon ElastiCache for Redis replication
 group to have encryption in transit activated
        [FIX]: Set the value of the 'TransitEncryptionEnabled' parameter to true.
        >>
}

#
# Parameterized Rules
#
```

```
rule check(elasticache_replication_group) {
    %elasticache_replication_group [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE
    ] {
        # Scenarios 3, 4 and 5
        TransitEncryptionEnabled exists
        TransitEncryptionEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICACHE.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      TransitEncryptionEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
```

```
      NumCacheClusters: 2
      Engine: redis
      TransitEncryptionEnabled: false
```

[CT.ELASTICACHE.PR.6] Require an Amazon ElastiCache cache cluster to use a custom subnet group

This control checks whether an Amazon ElastiCache cache cluster is configured with a custom subnet group.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::CacheCluster`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.6 example templates

Explanation

When you launch an ElastiCache cluster, AWS creates a default subnet group if none exists already. The default group utilizes subnets from the default VPC. Using custom subnet groups allows you to be more restrictive about network access to ElastiCache clusters.

Usage considerations

- This rule evaluates whether an Amazon ElastiCache cache cluster has been configured with a custom subnet group.
- Custom subnet groups may contain subnets that reside in the default Amazon VPC.

Remediation for rule failure

Set the `CacheSubnetGroupName` parameter to the name of a custom Amazon ElastiCache cache subnet group.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Cache Cluster - Example

An Amazon ElastiCache cache cluster configured with a custom subnet group. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticacheCluster": {
        "Type": "AWS::ElastiCache::CacheCluster",
        "Properties": {
            "Engine": "memcached",
            "CacheNodeType": "cache.t3.micro",
            "NumCacheNodes": "1",
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "VpcSecurityGroupIds": [
                {
                    "Fn::GetAtt": [
                        "SecurityGroup",
                        "GroupId"
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
ElasticacheCluster:
  Type: AWS::ElastiCache::CacheCluster
```

```
  Properties:
    Engine: memcached
    CacheNodeType: cache.t3.micro
    NumCacheNodes: '1'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    VpcSecurityGroupIds:
      - !GetAtt 'SecurityGroup.GroupId'
```

CT.ELASTICACHE.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   elasticache_subnet_group_check
#
# Description:
#   This control checks whether an Amazon ElastiCache cache cluster is configured with
 a custom subnet group.
#
# Reports on:
#   AWS::ElastiCache::CacheCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'CacheSubnetGroupName' has not been provided
#      Then: FAIL
```

```
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'CacheSubnetGroupName' has been provided as an empty string or as a non-
valid local reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'CacheSubnetGroupName' has been provided and set to a value of 'default'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache cluster resource
#       And: 'CacheSubnetGroupName' has been provided as a non-empty string or a valid
 local reference
#      Then: PASS

#
# Constants
#
let ELASTICACHE_CACHE_CLUSTER_TYPE = "AWS::ElastiCache::CacheCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_cache_clusters = Resources.*[ Type == %ELASTICACHE_CACHE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule elasticache_subnet_group_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %elasticache_cache_clusters not empty {
    check(%elasticache_cache_clusters.Properties)
        <<
        [CT.ELASTICACHE.PR.6]: Require an Amazon ElastiCache cache cluster to use a
 custom subnet group
        [FIX]: Set the 'CacheSubnetGroupName' parameter to the name of a custom Amazon
 ElastiCache cache subnet group.
        >>
}
```

```
rule elasticache_subnet_group_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_CACHE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_CACHE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.6]: Require an Amazon ElastiCache cache cluster to use a
 custom subnet group
        [FIX]: Set the 'CacheSubnetGroupName' parameter to the name of a custom Amazon
 ElastiCache cache subnet group.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_cache_cluster) {
    %elasticache_cache_cluster {
        # Scenario 2
        CacheSubnetGroupName exists

        # Scenarios 3, 4 and 5
        check_subnet_group_is_not_default(this) or
        check_local_references(%INPUT_DOCUMENT, CacheSubnetGroupName,
 "AWS::ElastiCache::SubnetGroup")
    }
}

rule check_subnet_group_is_not_default(elasticache_cache_cluster) {
        %elasticache_cache_cluster {
            check_is_string_and_not_empty(CacheSubnetGroupName)
            CacheSubnetGroupName != "default"
        }
    }

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICACHE.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
```

```
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Example subnet group
      SubnetIds:
      - Ref: Subnet
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 11211
        ToPort: 11211
        CidrIp: 10.0.0.0/24
  CacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      Engine: memcached
      CacheNodeType: cache.t3.micro
      NumCacheNodes: '1'
      CacheSubnetGroupName:
        Ref: SubnetGroup
      VpcSecurityGroupIds:
      - Fn::GetAtt: [SecurityGroup, GroupId]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 11211
        ToPort: 11211
        CidrIp: 10.0.0.0/24
  CacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      Engine: memcached
      CacheNodeType: cache.t3.micro
      NumCacheNodes: '1'
      VpcSecurityGroupIds:
      - Fn::GetAtt: [SecurityGroup, GroupId]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 11211
        ToPort: 11211
        CidrIp: 10.0.0.0/24
  CacheCluster:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      Engine: memcached
```

```
      CacheNodeType: cache.t3.micro
      NumCacheNodes: '1'
      CacheSubnetGroupName: default
      VpcSecurityGroupIds:
      - Fn::GetAtt: [SecurityGroup, GroupId]
```

[CT.ELASTICACHE.PR.7] Require an Amazon ElastiCache replication group of earlier Redis OSS versions to have Redis OSS AUTH activated

This control checks whether an Amazon ElastiCache replication group with an engine version earlier than 6.0 has Redis OSS AUTH enabled.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::ReplicationGroup`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.ELASTICACHE.PR.7 example templates

Explanation

Redis authentication tokens, or passwords, enable Redis to require a password before allowing clients to run commands, thereby improving data security.

Usage considerations

- This control applies only to Amazon ElastiCache replication groups of Redis OSS engine versions earlier than six (6).

- This control requires encryption-in-transit to be enabled on replication groups by means of the `TransitEncryptionEnabled` property.

Remediation for rule failure

Set the value of the `AuthToken` parameter to a string between 16 characters and 128 characters in length, which contains only printable ASCII characters and does not contain non-alphanumeric characters outside of the set (!, &,

The examples that follow show how to implement this remediation.

Amazon ElastiCache Replication Group - Example

An Amazon ElastiCache replication group configured with Redis AUTH authentication enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ReplicationGroup": {
        "Type": "AWS::ElastiCache::ReplicationGroup",
        "Properties": {
            "ReplicationGroupDescription": "Sample replication group",
            "CacheNodeType": "cache.t3.micro",
            "SecurityGroupIds": [
                {
                    "Ref": "SecurityGroup"
                }
            ],
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "NumCacheClusters": 2,
            "Engine": "redis",
            "EngineVersion": "5.0.6",
            "TransitEncryptionEnabled": true,
            "AuthToken": {
                "Fn::Sub": "{{resolve:secretsmanager:
${ReplicationGroupSecret}::password}}"
            }
        }
```

```
    }
}
```

YAML example

```
ReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    ReplicationGroupDescription: Sample replication group
    CacheNodeType: cache.t3.micro
    SecurityGroupIds:
      - !Ref 'SecurityGroup'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    NumCacheClusters: 2
    Engine: redis
    EngineVersion: 5.0.6
    TransitEncryptionEnabled: true
    AuthToken: !Sub '{{resolve:secretsmanager:${ReplicationGroupSecret}::password}}'
```

CT.ELASTICACHE.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticache_repl_grp_redis_auth_enabled_check
#
# Description:
#   This control checks whether an Amazon ElastiCache replication group with an engine
 version earlier than 6.0 has Redis OSS AUTH enabled.
#
# Reports on:
#   AWS::ElastiCache::ReplicationGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
```

```
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElastiCache replication group
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has not been provided or has been provided and set to a
 version greater than or equal to 6
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has been provided and set to a version less than 6
#       And: 'AuthToken' has not been provided or has been provided and set to an empty
 string
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElastiCache replication group resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has been provided and set to a version less than 6
#       And: 'AuthToken' has been provided and set to a non-empty string
#      Then: PASS

#
# Constants
#
let ELASTICACHE_REPLICATION_GROUP_TYPE = "AWS::ElastiCache::ReplicationGroup"
let REDIS_ENGINE_TYPE = "redis"
let SUPPORTED_REDIS_ENGINE_VERSIONS_FOR_REDIS_AUTH = [
```

```
    /^2\./,
    /^3\./,
    /^4\./,
    /^5\./
]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_replication_groups = Resources.*[ Type ==
 %ELASTICACHE_REPLICATION_GROUP_TYPE ]

#
# Primary Rules
#
rule elasticache_repl_grp_redis_auth_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                    %elasticache_replication_groups
 not empty {
    check(%elasticache_replication_groups.Properties)
        <<
        [CT.ELASTICACHE.PR.7]: Require an Amazon ElastiCache replication group of
 earlier Redis OSS versions to have Redis OSS AUTH activated
        [FIX]: Set the value of the 'AuthToken' parameter to a string between
 16 characters and 128 characters in length, which contains only printable ASCII
 characters and does not contain non-alphanumeric characters outside of the set (!, &,
        >>
}

rule elasticache_repl_grp_redis_auth_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_REPLICATION_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_REPLICATION_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.7]: Require an Amazon ElastiCache replication group of
 earlier Redis OSS versions to have Redis OSS AUTH activated
        [FIX]: Set the value of the 'AuthToken' parameter to a string between
 16 characters and 128 characters in length, which contains only printable ASCII
 characters and does not contain non-alphanumeric characters outside of the set (!, &,
        >>
}

#
# Parameterized Rules
```

```
#
rule check(elasticache_replication_group) {
    %elasticache_replication_group [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE

        # Scenario 3
        EngineVersion exists
        EngineVersion in %SUPPORTED_REDIS_ENGINE_VERSIONS_FOR_REDIS_AUTH
    ] {
        # Scenarios 4 and 5
        AuthToken exists
        check_is_string_and_not_empty(AuthToken)
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.ELASTICACHE.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Cache subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroupSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Replication group secret
```

```
      GenerateSecretString:
        SecretStringTemplate: "{}"
        GenerateStringKey: password
        PasswordLength: 64
        ExcludePunctuation: true
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      EngineVersion: 5.0.6
      TransitEncryptionEnabled: true
      AuthToken:
        Fn::Sub: '{{resolve:secretsmanager:${ReplicationGroupSecret}::password}}'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
```

```
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example security group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Cache subnet group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      EngineVersion: 3.2.6
      TransitEncryptionEnabled: true
```

[CT.ELASTICACHE.PR.8] Require an Amazon ElastiCache replication group of later Redis OSS versions to have RBAC authentication activated

This control checks whether Amazon ElastiCache replication groups with an engine version greater than or equal to 6.0 have RBAC authentication enabled.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::ElastiCache::ReplicationGroup`
- **CloudFormation guard rule:** CT.ELASTICACHE.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICACHE.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICACHE.PR.8 example templates

Explanation

Role-Based Access Control (RBAC) helps you create users and assign specific permissions to them, by using an access string. You assign the users to user groups aligned with a specific role, such as administrators, or human resources. The roles are deployed to Amazon ElastiCache (Redis OSS) (Redis OSS) replication groups. This technique establishes security boundaries between clients using the same Redis OSS replication groups, and it prevents clients from having access to other clients' data. If you use RBAC authentication over Redis OSS AUTH, it reduces the number of credentials required for authenticated access to an Amazon ElastiCache replication group.

Usage considerations

- This control applies only to Amazon ElastiCache replication groups of Redis OSS engine versions greater than or equal to 6.0
- This control requires encryption in transit to be enabled on replication groups by means of the **TransitEncryptionEnabled** property

Remediation for rule failure

Set the value of the UserGroupIds property to a list that contains at least one Amazon ElastiCache user group identifier.

The examples that follow show how to implement this remediation.

Amazon ElastiCache Replication Group - Example

An Amazon ElastiCache replication group configured with RBAC authentication enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ReplicationGroup": {
        "Type": "AWS::ElastiCache::ReplicationGroup",
        "Properties": {
            "ReplicationGroupDescription": "Sample replication group",
            "CacheNodeType": "cache.t3.micro",
            "SecurityGroupIds": [
                {
                    "Ref": "SecurityGroup"
                }
            ],
            "CacheSubnetGroupName": {
                "Ref": "SubnetGroup"
            },
            "NumCacheClusters": 2,
            "Engine": "redis",
            "EngineVersion": 6.2,
            "TransitEncryptionEnabled": true,
            "UserGroupIds": [
                {
                    "Ref": "UserGroup"
                }
            ]
        }
    }
}
```

YAML example

```
ReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    ReplicationGroupDescription: Sample replication group
```

```
    CacheNodeType: cache.t3.micro
    SecurityGroupIds:
      - !Ref 'SecurityGroup'
    CacheSubnetGroupName: !Ref 'SubnetGroup'
    NumCacheClusters: 2
    Engine: redis
    EngineVersion: 6.2
    TransitEncryptionEnabled: true
    UserGroupIds:
      - !Ref 'UserGroup'
```

CT.ELASTICACHE.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   elasticache_repl_grp_rbac_auth_enabled_check
#
# Description:
#   This control checks whether Amazon ElastiCache replication groups with an engine
 version greater than or equal to 6.0 have RBAC authentication enabled.
#
# Reports on:
#   AWS::ElastiCache::ReplicationGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon ElastiCache replication
 group resources
#      Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ElastiCache replication group
 resource
#       And: 'Engine' has not been provided or has been provided and is not set to
 'redis'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ElastiCache replication group
 resource
#       And: 'Engine' has been provided and is set to 'redis'
#       And: 'EngineVersion' has been provided and set to a version less than 6
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ElastiCache replication group
 resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'EngineVersion' has not been provided or has been provided and set to a
 version greater than or equal to 6
#       And: 'UserGroupIds' has not been provided or has been provided as an empty list
 or a list containing an empty
#            string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ElastiCache replication group
 resource
#       And: 'Engine' has been provided and set to 'redis'
#       And: 'EngineVersion' has not been provided or has been provided and set to a
 version greater than or equal to 6
#       And: 'UserGroupIds' has been provided as a list containing non-empty strings or
 valid local references
#      Then: PASS

#
# Constants
#
let ELASTICACHE_REPLICATION_GROUP_TYPE = "AWS::ElastiCache::ReplicationGroup"
let REDIS_ENGINE_TYPE = "redis"
let UNSUPPORTED_REDIS_ENGINE_VERSIONS_FOR_RBAC = [
    /^2\./,
    /^3\./,
    /^4\./,
```

```
    /^5\./
]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticache_replication_groups = Resources.*[ Type ==
 %ELASTICACHE_REPLICATION_GROUP_TYPE ]

#
# Primary Rules
#
rule elasticache_repl_grp_rbac_auth_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %elasticache_replication_groups
 not empty {
    check(%elasticache_replication_groups.Properties)
        <<
        [CT.ELASTICACHE.PR.8]: Require an Amazon ElastiCache replication group of later
 Redis OSS versions to have RBAC authentication activated
        [FIX]: Set the value of the UserGroupIds property to a list that contains at
 least one Amazon ElastiCache user group identifier.
        >>
}

rule elasticache_repl_grp_rbac_auth_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICACHE_REPLICATION_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICACHE_REPLICATION_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICACHE.PR.8]: Require an Amazon ElastiCache replication group of later
 Redis OSS versions to have RBAC authentication activated
        [FIX]: Set the value of the UserGroupIds property to a list that contains at
 least one Amazon ElastiCache user group identifier.
        >>
}

#
# Parameterized Rules
#
rule check(elasticache_replication_group) {
    %elasticache_replication_group [
        # Scenario 2
        Engine exists
        Engine == %REDIS_ENGINE_TYPE
```

```
        # Scenario 3
        EngineVersion not exists or
        EngineVersion not in %UNSUPPORTED_REDIS_ENGINE_VERSIONS_FOR_RBAC
    ] {
        # Scenarios 4, 5 and 6
        UserGroupIds exists
        UserGroupIds is_list
        UserGroupIds not empty

        UserGroupIds[*] {
            check_is_string_and_not_empty(this) or
            check_local_references(%INPUT_DOCUMENT, this,
 "AWS::ElastiCache::UserGroup")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
```

```
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICACHE.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
```

```
      GroupDescription: test
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Cache Subnet Group
      SubnetIds:
      - Ref: Subnet
  UserGroup:
    Type: AWS::ElastiCache::UserGroup
    Properties:
      Engine: redis
      UserGroupId:
        Fn::Sub: ${AWS::StackName}-example-group
      UserIds:
      - default
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
      UserGroupIds:
      - Ref: UserGroup
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: test
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - FromPort: 443
        IpProtocol: tcp
        ToPort: 443
        CidrIp: 0.0.0.0/0
  SubnetGroup:
    Type: AWS::ElastiCache::SubnetGroup
    Properties:
      Description: Cache Subnet Group
      SubnetIds:
      - Ref: Subnet
  ReplicationGroup:
    Type: AWS::ElastiCache::ReplicationGroup
    Properties:
      ReplicationGroupDescription:
        Fn::Sub: ${AWS::StackName}-example
      CacheNodeType: cache.t3.micro
      SecurityGroupIds:
      - Ref: SecurityGroup
      CacheSubnetGroupName:
        Ref: SubnetGroup
      NumCacheClusters: 2
      Engine: redis
```

Amazon Elastic Container Registry controls

Topics

- [CT.ECR.PR.1] Require Amazon ECR repositories to have a lifecycle policy configured
- [CT.ECR.PR.2] Require Amazon ECR private repositories to have image scanning enabled
- [CT.ECR.PR.3] Require Amazon ECR private repositories to have tag immutability enabled

[CT.ECR.PR.1] Require Amazon ECR repositories to have a lifecycle policy configured

This control checks whether a private Amazon Elastic Container Registry (Amazon ECR) repository has at least one lifecycle policy configured.

- **Control objective:** Manage vulnerabilities, Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECR::Repository`
- **CloudFormation guard rule:** CT.ECR.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECR.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECR.PR.1 example templates

Explanation

Amazon ECR lifecycle policies specify the lifecycle management of images in a repository. By configuring lifecycle policies, you can automate the cleanup of unused images and the expiration of images, based on age or count. Automating these tasks can help you to avoid unintentionally using outdated images in your repository.

Remediation for rule failure

Provide a `LifecyclePolicy` configuration and set `LifecyclePolicyText` to an Amazon ECR repository lifecycle policy.

The examples that follow show how to implement this remediation.

Amazon ECR Repository - Example

Amazon ECR repository configured with a lifecycle policy. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECRRepository": {
        "Type": "AWS::ECR::Repository",
        "Properties": {
            "LifecyclePolicy": {
                "LifecyclePolicyText": "{\n  \"rules\": [\n    {\n      \"rulePriority
\": 1,\n      \"description\": \"Expire images older than 14 days\",\n      \"selection
\": {\n        \"tagStatus\": \"untagged\",\n        \"countType\": \"sinceImagePushed
\",\n        \"countUnit\": \"days\",\n        \"countNumber\": 14\n      },\n
 \"action\": {\n        \"type\": \"expire\"\n      }\n    }\n  ]\n}\n"
            }
        }
    }
}
```

YAML example

```
ECRRepository:
  Type: AWS::ECR::Repository
  Properties:
    LifecyclePolicy:
      LifecyclePolicyText: |
        {
          "rules": [
            {
              "rulePriority": 1,
```

```
            "description": "Expire images older than 14 days",
            "selection": {
              "tagStatus": "untagged",
              "countType": "sinceImagePushed",
              "countUnit": "days",
              "countNumber": 14
            },
            "action": {
              "type": "expire"
            }
          }
        ]
      }
```

CT.ECR.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#    ecr_private_lifecycle_policy_configured_check
#
# Description:
#   This control checks whether a private Amazon Elastic Container Registry (ECR)
 repository has at least one lifecycle policy configured.
#
# Reports on:
#    AWS::ECR::Repository
#
# Evaluates:
#    CloudFormation, CloudFormation hook
#
# Rule Parameters:
#    None
#
# Scenarios:
#  Scenario: 1
#    Given: The input document is an CloudFormation or CloudFormation hook document
#      And: The input document does not contain any ECR repository resources
```

```
#     Then: SKIP
#  Scenario: 2
#    Given: The input document is an CloudFormation or CloudFormation hook document
#      And: The input document contains an ECR repository resource
#      And: 'LifecyclePolicy' is not present
#     Then: FAIL
#  Scenario: 3
#    Given: The input document is an CloudFormation or CloudFormation hook document
#      And: The input document contains an ECR repository resource
#      And: 'LifecyclePolicy' is present
#      And: 'LifecyclePolicyText' has not been provided in the 'LifecyclePolicy'
 configuration or has been provided as
#            an empty string
#     Then: FAIL
#  Scenario: 4
#    Given: The input document is an CloudFormation or CloudFormation hook document
#      And: The input document contains an ECR repository resource
#      And: 'LifecyclePolicy' is present
#      And: 'LifecyclePolicyText' has been provided in the 'LifecyclePolicy'
 configuration with a non-empty string
#     Then: PASS

#
# Constants
#
let ECR_REPOSITORY_TYPE = "AWS::ECR::Repository"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecr_repositories = Resources.*[ Type == %ECR_REPOSITORY_TYPE ]

#
# Primary Rules
#
rule ecr_private_lifecycle_policy_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                    %ecr_repositories not empty {
    check(%ecr_repositories.Properties)
        <<
        [CT.ECR.PR.1]: Require Amazon ECR repositories to have a lifecycle policy
 configured
```

```
            [FIX]: Provide a 'LifecyclePolicy' configuration and set
 'LifecyclePolicyText' to an Amazon ECR repository lifecycle policy.
        >>
}

rule ecr_private_lifecycle_policy_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECR_REPOSITORY_TYPE) {
    check(%INPUT_DOCUMENT.%ECR_REPOSITORY_TYPE.resourceProperties)
        <<
        [CT.ECR.PR.1]: Require Amazon ECR repositories to have a lifecycle policy
 configured
            [FIX]: Provide a 'LifecyclePolicy' configuration and set
 'LifecyclePolicyText' to an Amazon ECR repository lifecycle policy.
        >>
}

#
# Parameterized Rules
#
rule check(ecr_repository) {
    %ecr_repository {
        #Scenario 3
        LifecyclePolicy exists
        LifecyclePolicy is_struct
        LifecyclePolicy {
            #Scenario 4
            LifecyclePolicyText exists
            check_is_string_and_not_empty(LifecyclePolicyText)
        }
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
     %value {
         this is_string
         this != /\A\s*\z/
     }
}

rule is_cfn_template(doc) {
    %doc {
```

```
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECR.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      LifecyclePolicy:
        LifecyclePolicyText: |
          {
            "rules": [
              {
                "rulePriority": 1,
                "description": "Expire images older than 14 days",
                "selection": {
                  "tagStatus": "untagged",
                  "countType": "sinceImagePushed",
                  "countUnit": "days",
                  "countNumber": 14
                },
                "action": {
                  "type": "expire"
                }
              }
            ]
          }
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties: {}
```

[CT.ECR.PR.2] Require Amazon ECR private repositories to have image scanning enabled

This control checks whether a private Amazon Elastic Container Registry (Amazon ECR) repository has image scanning enabled.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECR::Repository`
- **CloudFormation guard rule:** CT.ECR.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECR.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECR.PR.2 example templates

Explanation

Amazon ECR image scanning helps with identifying software vulnerabilities in your container images. Amazon ECR uses the Common Vulnerabilities and Exposures (CVEs) database from the open-source Clair project, and it provides a list of scan findings. Enabling image scanning on

Amazon ECR repositories adds a layer of verification regarding the integrity and safety of the images being stored.

Remediation for rule failure

Set `ScanOnPush` in `ImageScanningConfiguration` to `true`.

The examples that follow show how to implement this remediation.

Amazon ECR Repository - Example

Amazon ECR repository with image scanning enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECRRepository": {
        "Type": "AWS::ECR::Repository",
        "Properties": {
            "ImageScanningConfiguration": {
                "ScanOnPush": true
            }
        }
    }
}
```

YAML example

```
ECRRepository:
  Type: AWS::ECR::Repository
  Properties:
    ImageScanningConfiguration:
      ScanOnPush: true
```

CT.ECR.PR.2 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   ecr_private_image_scanning_enabled_check
#
# Description:
#   This control checks whether a private Amazon Elastic Container Registry (Amazon
 ECR) repository has image scanning enabled.
#
# Reports on:
#   AWS::ECR::Repository
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon ECR repository resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECR repository resource
#       And: 'ImageScanningConfiguration.ScanOnPush' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECR repository resource
#       And: 'ImageScanningConfiguration.ScanOnPush' has been provided and set to a
 value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECR repository resource
#       And: 'ImageScanningConfiguration.ScanOnPush' has been provided and set to
 bool(true)
#      Then: PASS

#
# Constants
```

```
#
let ECR_REPOSITORY_TYPE = "AWS::ECR::Repository"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecr_repositories = Resources.*[ Type == %ECR_REPOSITORY_TYPE ]

#
# Primary Rules
#
rule ecr_private_image_scanning_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %ecr_repositories not empty {
    check(%ecr_repositories.Properties)
        <<
        [CT.ECR.PR.2]: Require Amazon ECR private repositories to have image scanning
 enabled
        [FIX]: Set 'ScanOnPush' in 'ImageScanningConfiguration' to 'true'.
        >>
}

rule ecr_private_image_scanning_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECR_REPOSITORY_TYPE) {
    check(%INPUT_DOCUMENT.%ECR_REPOSITORY_TYPE.resourceProperties)
        <<
        [CT.ECR.PR.2]: Require Amazon ECR private repositories to have image scanning
 enabled
        [FIX]: Set 'ScanOnPush' in 'ImageScanningConfiguration' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(ecr_repository) {
    %ecr_repository {
        # Scenario 2
        ImageScanningConfiguration exists
        ImageScanningConfiguration is_struct

        ImageScanningConfiguration {
            # Scenario 3 and 4
            ScanOnPush exists
```

```
            ScanOnPush == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECR.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      ImageScanningConfiguration:
        ScanOnPush: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
```

```
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      ImageScanningConfiguration:
        ScanOnPush: false
```

[CT.ECR.PR.3] Require Amazon ECR private repositories to have tag immutability enabled

This control checks whether a private Amazon Elastic Container Registry (Amazon ECR) repository has tag immutability enabled.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECR::Repository`
- **CloudFormation guard rule:** CT.ECR.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECR.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECR.PR.3 example templates

Explanation

Amazon ECR tag immutability enables customers to rely on the descriptive tags of an image as a reliable mechanism that tracks and uniquely identifies images. An immutable tag is static, which means that each tag refers to a unique image. This tagging improves reliability and scalability, because the use of a static tag always results in the same image being deployed. When configured, tag immutability prevents the tags from being overridden, which reduces the attack surface.

Remediation for rule failure

Set `ImageTagMutability` to `IMMUTABLE`.

The examples that follow show how to implement this remediation.

Amazon ECR Repository - Example

Amazon ECR repository configured with immutable tags. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECRRepository": {
        "Type": "AWS::ECR::Repository",
        "Properties": {
            "ImageTagMutability": "IMMUTABLE"
        }
    }
}
```

YAML example

```
ECRRepository:
  Type: AWS::ECR::Repository
  Properties:
    ImageTagMutability: IMMUTABLE
```

CT.ECR.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ecr_private_tag_immutability_enabled_check
#
# Description:
#   This control checks whether a private Amazon Elastic Container Registry (Amazon
 ECR) repository has tag immutability enabled.
```

```
#
# Reports on:
#   AWS::ECR::Repository
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ECR repository resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECR repository resource
#       And: 'ImageTagMutability' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECR repository resource
#       And: 'ImageTagMutability' has been provided with a value of 'MUTABLE'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECR repository resource
#       And: 'ImageTagMutability' has been provided with a value of 'IMMUTABLE'
#      Then: PASS

#
# Constants
#
let ECR_REPOSITORY_TYPE = "AWS::ECR::Repository"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecr_repositories = Resources.*[ Type == %ECR_REPOSITORY_TYPE ]

#
# Primary Rules
```

```
#
rule ecr_private_tag_immutability_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %ecr_repositories not empty {
    check(%ecr_repositories.Properties)
        <<
        [CT.ECR.PR.3]: Require Amazon ECR private repositories to have tag immutability
 enabled
        [FIX]: Set 'ImageTagMutability' to 'IMMUTABLE'.
        >>
}

rule ecr_private_tag_immutability_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECR_REPOSITORY_TYPE) {
    check(%INPUT_DOCUMENT.%ECR_REPOSITORY_TYPE.resourceProperties)
        <<
        [CT.ECR.PR.3]: Require Amazon ECR private repositories to have tag immutability
 enabled
        [FIX]: Set 'ImageTagMutability' to 'IMMUTABLE'.
        >>
}

#
# Parameterized Rules
#
rule check(ecr_repository) {
    %ecr_repository {
        # Scenario 2, 3 and 4
        ImageTagMutability exists
        ImageTagMutability == "IMMUTABLE"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.ECR.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      ImageTagMutability: IMMUTABLE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      ImageTagMutability: MUTABLE
```

Amazon Elastic Container Service controls

Topics

- [CT.ECS.PR.1] Require Amazon ECS Fargate Services to run on the latest Fargate platform version
- [CT.ECS.PR.2] Require any Amazon ECS cluster to have container insights activated
- [CT.ECS.PR.3] Require any Amazon ECS task definition to specify a user that is not the root
- [CT.ECS.PR.4] Require Amazon ECS tasks to use 'awsvpc' networking mode
- [CT.ECS.PR.5] Require an active Amazon ECS task definition to have a logging configuration

- [CT.ECS.PR.6] Require Amazon ECS containers to allow read-only access to the root filesystem
- [CT.ECS.PR.7] Require an Amazon ECS task definition to have a specific memory usage limit
- [CT.ECS.PR.8] Require Amazon ECS task definitions to have secure networking modes and user definitions
- [CT.ECS.PR.9] Require Amazon ECS services not to assign public IP addresses automatically
- [CT.ECS.PR.10] Require that Amazon ECS task definitions do not share the host's process namespace
- [CT.ECS.PR.11] Require an Amazon ECS container to run as non-privileged
- [CT.ECS.PR.12] Require that Amazon ECS task definitions do not pass secrets as container environment variables

[CT.ECS.PR.1] Require Amazon ECS Fargate Services to run on the latest Fargate platform version

This control checks whether Amazon Elastic Container Service (Amazon ECS) Fargate services are configured to deploy using the `LATEST` platform version rather than a specified version number.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::Service`
- **CloudFormation guard rule:** CT.ECS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.1 example templates

Explanation

AWS Fargate platform versions refer to a specific runtime environment for Fargate task infrastructure, which is a combination of kernel and container runtime versions. New platform

versions are released as the runtime environment evolves. For example, a new version may be released for kernel or operating system updates, new features, bug fixes, or security updates. Security updates and patches are deployed automatically for your Fargate tasks. If a security issue is found that affects a platform version, AWS patches the platform version.

Usage considerations

- This control only applies to Amazon ECS services with a `LaunchType` of `FARGATE`

Remediation for rule failure

When `LaunchType` is set to `FARGATE`, set the `PlatformVersion` property to `LATEST` or omit the `PlatformVersion` property (default: `LATEST`).

The examples that follow show how to implement this remediation.

Amazon ECS Service - Example One

Amazon ECS service configured to deploy using the `LATEST` platform version by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DesiredCount": 1,
            "TaskDefinition": {
                "Ref": "ECSTaskDefinition"
            },
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "DISABLED",
                    "Subnets": [
                        {
```

```
                            "Ref": "SubnetOne"
                        },
                        {
                            "Ref": "SubnetTwo"
                        }
                    ]
                }
            },
            "LaunchType": "FARGATE"
        }
    }
}
```

YAML example

```
ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref 'ECSCluster'
    DesiredCount: 1
    TaskDefinition: !Ref 'ECSTaskDefinition'
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: DISABLED
        Subnets:
          - !Ref 'SubnetOne'
          - !Ref 'SubnetTwo'
    LaunchType: FARGATE
```

The examples that follow show how to implement this remediation.

Amazon ECS Service - Example Two

Amazon ECS service configured to deploy using the `LATEST` platform version by means of the `PlatformVersion` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DesiredCount": 1,
            "TaskDefinition": {
                "Ref": "ECSTaskDefinition"
            },
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "DISABLED",
                    "Subnets": [
                        {
                            "Ref": "SubnetOne"
                        },
                        {
                            "Ref": "SubnetTwo"
                        }
                    ]
                }
            },
            "LaunchType": "FARGATE",
            "PlatformVersion": "LATEST"
        }
    }
}
```

YAML example

```
ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref 'ECSCluster'
    DesiredCount: 1
    TaskDefinition: !Ref 'ECSTaskDefinition'
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: DISABLED
```

```
        Subnets:
          - !Ref 'SubnetOne'
          - !Ref 'SubnetTwo'
    LaunchType: FARGATE
    PlatformVersion: LATEST
```

CT.ECS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_fargate_latest_platform_version_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (Amazon ECS) Fargate
 services are configured to deploy using the 'LATEST' platform version rather than a
 specified version number.
#
# Reports on:
#   AWS::ECS::Service
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an Amazon ECS service resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS service resource
#       And: 'LaunchType' is not present
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS service resource
#       And: 'LaunchType' is present and not set to 'FARGATE'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS service resource
#       And: 'LaunchType' is present and set to 'FARGATE'
#       And: 'PlatformVersion' is present and not set to 'LATEST'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS service resource
#       And: 'LaunchType' is present and set to 'FARGATE'
#       And: 'PlatformVersion' is not present
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS service resource
#       And: 'LaunchType' is present and set to 'FARGATE'
#       And: 'PlatformVersion' is set to 'LATEST'
#      Then: PASS

#
# Constants
#
let ECS_SERVICE_TYPE = "AWS::ECS::Service"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_services = Resources.*[ Type == %ECS_SERVICE_TYPE ]

#
# Primary Rules
#
rule ecs_fargate_latest_platform_version_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                    %ecs_services not empty {
    check(%ecs_services.Properties)
        <<
        [CT.ECS.PR.1]: Require Amazon ECS Fargate Services to run on the latest Fargate
 platform version
        [FIX]: When 'LaunchType' is set to 'FARGATE', set the 'PlatformVersion'
 property to 'LATEST' or omit the 'PlatformVersion' property (default: 'LATEST').
        >>
}

rule ecs_fargate_latest_platform_version_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_SERVICE_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_SERVICE_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.1]: Require Amazon ECS Fargate Services to run on the latest Fargate
 platform version
        [FIX]: When 'LaunchType' is set to 'FARGATE', set the 'PlatformVersion'
 property to 'LATEST' or omit the 'PlatformVersion' property (default: 'LATEST').
        >>
}

#
# Parameterized Rules
#
rule check(ecs_service) {
    %ecs_service [ filter_launch_type_is_fargate(this) ]{
        # Scenario 5
        PlatformVersion not exists  or

        # Scenario 4 and 6
        check_fargate_version_latest(PlatformVersion)
    }
}

rule filter_launch_type_is_fargate(ecs_service) {
    %ecs_service {
        # Scenario 2
        LaunchType exists
        LaunchType is_string

        # Scenario 3
        LaunchType == "FARGATE"
    }
}
```

```
rule check_fargate_version_latest(property) {
    %property {
        this is_string
        this == "LATEST"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
```

```
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      CapacityProviders:
        - FARGATE
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
      Memory: '512'
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: 256
  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster:
        Ref: ECSCluster
      DesiredCount: 0
      TaskDefinition:
        Ref: ECSTaskDefinition
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: DISABLED
          Subnets:
```

```
            - Ref: SubnetOne
            - Ref: SubnetTwo
      LaunchType: FARGATE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      CapacityProviders:
        - FARGATE
  ECSTaskDefinition:
```

```
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
      Memory: '512'
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: 256
  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster:
        Ref: ECSCluster
      DesiredCount: 0
      TaskDefinition:
        Ref: ECSTaskDefinition
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: DISABLED
          Subnets:
           - Ref: SubnetOne
           - Ref: SubnetTwo
      LaunchType: FARGATE
      PlatformVersion: 1.4.0
```

[CT.ECS.PR.2] Require any Amazon ECS cluster to have container insights activated

This control checks whether your Amazon Elastic Container Service (Amazon ECS) clusters have container insights enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::Cluster`
- **CloudFormation guard rule:** CT.ECS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.2 example templates

Explanation

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon ECS clusters. Use CloudWatch container insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. CloudWatch automatically collects metrics for many resources, such as CPU, memory, disk, and network. The container insights capability also provides diagnostic information, such as container restart failures, which helps you isolate issues and resolve them quickly. You can set CloudWatch alarms on the metrics that container insights collects.

Remediation for rule failure

Enable container insights on Amazon ECS clusters with an entry in `ClusterSettings` that has `Name` set to `containerInsights` and `Value` set to `enabled`.

The examples that follow show how to implement this remediation.

Amazon ECS Cluster - Example

Amazon ECS cluster configured with container insights enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterSettings": [
                {
                    "Name": "containerInsights",
                    "Value": "enabled"
                }
```

```
            ]
        }
    }
}
```

YAML example

```
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterSettings:
      - Name: containerInsights
        Value: enabled
```

CT.ECS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_container_insights_enabled_check
#
# Description:
#   This control checks whether your Amazon Elastic Container Service (Amazon ECS)
 clusters have container insights enabled.
#
# Reports on:
#   AWS::ECS::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an Amazon ECS cluster resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS cluster resource
#       And: 'ClusterSettings' property is not present or is an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS cluster resource
#       And: 'ClusterSettings' property is present
#       And: An entry with 'Name' set to 'containerInsights' is not present in
 'ClusterSettings'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS cluster resource
#       And: 'ClusterSettings' property is present
#       And: An entry with 'Name' set to 'containerInsights' is present in
 'ClusterSettings' with a 'Value' not set
#            to 'enabled'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an Amazon ECS cluster resource
#       And: 'ClusterSettings' property is present
#       And: An entry with 'Name' set to 'containerInsights' is present in
 'ClusterSettings' with a 'Value' set to
#            'enabled'
#      Then: PASS

#
# Constants
#
let ECS_CLUSTER_TYPE = "AWS::ECS::Cluster"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let ecs_clusters = Resources.*[ Type == %ECS_CLUSTER_TYPE ]

#
# Primary Rules
#
rule ecs_container_insights_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %ecs_clusters not empty {
    check(%ecs_clusters.Properties)
        <<
        [CT.ECS.PR.2]: Require any Amazon ECS cluster to have container insights
 activated
        [FIX]: Enable container insights on Amazon ECS clusters with an entry in
 'ClusterSettings' that has 'Name' set to 'containerInsights' and 'Value' set to
 'enabled'.
        >>
}

rule ecs_container_insights_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.2]: Require any Amazon ECS cluster to have container insights
 activated
        [FIX]: Enable container insights on Amazon ECS clusters with an entry in
 'ClusterSettings' that has 'Name' set to 'containerInsights' and 'Value' set to
 'enabled'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_cluster) {
    %ecs_cluster {
        # Scenario 2
        ClusterSettings exists
        ClusterSettings is_list
        ClusterSettings not empty

        # Scenario 3, 4 and 5
        some ClusterSettings[*] {
            Name exists
```

```
            Value exists

            Name is_string
            Value is_string

            Name == "containerInsights"
            Value == "enabled"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterSettings:
      - Name: containerInsights
        Value: enabled
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterSettings:
      - Name: containerInsights
        Value: disabled
```

[CT.ECS.PR.3] Require any Amazon ECS task definition to specify a user that is not the root

This control checks whether Amazon Elastic Container Service (ECS) task definitions run as a non-root user user within Amazon ECS containers.

- **Control objective:** Enforce least privilege, Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.3 example templates

Explanation

It is a best practice to run containers as a non-root user user. By default, containers run as the root user user, unless the `User` directive is included in your Dockerfile. The default Linux capabilities that are assigned by Docker restrict the actions that can be run as the root user user, but only marginally. For example, a container running as the root user user does not have access to devices.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.

Remediation for rule failure

Set the `User` property to a non-root user user.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured with a container definition and a non-root user user. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainerA",
                    "User": "sampleuser",
                    "Memory": 256
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainerA
        User: sampleuser
        Memory: 256
```

CT.ECS.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_task_definition_nonroot_user_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definitions
 run as a non-root user within Amazon ECS containers.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' is not present or is an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present and is not an empty list
#       And: One or more containers defined in 'ContainerDefinitions' do not provide a
 'User' property
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' is present and is not an empty list
#       And: One or more containers defined in 'ContainerDefinitions' has a 'User'
 property set to a root user
#            value (0, 'root', '0:<group>', 'root:<group>')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' is present and is not an empty list
#       And: All containers defined in 'ContainerDefinitions' do not have a 'User'
 property set to a root user
#            value (0, 'root', '0:<group>', 'root:<group>')
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this
let ROOT_USER_PATTERNS = [ 0 , "0" , "root" , /^0:.*$/ , /^root:.*$/ ]
```

```
#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_task_definition_nonroot_user_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.3]: Require any Amazon ECS task definition to specify a user that is
 not the root
            [FIX]: Set the 'User' property to a non-root user.
        >>
}

rule ecs_task_definition_nonroot_user_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.3]: Require any Amazon ECS task definition to specify a user that is
 not the root
            [FIX]: Set the 'User' property to a non-root user.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [
        filter_container_definitions_is_present(this)
    ]{
        ContainerDefinitions[*] {
            # Scenario 3
            User exists

            # Scenario 4 and 5
            User not in %ROOT_USER_PATTERNS
        }
    }
}
```

```
rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
        ContainerDefinitions not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        User: exampleuser
```

```
        Memory: 256
      - Name: ExampleContainerB
        Image: alpine:latest
        User: exampleuser
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        User: exampleuser
        Memory: 256
      - Name: ExampleContainerB
        Image: alpine:latest
        User: root
```

[CT.ECS.PR.4] Require Amazon ECS tasks to use 'awsvpc' networking mode

This control checks whether the networking mode for Amazon Elastic Container Service (ECS) task definitions is set to `awsvpc`.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.4 example templates

Explanation

Amazon ECS recommends using the `awsvpc` network mode, unless you have a specific reason to use a different network mode. The `awsvpc` network mode simplifies container networking, and it gives you control over the ways that containerized applications communicate with each other, or with other services, within your VPCs. The `awsvpc` network mode provides improved security for your containers, because it empowers you to use security groups and network monitoring tools at a detailed level within your tasks. Each task has its own elastic network interface (ENI); therefore, you can include other Amazon EC2 networking features, such as VPC Flow Logs, which help you monitor traffic among your tasks.

Usage considerations

- This control applies only to Amazon ECS task definitions for launch types of `EC2` and `Fargate`.

Remediation for rule failure

Set `NetworkMode` to `awsvpc` for Amazon ECS tasks that deploy to Amazon EC2 or AWS Fargate.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured with `awsvpc` networking mode. The example is shown in JSON and in YAML.

JSON example

```
{
    "Resources": {
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
```

```
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Essential": true,
                        "Image": "nginx:latest",
                        "Name": "SampleContainer"
                    }
                ],
                "Memory": 512,
                "NetworkMode": "awsvpc"
            }
        }
    }
}
```

YAML example

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Essential: true
          Image: nginx:latest
          Name: SampleContainer
      Memory: 512
      NetworkMode: awsvpc
```

CT.ECS.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_awsvpc_networking_enabled_check
#
# Description:
```

```
#   This control checks whether the networking mode for Amazon Elastic Container
 Service (ECS) task definitions is set to 'awsvpc'.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'RequiresCompatibilities' is present and only has one entry in the list
 set to 'EXTERNAL'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'RequiresCompatibilities' is either not present or set to a list with
 entries that include 'EC2',
#            'FARGATE' or both.
#       And: 'NetworkMode' is not present
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'RequiresCompatibilities' is either not present or set to a list with
 entries that include 'EC2',
#            'FARGATE' or both.
#       And: 'NetworkMode' is present
#       And: 'NetworkMode' is not set to 'awsvpc'
#      Then: FAIL
```

```
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'RequiresCompatibilities' is either not present or set to a list with
 entries that include 'EC2',
#            'FARGATE' or both.
#       And: 'NetworkMode' is present
#       And: 'NetworkMode' is set to 'awsvpc'
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let ALLOWED_NETWORK_MODES = [ "awsvpc" ]
let SUPPORTED_LAUNCH_PLATFORMS = [ "EC2" , "FARGATE" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_awsvpc_networking_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.4]: Require Amazon ECS tasks to use 'awsvpc' networking mode
            [FIX]: Set 'NetworkMode' to 'awsvpc' for Amazon ECS tasks that deploy to
 Amazon EC2 or AWS Fargate.
        >>
}

rule ecs_awsvpc_networking_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.4]: Require Amazon ECS tasks to use 'awsvpc' networking mode
            [FIX]: Set 'NetworkMode' to 'awsvpc' for Amazon ECS tasks that deploy to
 Amazon EC2 or AWS Fargate.
```

```
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [ filter_external_task_definitions(this) ] {
        # Scenario 3
        NetworkMode exists

        # Scenario 4 and 5
        NetworkMode is_string
        NetworkMode in %ALLOWED_NETWORK_MODES
    }
}

rule filter_external_task_definitions(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        RequiresCompatibilities not exists or
        filter_supported_task_definitions(RequiresCompatibilities)
    }
}

rule filter_supported_task_definitions(requires_compatibilities) {
    %requires_compatibilities {
        this is_list
        this not empty
        some this[*] in %SUPPORTED_LAUNCH_PLATFORMS
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainer
      Memory: 512
      NetworkMode: awsvpc
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainer
      Memory: 512
```

[CT.ECS.PR.5] Require an active Amazon ECS task definition to have a logging configuration

This control checks whether Amazon Elastic Container Service (ECS) task definitions have a logging configuration specified.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.5 example templates

Explanation

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Elastic Container Service (ECS) and your AWS environments. We recommend that you collect monitoring data from all parts of your AWS environment, because this data can help you debug a multi-point failure, if such a failure occurs.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.

Remediation for rule failure

For each container definition, within `LogConfiguration` set the `LogDriver` property to a supported log driver.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured to send log information to Amazon CloudWatch Logs for each container definition. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "ContainerDefinitions": [
                {
                    "Name": "ContainerA",
                    "Image": "nginx:latest",
                    "Essential": true,
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-stream-prefix": "Container-A-LogStream",
                            "awslogs-region": {
                                "Ref": "AWS::Region"
                            }
                        }
                    }
                },
                {
                    "Name": "ContainerB",
                    "Image": "nginx:latest",
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-stream-prefix": "Container-B-LogStream",
                            "awslogs-region": {
```

```
                                "Ref": "AWS::Region"
                            }
                        }
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    ContainerDefinitions:
      - Name: ContainerA
        Image: nginx:latest
        Essential: true
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: !Ref 'LogGroup'
            awslogs-stream-prefix: Container-A-LogStream
            awslogs-region: !Ref 'AWS::Region'
      - Name: ContainerB
        Image: nginx:latest
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: !Ref 'LogGroup'
            awslogs-stream-prefix: Container-B-LogStream
            awslogs-region: !Ref 'AWS::Region'
```

CT.ECS.PR.5 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   ecs_task_definition_log_configuration_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definitions
 have a logging configuration specified.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is not present or is an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present and is not an empty list
#       And: One or more containers defined  in 'ContainerDefinitions' do not have
 'LogConfiguration' set or it is set
#            to an empty struct
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
```

```
#       And: 'ContainerDefinitions' property is present and is not an empty list
#       And: One or more containers defined  in 'ContainerDefinitions' have
 'LogConfiguration' property present
#       And: 'LogConfiguration.LogDriver' is not present or is set to an empty string
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present
#       And: All containers defined  in 'ContainerDefinitions' have 'LogConfiguration'
 property present
#       And: 'LogConfiguration.LogDriver' is present and set to a non-empty string
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_task_definition_log_configuration_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.5]: Require an active Amazon ECS task definition to have a logging
 configuration
            [FIX]: For each container definition, within 'LogConfiguration' set the
 'LogDriver' property to a supported log driver.
        >>
}

rule ecs_task_definition_log_configuration_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
```

```
        [CT.ECS.PR.5]: Require an active Amazon ECS task definition to have a logging
 configuration
            [FIX]: For each container definition, within 'LogConfiguration' set the
 'LogDriver' property to a supported log driver.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [
        filter_container_definitions_is_present(this)
    ]{
        ContainerDefinitions[*] {
            # Scenario 3
            LogConfiguration exists
            LogConfiguration is_struct
            LogConfiguration not empty

            # Scenario 4 and 5
            LogConfiguration {
                LogDriver exists
                check_is_string_and_not_empty(LogDriver)
            }
        }
    }
}

rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
        ContainerDefinitions not empty
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
```

```
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Delete
    UpdateReplacePolicy: Delete
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Name: ContainerA
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group:
              Ref: LogGroup
            awslogs-stream-prefix: Container-A-LogStream
            awslogs-region:
```

```
              Ref: AWS::Region
        Essential: true
        Image: nginx:latest
      - Name: ContainerB
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group:
              Ref: LogGroup
            awslogs-stream-prefix: Container-B-LogStream
            awslogs-region:
              Ref: AWS::Region
        Image: nginx:latest
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Delete
    UpdateReplacePolicy: Delete
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Name: ContainerA
        Essential: true
        Image: nginx:latest
      - Name: ContainerB
        Image: nginx:latest
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group:
              Ref: LogGroup
            awslogs-stream-prefix: Container-B-LogStream
            awslogs-region:
              Ref: AWS::Region
```

[CT.ECS.PR.6] Require Amazon ECS containers to allow read-only access to the root filesystem

This control checks whether Amazon Elastic Container Service (Amazon ECS) task definitions have been configured to require read-only access to container root filesystems.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.6 example templates

Explanation

Enabling this option reduces security attack vectors. When it is enabled, the container instance's filesystem cannot be tampered with or written to unless it has explicitly granted read-write permissions on its filesystem folder and directories. This control adheres to the principle of least privilege.

Usage considerations

- This control is incompatible with Amazon ECS task definitions that use Windows containers.

Remediation for rule failure

Set the `ReadonlyRootFilesystem` property to `true` for all `ContainerDefinitions`.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition with read-only access to container root filesystems. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainerA",
                    "ReadonlyRootFilesystem": true
                },
                {
                    "Image": "alpine:latest",
                    "Name": "SampleContainerB",
                    "ReadonlyRootFilesystem": true
                }
            ],
            "Memory": "512"
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
```

```
  Properties:
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainerA
        ReadonlyRootFilesystem: true
      - Image: alpine:latest
        Name: SampleContainerB
        ReadonlyRootFilesystem: true
    Memory: '512'
```

CT.ECS.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
# Rule Identifier:
#   ecs_containers_readonly_access_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (Amazon ECS) task
 definitions have been configured to require read-only access to container root
 filesystems.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
# Scenario: 1
#   Given: The input document is an CloudFormation document or CloudFormation hook
 document
#     And: The input document does not contain an Amazon ECS task definition resource
#    Then: SKIP
# Scenario: 2
```

```
#   Given: The input document is an CloudFormation document or CloudFormation hook
 document
#     And: The input document contains an Amazon ECS task definition resource
#     And: 'ContainerDefinitions' property is not present or is empty
#    Then: SKIP
# Scenario: 3
#   Given: The input document is an CloudFormation document or CloudFormation hook
 document
#     And: The input document contains an Amazon ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: One or more containers defined  in 'ContainerDefinitions' do not have
 'ReadonlyRootFilesystem' present
#    Then: FAIL
# Scenario: 4
#   Given: The input document is an CloudFormation document or CloudFormation hook
 document
#     And: The input document contains an Amazon ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: One or more containers defined  in 'ContainerDefinitions' have the value of
 'ReadonlyRootFilesystem' set to
#          bool(false)
#     Then: FAIL
# Scenario: 5
#   Given: The input document is an CloudFormation document or CloudFormation hook
 document
#     And: The input document contains an Amazon ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: All containers defined  in 'ContainerDefinitions' have the value of
 'ReadonlyRootFilesystem' set to
#          bool(true)
#    Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
```

```
# Primary Rules
#
rule ecs_containers_readonly_access_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.6]: Require Amazon ECS containers to allow read-only access to the
 root filesystem
        [FIX]: Set the 'ReadonlyRootFilesystem' property to 'true' for all
 'ContainerDefinitions'.
        >>
}

rule ecs_containers_readonly_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.6]: Require Amazon ECS containers to allow read-only access to the
 root filesystem
        [FIX]: Set the 'ReadonlyRootFilesystem' property to 'true' for all
 'ContainerDefinitions'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [ filter_container_definitions_is_present(this) ]{
        ContainerDefinitions[*] {
            # Scenario 3
            ReadonlyRootFilesystem exists

            # Scenario 4
            ReadonlyRootFilesystem == true
        }
    }
}

rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
```

```
        ContainerDefinitions not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainerA
        ReadonlyRootFilesystem: true
      - Image: alpine:latest
        Name: SampleContainerB
        ReadonlyRootFilesystem: true
      Memory: '512'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainerA
        ReadonlyRootFilesystem: false
      - Image: alpine:latest
        Name: SampleContainerB
        ReadonlyRootFilesystem: false
      Memory: '512'
```

[CT.ECS.PR.7] Require an Amazon ECS task definition to have a specific memory usage limit

This control checks whether Amazon Elastic Container Service (ECS) task definitions have specified a memory limit for container definitions.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.7 example templates

Explanation

We recommend that you specify the maximum memory available to your containers, because this limit protects your resources in the event of malicious access to your containers.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.

Remediation for rule failure

Set the `Memory` property in `ContainerDefinitions` for Amazon ECS task definitions.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured with a specified memory limit for container definitions. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainer",
                    "Memory": 256
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
        Memory: 256
```

CT.ECS.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_task_definition_memory_hard_limit_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definitions
 have specified a memory limit for container definitions.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
```

```
#   Scenario: 2
#    Given: The input document is an CloudFormation document or CloudFormation hook
 document
#      And: The input document contains an ECS task definition resource
#      And: 'ContainerDefinitions' property is not present or is an empty list
#     Then: SKIP
#   Scenario: 3
#    Given: The input document is an CloudFormation document or CloudFormation hook
 document
#      And: The input document contains an ECS task definition resource
#      And: 'ContainerDefinitions' property is present and is not an empty list
#      And: One or more containers defined in 'ContainerDefinitions' do not have
 'Memory' property set
#     Then: FAIL
#   Scenario: 4
#    Given: The input document is an CloudFormation document or CloudFormation hook
 document
#      And: The input document contains an ECS task definition resource
#      And: 'ContainerDefinitions' property is present and is not an empty list
#      And: One or more containers defined in 'ContainerDefinitions' have 'Memory'
 property set to an integer
#           value less than four (< 4)
#     Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present
#       And: All containers defined in 'ContainerDefinitions' have 'Memory' property
 set to an integer value
#            greater than or equal to four (>= 4)
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]
```

```
#
# Primary Rules
#
rule ecs_task_definition_memory_hard_limit_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.7]: Require an Amazon ECS task definition to have a specific memory
 usage limit
            [FIX]: Set the 'Memory' property in 'ContainerDefinitions' for Amazon ECS
 task definitions.
        >>
}

rule ecs_task_definition_memory_hard_limit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.7]: Require an Amazon ECS task definition to have a specific memory
 usage limit
            [FIX]: Set the 'Memory' property in 'ContainerDefinitions' for Amazon ECS
 task definitions.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [filter_container_definitions_is_present(this)]{
        ContainerDefinitions[*] {
            # Scenario 3
            Memory exists

            # Scenario 4 and 5
            Memory >= 4
        }
    }
}

rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
```

```
        ContainerDefinitions is_list
        ContainerDefinitions not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        Memory: 256
      - Image: alpine:latest
        Name: ExampleContainerB
        Memory: 512
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: "512"
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        Memory: 256
      - Image: alpine:latest
        Name: ExampleContainerB
        Memory: 3
```

[CT.ECS.PR.8] Require Amazon ECS task definitions to have secure networking modes and user definitions

This control checks whether Amazon Elastic Container Service (ECS) task definitions that use `host` networking mode have a privileged container definition, and whether they specify a non-root user definition.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.8 example templates

Explanation

If a task definition has elevated privileges, it implies that a customer has specifically chosen that configuration. This control checks for unexpected privilege escalation, which occurs when a task definition enables host networking, but a customer has not opted into elevated privileges.

Usage considerations

- This control applies only to Amazon ECS task definitions that include `host` networking mode and one or more container definitions.
- This control is incompatible with Amazon ECS task definitions that use Windows containers.

Remediation for rule failure

For Amazon ECS task definitions that use `host` networking mode, your container definitions must set the `User` property to a non-root user. Also, to opt into elevated privileges, configure containers to run in privileged mode by setting the `Privileged` property to `true`.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example One

Amazon ECS task definition with host networking mode configured for privileged container definitions. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "NetworkMode": "host",
            "ContainerDefinitions": [
                {
                    "Name": "SampleContainerA",
                    "User": "root",
                    "Privileged": true,
```

```
                    "Image": "nginx:latest",
                    "Essential": true
                },
                {
                    "Name": "SampleContainerB",
                    "User": "root",
                    "Privileged": true,
                    "Image": "alpine:latest"
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    NetworkMode: host
    ContainerDefinitions:
      - Name: SampleContainerA
        User: root
        Privileged: true
        Image: nginx:latest
        Essential: true
      - Name: SampleContainerB
        User: root
        Privileged: true
        Image: alpine:latest
```

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example Two

Amazon ECS task definition with host networking mode configured for non-root user container definitions and privileged mode deactivated, by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "NetworkMode": "host",
            "ContainerDefinitions": [
                {
                    "Name": "SampleContainerA",
                    "User": "root",
                    "Privileged": true,
                    "Image": "nginx:latest",
                    "Essential": true
                },
                {
                    "Name": "SampleContainerB",
                    "User": "root",
                    "Privileged": true,
                    "Image": "alpine:latest"
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    NetworkMode: host
    ContainerDefinitions:
      - Name: SampleContainerA
        User: root
        Privileged: true
        Image: nginx:latest
        Essential: true
```

```
      - Name: SampleContainerB
        User: root
        Privileged: true
        Image: alpine:latest
```

CT.ECS.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_task_definition_user_for_host_mode_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definitions
 that use 'host' networking mode have a privileged container definition, and whether
 they specify a non-root user definition.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is not present or is an empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present and is not an empty list
#       And: 'NetworkMode' property is either not present or set to a value other than
 'host'
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present and is not an empty list
#       And: 'NetworkMode' property is present and set to 'host'
#       And: A container defined in 'ContainerDefinitions' has 'Privileged' property
 not set or is set as bool(false)
#       And: This same container either does not have the 'User' property set or has it
 set to a value that translates
#            to root user
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present
#       And: 'NetworkMode' property is present and set to 'host'
#       And: All Containers defined in 'ContainerDefinitions' either have the
 'Privileged' property set to bool(true)
#            or have their 'User' property set to a value that does not translate to
 root user
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this
let ROOT_USER_PATTERNS = [ 0 , "0" , "root" , /^0:.*$/ , /^root:.*$/ ]
let VALID_NETWORK_MODES = [ "host" ]

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_task_definition_user_for_host_mode_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                        %ecs_task_definitions not empty
 {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.8]: Require Amazon ECS task definitions to have secure networking
 modes and user definitions
            [FIX]: For Amazon ECS task definitions that use 'host' networking mode,
 your container definitions must set the 'User' property to a non-root user. Also,
 to opt into elevated privileges, configure containers to run in privileged mode by
 setting the  'Privileged' property to 'true'.
        >>
}

rule ecs_task_definition_user_for_host_mode_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.8]: Require Amazon ECS task definitions to have secure networking
 modes and user definitions
            [FIX]: For Amazon ECS task definitions that use 'host' networking mode,
 your container definitions must set the 'User' property to a non-root user. Also,
 to opt into elevated privileges, configure containers to run in privileged mode by
 setting the  'Privileged' property to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [
        filter_nw_mode_container_definitions(this)
    ]{
        ContainerDefinitions[*] {
            # Scenario 4 and 5
            check_elevated_privilege_containers(this) or
            check_nonroot_user_containers(this)
        }
    }
}

rule check_elevated_privilege_containers(container_definition) {
    %container_definition {
        Privileged exists
```

```
        Privileged == true
    }
}

rule check_nonroot_user_containers(container_definition) {
    %container_definition {
        User exists
        User not in %ROOT_USER_PATTERNS
    }
}

rule filter_nw_mode_container_definitions(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
        ContainerDefinitions not empty

        # Scenario 3
        NetworkMode exists
        NetworkMode is_string
        NetworkMode in %VALID_NETWORK_MODES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      NetworkMode: host
      ContainerDefinitions:
      - Name: ExampleContainerA
        User: root
        Privileged: true
        Image: nginx:latest
        Essential: true
      - Name: ExampleContainerB
        User: root
        Privileged: true
        Image: alpine:latest
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      NetworkMode: host
      ContainerDefinitions:
      - Name: ExampleContainerA
        User: root
        Privileged: true
        Image: nginx:latest
        Essential: true
```

```
      - Name: ExampleContainerB
        Image: alpine:latest
        User: root
```

[CT.ECS.PR.9] Require Amazon ECS services not to assign public IP addresses automatically

This control checks whether your Amazon Elastic Container Service (Amazon ECS) service resources are configured to assign public IP addresses automatically.

- **Control objective:** Limit network access, Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::Service`
- **CloudFormation guard rule:** CT.ECS.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.9 example templates

Explanation

A public IP address is an IP address that is reachable from the internet. If you launch your Amazon ECS instances with a public IP address, then your Amazon ECS instances are reachable from the internet. Amazon ECS services should not be publicly accessible, because it may allow unintended access to your container application servers.

Remediation for rule failure

Set `AssignPublicIp` in `NetworkConfiguration.AwsvpcConfiguration` to `DISABLED`.

The examples that follow show how to implement this remediation.

Amazon ECS Service - Example One

Amazon ECS service configured to disallow automatic public IP address assignment, by means of AWS CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DesiredCount": 0,
            "TaskDefinition": {
                "Ref": "ECSTaskDefinition"
            },
            "LaunchType": "FARGATE",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "Subnets": [
                        {
                            "Ref": "SubnetOne"
                        },
                        {
                            "Ref": "SubnetTwo"
                        }
                    ]
                }
            }
        }
    }
}
```

YAML example

```
ECSService:
  Type: AWS::ECS::Service
  Properties:
```

```
    Cluster: !Ref 'ECSCluster'
    DesiredCount: 0
    TaskDefinition: !Ref 'ECSTaskDefinition'
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsvpcConfiguration:
        Subnets:
          - !Ref 'SubnetOne'
          - !Ref 'SubnetTwo'
```

The examples that follow show how to implement this remediation.

Amazon ECS Service - Example Two

Amazon ECS service configured to disallow automatic public IP address assignment, by means of the `AssignPublicIp` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DesiredCount": 0,
            "TaskDefinition": {
                "Ref": "ECSTaskDefinition"
            },
            "LaunchType": "FARGATE",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "DISABLED",
                    "Subnets": [
                        {
                            "Ref": "SubnetOne"
                        },
                        {
                            "Ref": "SubnetTwo"
                        }
```

```
                    ]
                }
            }
        }
    }
}
```

YAML example

```
ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref 'ECSCluster'
    DesiredCount: 0
    TaskDefinition: !Ref 'ECSTaskDefinition'
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: DISABLED
        Subnets:
          - !Ref 'SubnetOne'
          - !Ref 'SubnetTwo'
```

CT.ECS.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_service_assign_public_ip_disabled_check
#
# Description:
#   This control checks whether your Amazon Elastic Container Service (Amazon ECS)
 service resources are configured to assign public IP addresses automatically.
#
# Reports on:
#   AWS::ECS::Service
```

```
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain an Amazon ECS service resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECS service resource
#       And: 'NetworkConfiguration' property is not present
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECS service resource
#       And: 'NetworkConfiguration.AwsvpcConfiguration' property is not present
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECS service resource
#       And: 'NetworkConfiguration.AwsvpcConfiguration' property is present
#       And: 'AssignPublicIp' property is present and set to 'ENABLED'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECS service resource
#       And: 'NetworkConfiguration.AwsvpcConfiguration' property is present
#       And: 'AssignPublicIp' property is not present
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon ECS service resource
#       And: 'NetworkConfiguration.AwsvpcConfiguration' property is present
#       And: 'AssignPublicIp' property is present and set to 'DISABLED'
#      Then: PASS

#
# Constants
#
```

```
let ECS_SERVICE_TYPE = "AWS::ECS::Service"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_services = Resources.*[ Type == %ECS_SERVICE_TYPE ]

#
# Primary Rules
#
rule ecs_service_assign_public_ip_disabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %ecs_services not empty {
    check(%ecs_services.Properties)
        <<
        [CT.ECS.PR.9]: Require Amazon ECS services not to assign public IP addresses
 automatically
        [FIX]: Set 'AssignPublicIp' in 'NetworkConfiguration.AwsvpcConfiguration' to
 'DISABLED'.
        >>
}

rule ecs_service_assign_public_ip_disabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_SERVICE_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_SERVICE_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.9]: Require Amazon ECS services not to assign public IP addresses
 automatically
        [FIX]: Set 'AssignPublicIp' in 'NetworkConfiguration.AwsvpcConfiguration' to
 'DISABLED'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_service) {
    %ecs_service [filter_ecs_service_with_vpc_configuration(this)] {
        NetworkConfiguration {
            AwsvpcConfiguration {
                # Scenario 5
                AssignPublicIp not exists or

                # Scenario 6
```

```
                check_assign_public_ip_property(AssignPublicIp)
            }
        }
    }
}

rule check_assign_public_ip_property(public_ip) {
    %public_ip {
        this is_string
        this == "DISABLED"
    }
}

rule filter_ecs_service_with_vpc_configuration(ecs_service) {
    %ecs_service {
        # Scenario 2
        NetworkConfiguration exists
        NetworkConfiguration is_struct

        # Scenarion 3 and 4
        NetworkConfiguration {
            AwsvpcConfiguration exists
            AwsvpcConfiguration is_struct
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      CapacityProviders:
        - FARGATE
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
```

```
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
      Memory: '512'
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: 256
  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster:
        Ref: ECSCluster
      DesiredCount: 0
      TaskDefinition:
        Ref: ECSTaskDefinition
      NetworkConfiguration:
        AwsvpcConfiguration:
          Subnets:
           - Ref: SubnetOne
           - Ref: SubnetTwo
      LaunchType: FARGATE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
```

```
      Fn::Select:
      - 0
      - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AavailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      CapacityProviders:
        - FARGATE
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
      Memory: '512'
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: 256
  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster:
        Ref: ECSCluster
      DesiredCount: 0
      TaskDefinition:
        Ref: ECSTaskDefinition
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          Subnets:
           - Ref: SubnetOne
           - Ref: SubnetTwo
```

```
      LaunchType: FARGATE
```

[CT.ECS.PR.10] Require that Amazon ECS task definitions do not share the host's process namespace

This control checks whether Amazon Elastic Container Service (ECS) task definitions are configured to share a host's process namespace with its containers.

- **Control objective:** Protect configurations, Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.10 example templates

Explanation

A process ID (PID) namespace provides separation between processes. It prevents system processes from being visible to other processes, and it allows PIDs to be reused, including PID 1. If the hosts `PID namespace is shared with containers, those containers can see all of the processes on the host system. Process visibility reduces the benefit of process-level isolation between the host and the containers. Reduced isolation can allow unauthorized access to processes on the host itself, including the ability to manipulate and terminate the hosts` processes. As a best practice, do not share the host's process namespace with containers running on the host.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.
- This control is not compatible with Amazon ECS task definitions that are configured to run on AWS Fargate, or definitions that use Windows containers.

Remediation for rule failure

Omit the `PidMode` property, or set `PidMode` to `task`.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example One

Amazon ECS task definition configured with a task-level process namespace, by means of CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "TaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainer"
                }
            ]
        }
    }
}
```

YAML example

```
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
```

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example Two

Amazon ECS task definition configured with a task-level process namespace, by means of the `PidMode` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "TaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainer"
                }
            ],
            "PidMode": "task"
        }
    }
}
```

YAML example

```
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
    PidMode: task
```

CT.ECS.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   ecs_task_definition_pid_mode_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definitions
 are configured to share a host's process namespace with its containers.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
```

```
#       And: 'PidMode' is provided as an empty string
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'PidMode' is set to 'host'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'PidMode' is not present
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'PidMode' is provided as a non-empty string that is not 'host'
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_task_definition_pid_mode_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.10]: Require that Amazon ECS task definitions do not share the
 host's process namespace
            [FIX]: Omit the 'PidMode' property, or set 'PidMode' to 'task'.
        >>
}

rule ecs_task_definition_pid_mode_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
        [CT.ECS.PR.10]: Require that Amazon ECS task definitions do not share the
 host's process namespace
            [FIX]: Omit the 'PidMode' property, or set 'PidMode' to 'task'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        PidMode not exists or

        # Scenario 3 and 4
        check_pidmode_value(PidMode)
    }
}

rule check_pidmode_value(pid_mode) {
    %pid_mode {
        check_is_string_and_not_empty(this)
        this != "host"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
```

```
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.ECS.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainer
      PidMode: task
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainer
      PidMode: host
```

[CT.ECS.PR.11] Require an Amazon ECS container to run as non-privileged

This control checks whether container definitions in Amazon Elastic Container Service (ECS) task definitions are configured with elevated privileges.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.11 example templates

Explanation

We recommend that you remove elevated privileges from your Amazon ECS task definitions. When the privilege parameter is `true`, the container can operate with elevated privileges on the host container instance. These privileges are similar to the root user privileges.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.
- This control is incompatible with Amazon ECS task definitions that use Windows containers.

Remediation for rule failure

Be sure that all containers defined in `ContainerDefinitions` either omit the `Privileged` property, or that they set `Privileged` to `false`.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured with privileged mode deactivated for container definitions. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "alpine:latest",
                    "Name": "SampleContainerA"
                },
                {
                    "Image": "nginx:latest",
                    "Name": "SampleContainerB",
                    "Privileged": false
                }
            ],
            "Memory": "512"
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
```

```
    ContainerDefinitions:
      - Essential: true
        Image: alpine:latest
        Name: SampleContainerA
      - Image: nginx:latest
        Name: SampleContainerB
        Privileged: false
    Memory: '512'
```

CT.ECS.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   ecs_containers_nonprivileged_check
#
# Description:
#   This control checks whether container definitions in Amazon Elastic Container
 Service (ECS) task definitions are configured with elevated privileges.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain an ECS task definition resource
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is not present or is an empty list
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present
#       And: One or more containers defined  in 'ContainerDefinitions' have
 'Privileged' set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ECS task definition resource
#       And: 'ContainerDefinitions' property is present
#       And: All containers defined in 'ContainerDefinitions' either do not have the
 'Privileged' property present or
#            'Privileged' is present and set to bool(false)
#      Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_containers_nonprivileged_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.11]: Require an Amazon ECS container to run as non-privileged
            [FIX]: Be sure that all containers defined in 'ContainerDefinitions' either
 omit the 'Privileged' property, or that they set 'Privileged' to 'false'.
        >>
}

rule ecs_containers_nonprivileged_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
```

```
        <<
        [CT.ECS.PR.11]: Require an Amazon ECS container to run as non-privileged
            [FIX]: Be sure that all containers defined in 'ContainerDefinitions' either
 omit the 'Privileged' property, or that they set 'Privileged' to 'false'.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [
        filter_container_definitions_is_present(this)
    ]{
        ContainerDefinitions[*] {
            # Scenario 3
            Privileged not exists or

            # Scenario 4
            Privileged == false
        }
    }
}

rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
        ContainerDefinitions not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: alpine:latest
        Name: ExampleContainerA
      - Image: nginx:latest
        Name: ExampleContainerB
        Privileged: false
      Memory: '512'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        Privileged: false
      - Essential: true
```

```
        Image: alpine:latest
        Name: ExampleContainerB
      - Image: nginx:latest
        Name: ExampleContainerC
        Privileged: true
      Memory: '512'
```

[CT.ECS.PR.12] Require that Amazon ECS task definitions do not pass secrets as container environment variables

This control checks whether Amazon Elastic Container Service (ECS) task definition container definitions include environment variables named `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, or `ECS_ENGINE_AUTH_DATA`.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ECS::TaskDefinition`
- **CloudFormation guard rule:** CT.ECS.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ECS.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ECS.PR.12 example templates

Explanation

AWS Systems Manager Parameter Store can help you improve the security posture of your organization. We recommend the Parameter Store as a way to store secrets and credentials, instead of passing them into your container instances or entering them into your source code.

Usage considerations

- This control applies only to Amazon ECS task definitions that are configured with container definitions.
- This control evaluates plaintext environment variables configured directly on container definitions.

Remediation for rule failure

Omit environment variables with `Name` set to `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` or `ECS_ENGINE_AUTH_DATA` from container definitions.

The examples that follow show how to implement this remediation.

Amazon ECS Task Definition - Example

Amazon ECS task definition configured to inject sensitive data into a container as an environment variable. The example is shown in JSON and in YAML.

JSON example

```
{
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Memory": "512",
            "ExecutionRoleArn": {
                "Fn::GetAtt": [
                    "ECSTaskExecutionRole",
                    "Arn"
                ]
            },
            "ContainerDefinitions": [
                {
                    "Essential": true,
                    "Image": "nginx:latest",
                    "Name": "SampleContainer",
                    "Environment": [
                        {
                            "Name": "SAMPLE_ENV_VAR",
```

```
                            "Value": "sampleValue"
                        }
                    ],
                    "Secrets": [
                        {
                            "Name": "SAMPLE_SENSITIVE_ENV_VAR",
                            "ValueFrom": "arn:aws:ssm:us-east-1:123456789012:parameter/
sample_parameter"
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: '512'
    ExecutionRoleArn: !GetAtt 'ECSTaskExecutionRole.Arn'
    ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: SampleContainer
        Environment:
          - Name: SAMPLE_ENV_VAR
            Value: sampleValue
        Secrets:
          - Name: SAMPLE_SENSITIVE_ENV_VAR
            ValueFrom: arn:aws:ssm:us-east-1:123456789012:parameter/sample_parameter
```

CT.ECS.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   ecs_no_environment_secrets_check
#
# Description:
#   This control checks whether Amazon Elastic Container Service (ECS) task definition
 container definitions include environment variables named 'AWS_ACCESS_KEY_ID',
 'AWS_SECRET_ACCESS_KEY', or 'ECS_ENGINE_AUTH_DATA'.
#
# Reports on:
#   AWS::ECS::TaskDefinition
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
# Scenario: 1
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document does not contain an ECS task definition resource
#    Then: SKIP
# Scenario: 2
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an ECS task definition resource
#     And: 'ContainerDefinitions' property is not present or is empty
#    Then: SKIP
# Scenario: 3
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: Containers defined in 'ContainerDefinitions' do not have 'Environment'
 property present
#    Then: SKIP
# Scenario: 4
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: One or more containers defined  in 'ContainerDefinitions' have 'Environment'
 present
#     And: 'Environment' property has an entry with 'Name' set to 'AWS_ACCESS_KEY_ID',
 'AWS_SECRET_ACCESS_KEY', or
```

```
#           'ECS_ENGINE_AUTH_DATA'
#     Then: FAIL
# Scenario: 5
#   Given: The input document is an CloudFormation or CloudFormation hook document
#     And: The input document contains an ECS task definition resource
#     And: 'ContainerDefinitions' property is present
#     And: One or more containers defined  in 'ContainerDefinitions' have 'Environment'
 present
#     And: 'Environment' property does not have an entry with 'Name' set to
 'AWS_ACCESS_KEY_ID', 'AWS_SECRET_ACCESS_KEY',
#           or 'ECS_ENGINE_AUTH_DATA'
#    Then: PASS

#
# Constants
#
let ECS_TASK_DEFINITION_TYPE = "AWS::ECS::TaskDefinition"
let INPUT_DOCUMENT = this
let RESTRICTED_ENVIRONMENT_VARIABLES = ["AWS_ACCESS_KEY_ID", "AWS_SECRET_ACCESS_KEY",
 "ECS_ENGINE_AUTH_DATA"]

#
# Assignments
#
let ecs_task_definitions = Resources.*[ Type == %ECS_TASK_DEFINITION_TYPE ]

#
# Primary Rules
#
rule ecs_no_environment_secrets_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %ecs_task_definitions not empty {
    check(%ecs_task_definitions.Properties)
        <<
        [CT.ECS.PR.12]: Require that Amazon ECS task definitions do not pass secrets as
 container environment variables
            [FIX]: Omit environment variables with 'Name' set to 'AWS_ACCESS_KEY_ID',
 'AWS_SECRET_ACCESS_KEY' or 'ECS_ENGINE_AUTH_DATA' from container definitions.
        >>
}

rule ecs_no_environment_secrets_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ECS_TASK_DEFINITION_TYPE) {
    check(%INPUT_DOCUMENT.%ECS_TASK_DEFINITION_TYPE.resourceProperties)
        <<
```

```
        [CT.ECS.PR.12]: Require that Amazon ECS task definitions do not pass secrets as
 container environment variables
            [FIX]: Omit environment variables with 'Name' set to 'AWS_ACCESS_KEY_ID',
 'AWS_SECRET_ACCESS_KEY' or 'ECS_ENGINE_AUTH_DATA' from container definitions.
        >>
}

#
# Parameterized Rules
#
rule check(ecs_task_definition) {
    %ecs_task_definition [
        filter_container_definitions_is_present(this)
    ]{
        ContainerDefinitions[
            filter_environment_is_present(this)
        ] {
            # Scenario 4 and 5
            Environment[*] {
                Name not in %RESTRICTED_ENVIRONMENT_VARIABLES
            }
        }
    }
}

rule filter_container_definitions_is_present(ecs_task_definition) {
    %ecs_task_definition {
        # Scenario 2
        ContainerDefinitions exists
        ContainerDefinitions is_list
        ContainerDefinitions not empty
    }
}

rule filter_environment_is_present(container_definition) {
    %container_definition {
        # Scenario 3
        Environment exists
        Environment is_list
        Environment not empty
    }
}

#
```

```
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ECS.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ecs-tasks.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
      Policies:
      - PolicyName: ECSTaskPolicy
        PolicyDocument:
          Version: '2012-10-17       '
```

```
          Statement:
          - Effect: Allow
            Action:
            - ssm:GetParameters
            Resource: arn:aws:ssm:us-east-1:123456789012:parameter/example_parameter
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ExecutionRoleArn:
        Fn::GetAtt: [ ECSTaskExecutionRole, Arn ]
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainer
        Environment:
        - Name: EXAMPLE_ENV_VAR
          Value: exampleValue
        Secrets:
        - Name: EXAMPLE_SENSITIVE_ENV_VAR
          ValueFrom: arn:aws:ssm:us-east-1:123456789012:parameter/example_parameter
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Memory: '512'
      ContainerDefinitions:
      - Essential: true
        Image: nginx:latest
        Name: ExampleContainerA
        Environment:
        - Name: AWS_ACCESS_KEY_ID
          Value: exampleKey
        - Name: AWS_SECRET_ACCESS_KEY
          Value: exampleSecretKey
```

```
      - Image: alpine:latest
        Name: ExampleContainerB
```

Amazon Elastic File System controls

Topics

- [CT.ELASTICFILESYSTEM.PR.1] Require an Amazon EFS file system to encrypt file data at rest using AWS KMS
- [CT.ELASTICFILESYSTEM.PR.2] Require an Amazon EFS volume to have an automated backup plan
- [CT.ELASTICFILESYSTEM.PR.3] Require Amazon EFS access points to have a root directory
- [CT.ELASTICFILESYSTEM.PR.4] Require Amazon EFS access points to enforce a user identity

[CT.ELASTICFILESYSTEM.PR.1] Require an Amazon EFS file system to encrypt file data at rest using AWS KMS

This control checks whether an Amazon Elastic File System (Amazon EFS) file system is configured to encrypt file data using AWS KMS.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EFS::FileSystem`
- **CloudFormation guard rule:** CT.ELASTICFILESYSTEM.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICFILESYSTEM.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICFILESYSTEM.PR.1 example templates

Explanation

For an added layer of security for your sensitive data in Amazon EFS, you should create encrypted file systems. Amazon EFS supports encryption for file systems at rest. You can enable encryption of data at rest when you create an Amazon EFS file system.

Usage considerations

- This control requires only the `Encrypted` property to be set to `true`, and it does not require the `KmsKeyId` property to be provided.
- If the `KmsKeyId` property is not provided, the default AWS KMS key for Amazon EFS, `/aws/elasticfilesystem`, is used to protect the encrypted file system.

Remediation for rule failure

Set `Encrypted` to `true` and optionally set `KmsKeyId` to a valid AWS KMS key identifier.

The examples that follow show how to implement this remediation.

Amazon EFS File System - Example One

Amazon EFS file system configured with encryption enabled, by means of the default AWS KMS key for Amazon EFS. The example is shown in JSON and in YAML.

JSON example

```
{
    "EFSFileSystem": {
        "Type": "AWS::EFS::FileSystem",
        "Properties": {
            "Encrypted": true
        }
    }
}
```

YAML example

```
EFSFileSystem:
  Type: AWS::EFS::FileSystem
```

```
  Properties:
    Encrypted: true
```

The examples that follow show how to implement this remediation.

Amazon EFS File System - Example Two

Amazon EFS file system configured with encryption enabled, by means of a customer-managed AWS KMS key. The example is shown in JSON and in YAML.

JSON example

```
{
    "EFSFileSystem": {
        "Type": "AWS::EFS::FileSystem",
        "Properties": {
            "Encrypted": true,
            "KmsKeyId": {
                "Ref": "KMSKey"
            }
        }
    }
}
```

YAML example

```
EFSFileSystem:
  Type: AWS::EFS::FileSystem
  Properties:
    Encrypted: true
    KmsKeyId: !Ref 'KMSKey'
```

CT.ELASTICFILESYSTEM.PR.1 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
# Rule Identifier:
#   efs_encrypted_check
#
# Description:
#   This control checks whether an Amazon Elastic File System (Amazon EFS) file system
 is configured to encrypt file data using AWS KMS.
#
# Reports on:
#   AWS::EFS::FileSystem
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EFS file system resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'Encrypted' is not present
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'Encrypted' is present and set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'Encrypted' is present and set to bool(true)
#      Then: PASS

#
# Constants
#
let RESOURCE_TYPE = "AWS::EFS::FileSystem"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let efs_file_systems = Resources.*[ Type == %RESOURCE_TYPE ]

#
# Primary Rules
#
rule efs_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                              %efs_file_systems not empty {
    check(%efs_file_systems.Properties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.1]: Require an Amazon EFS file system to encrypt
 file data at rest using AWS KMS
        [FIX]: Set 'Encrypted' to 'true' and optionally set 'KmsKeyId' to a valid AWS
 KMS key identifier.
        >>
}

rule efs_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT, %RESOURCE_TYPE) {
    check(%INPUT_DOCUMENT.%RESOURCE_TYPE.resourceProperties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.1]: Require an Amazon EFS file system to encrypt
 file data at rest using AWS KMS
        [FIX]: Set 'Encrypted' to 'true' and optionally set 'KmsKeyId' to a valid AWS
 KMS key identifier.
        >>
}

#
# Parameterized Rules
#
rule check(efs_file_systems) {
    %efs_file_systems {
        # Scenario 2
        Encrypted exists
        # Scenario 3 and 4
        Encrypted == true
    }
}

#
# Utility Rules
```

```
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICFILESYSTEM.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      Encrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      Encrypted: false
```

[CT.ELASTICFILESYSTEM.PR.2] Require an Amazon EFS volume to have an automated backup plan

This control checks whether your Amazon Elastic File System (Amazon EFS) file system has been configured with automatic backups through AWS Backup.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EFS::FileSystem`
- **CloudFormation guard rule:** CT.ELASTICFILESYSTEM.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICFILESYSTEM.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICFILESYSTEM.PR.2 example templates

Explanation

Including Amazon EFS file systems in the backup plans helps you to protect your data from deletion and data loss.

Remediation for rule failure

Enable automatic backups by setting `BackupPolicy.Status` to `ENABLED`.

The examples that follow show how to implement this remediation.

Amazon EFS File System - Example

Amazon EFS file system configured with automatic backups enabled. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "EFSFileSystem": {
        "Type": "AWS::EFS::FileSystem",
        "Properties": {
            "BackupPolicy": {
                "Status": "ENABLED"
            }
        }
    }
}
```

YAML example

```
EFSFileSystem:
  Type: AWS::EFS::FileSystem
  Properties:
    BackupPolicy:
      Status: ENABLED
```

CT.ELASTICFILESYSTEM.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   efs_automatic_backups_enabled_check
#
# Description:
#   This control checks whether your Amazon Elastic File System (Amazon EFS) file
 system has been configured with automatic backups through AWS Backup.
#
# Reports on:
#   AWS::EFS::FileSystem
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EFS file system resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'BackupPolicy' is not present
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'BackupPolicy' is present and 'Status' is set to 'DISABLED'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS file system resource
#       And: 'BackupPolicy' is present and 'Status' is set to 'ENABLED'
#      Then: PASS

#
# Constants
#
let RESOURCE_TYPE = "AWS::EFS::FileSystem"
let INPUT_DOCUMENT = this

#
# Assignments
#
let efs_file_systems = Resources.*[ Type == %RESOURCE_TYPE ]

#
# Primary Rules
#
rule efs_automatic_backups_enabled_check when is_cfn_template(this)
                                              %efs_file_systems not empty {
    check(%efs_file_systems.Properties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.2]: Require an Amazon EFS volume to have an
 automated backup plan
        [FIX]: Enable automatic backups by setting 'BackupPolicy.Status' to 'ENABLED'.
```

```
        >>

}

rule efs_automatic_backups_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RESOURCE_TYPE) {
    check(%INPUT_DOCUMENT.%RESOURCE_TYPE.resourceProperties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.2]: Require an Amazon EFS volume to have an
 automated backup plan
        [FIX]: Enable automatic backups by setting 'BackupPolicy.Status' to 'ENABLED'.
        >>
}

#
# Parameterized Rules
#
rule check(efs_file_systems) {
    %efs_file_systems {
        # Scenario 2
        BackupPolicy exists
        BackupPolicy is_struct
        BackupPolicy {
            # Scenario 3 and 4
            Status exists
            Status == "ENABLED"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICFILESYSTEM.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      BackupPolicy:
        Status: ENABLED
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      BackupPolicy:
        Status: DISABLED
```

[CT.ELASTICFILESYSTEM.PR.3] Require Amazon EFS access points to have a root directory

This control checks whether your Amazon Elastic File System (Amazon EFS) access points are configured to enforce a root directory.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::EFS::AccessPoint`
- **CloudFormation guard rule:** CT.ELASTICFILESYSTEM.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICFILESYSTEM.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICFILESYSTEM.PR.3 example templates

Explanation

When you enforce a root directory, the NFS client at the access point uses the root directory configured on the access point, instead of the file system's root directory. Enforcing a root directory for an access point helps restrict data access by ensuring that users of the access point can reach only the files of the specified subdirectory.

Remediation for rule failure

Provide a `RootDirectory.Path` configuration with a value for `Path` that does not equal `/`.

The examples that follow show how to implement this remediation.

Amazon EFS Access Point - Example

Amazon EFS access point configured with a root directory set to a specific subdirectory. The example is shown in JSON and in YAML.

JSON example

```
{
    "EFSAccessPoint": {
        "Type": "AWS::EFS::AccessPoint",
        "Properties": {
            "FileSystemId": {
                "Ref": "EFSFileSystem"
            },
            "RootDirectory": {
                "Path": "/dir1/child1"
            }
```

```
        }
    }
}
```

YAML example

```
EFSAccessPoint:
  Type: AWS::EFS::AccessPoint
  Properties:
    FileSystemId: !Ref 'EFSFileSystem'
    RootDirectory:
      Path: /dir1/child1
```

CT.ELASTICFILESYSTEM.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   efs_access_point_enforce_root_directory_check
#
# Description:
#   This control checks whether your Amazon Elastic File System (Amazon EFS) access
 points are configured to enforce a root directory.
#
# Reports on:
#   AWS::EFS::AccessPoint
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any Amazon EFS access point resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'RootDirectory' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'RootDirectory' has been provided
#       And: 'Path' within 'RootDirectory' has not been provided or has been provided
 with an empty string value
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'RootDirectory' has been provided
#       And: 'Path' within 'RootDirectory' been provided with a value of '/'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'RootDirectory' has been provided
#       And: 'Path' within 'RootDirectory' been provided with a non-empty string value
 not equal to '/'
#      Then: PASS

#
# Constants
#
let EFS_ACCESS_POINT_TYPE = "AWS::EFS::AccessPoint"
let INPUT_DOCUMENT = this

#
# Assignments
#
let efs_access_points = Resources.*[ Type == %EFS_ACCESS_POINT_TYPE ]

#
# Primary Rules
#
rule efs_access_point_enforce_root_directory_check when
 is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                        %efs_access_points not empty {
    check(%efs_access_points.Properties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.3]: Require Amazon EFS access points to have a root
 directory
        [FIX]: Provide a 'RootDirectory.Path' configuration with a value for 'Path'
 that does not equal '/'.
        >>
}

rule efs_access_point_enforce_root_directory_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EFS_ACCESS_POINT_TYPE) {
    check(%INPUT_DOCUMENT.%EFS_ACCESS_POINT_TYPE.resourceProperties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.3]: Require Amazon EFS access points to have a root
 directory
        [FIX]: Provide a 'RootDirectory.Path' configuration with a value for 'Path'
 that does not equal '/'.
        >>
}

#
# Parameterized Rules
#
rule check(efs_access_points) {
    %efs_access_points {
        # Scenario 2
        RootDirectory exists
        RootDirectory {
            # Scenario 3,4 and 5
            Path exists
            check_is_string_and_not_empty(Path)
            Path != "/"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.ELASTICFILESYSTEM.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  EFSAccessPoint:
    Type: AWS::EFS::AccessPoint
    Properties:
      FileSystemId:
        Ref: EFSFileSystem
      RootDirectory:
        Path: /dir1/child1
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties: {}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EFSAccessPoint:
    Type: AWS::EFS::AccessPoint
    Properties:
      FileSystemId:
        Ref: EFSFileSystem
      RootDirectory:
        Path: /
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties: {}
```

[CT.ELASTICFILESYSTEM.PR.4] Require Amazon EFS access points to enforce a user identity

This control checks whether your Amazon Elastic File System (Amazon EFS) access points are configured to enforce a user identity.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EFS::AccessPoint`
- **CloudFormation guard rule:** CT.ELASTICFILESYSTEM.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICFILESYSTEM.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICFILESYSTEM.PR.4 example templates

Explanation

Amazon EFS access points are application-specific entry points into an Amazon EFS file system that make it easier to manage application access to shared datasets. Access points can enforce a user identity, including the user's POSIX groups, for all file system requests that are made through the

access point. Access points also can enforce a different root directory for the file system, so that clients gain access only to data in the specified directory or its subdirectories.

Remediation for rule failure

Provide a `PosixUser` configuration with a POSIX user ID (`Uid`) and POSIX group ID (`Gid`).

The examples that follow show how to implement this remediation.

Amazon EFS Access Point - Example

Amazon EFS access point configured to enforce a user identity for all file system requests made through the access point. The example is shown in JSON and in YAML.

JSON example

```
{
    "EFSAccessPoint": {
        "Type": "AWS::EFS::AccessPoint",
        "Properties": {
            "FileSystemId": {
                "Ref": "EFSFileSystem"
            },
            "PosixUser": {
                "Uid": "111",
                "Gid": "222"
            }
        }
    }
}
```

YAML example

```
EFSAccessPoint:
  Type: AWS::EFS::AccessPoint
  Properties:
    FileSystemId: !Ref 'EFSFileSystem'
    PosixUser:
      Uid: '111'
      Gid: '222'
```

CT.ELASTICFILESYSTEM.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
# Rule Identifier:
#   efs_access_point_enforce_user_identity_check
#
# Description:
#   This control checks whether your Amazon Elastic File System (Amazon EFS) access
 points are configured to enforce a user identity.
#
# Reports on:
#   AWS::EFS::AccessPoint
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EFS access point resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'PosixUser' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'PosixUser' has been provided
#       And: 'Uid' within 'PosixUser' has not been provided or has been provided with
 an empty string value
#       And: 'Gid' within 'PosixUser' has not been provided or has been provided with
 an empty string value
```

```
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EFS access point resource
#       And: 'PosixUser' has been provided
#       And: 'Uid' within 'PosixUser' has been provided with a non-empty string value
#       And: 'Gid' within 'PosixUser' has been provided with a non-empty string value
#      Then: PASS

#
# Constants
#
let EFS_ACCESS_POINT_TYPE = "AWS::EFS::AccessPoint"
let INPUT_DOCUMENT = this

#
# Assignments
#
let efs_access_points = Resources.*[ Type == %EFS_ACCESS_POINT_TYPE ]

#
# Primary Rules
#
rule efs_access_point_enforce_user_identity_check when is_cfn_template(this)
                                                       %efs_access_points not empty {
    check(%efs_access_points.Properties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.4]: Require Amazon EFS access points to enforce a
 user identity
        [FIX]: Provide a 'PosixUser' configuration with a POSIX user ID ('Uid') and
 POSIX group ID ('Gid').
        >>
}

rule efs_access_point_enforce_user_identity_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EFS_ACCESS_POINT_TYPE) {
    check(%INPUT_DOCUMENT.%EFS_ACCESS_POINT_TYPE.resourceProperties)
        <<
        [CT.ELASTICFILESYSYSTEM.PR.4]: Require Amazon EFS access points to enforce a
 user identity
        [FIX]: Provide a 'PosixUser' configuration with a POSIX user ID ('Uid') and
 POSIX group ID ('Gid').
        >>
}
```

```
#
# Parameterized Rules
#
rule check(efs_access_points) {
    %efs_access_points {
        # Scenario 2
        PosixUser exists
        PosixUser {
            # Scenario 3 and 4
            Uid exists
            check_is_string_and_not_empty(Uid)
            Gid exists
            check_is_string_and_not_empty(Gid)
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.ELASTICFILESYSTEM.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  EFSAccessPoint:
    Type: AWS::EFS::AccessPoint
    Properties:
      FileSystemId:
        Ref: EFSFileSystem
      PosixUser:
        Uid: '111'
        Gid: '222'
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties: {}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  EFSAccessPoint:
    Type: AWS::EFS::AccessPoint
    Properties:
      FileSystemId:
        Ref: EFSFileSystem
  EFSFileSystem:
    Type: AWS::EFS::FileSystem
    Properties: {}
```

Amazon Elastic Kubernetes Service (EKS) controls

Topics

- [CT.EKS.PR.1] Require an Amazon EKS cluster to be configured with public access disabled to the cluster Kubernetes API server endpoint
- [CT.EKS.PR.2] Require an Amazon EKS cluster to be configured with secret encryption using AWS Key Management Service (KMS) keys

[CT.EKS.PR.1] Require an Amazon EKS cluster to be configured with public access disabled to the cluster Kubernetes API server endpoint

This control checks whether an Amazon Elastic Kubernetes Service (EKS) cluster endpoint disallows public access to the cluster Kubernetes API server endpoint.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EKS::Cluster`
- **CloudFormation guard rule:** CT.EKS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EKS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.EKS.PR.1 example templates

Explanation

When you create a new cluster, Amazon Elastic Kubernetes Service (EKS) creates an endpoint for the managed Kubernetes API server, which you can use to communicate with your cluster using Kubernetes management tools, such as `kubectl`. By default, this API server endpoint is public to the internet, and access to the API server is secured using a combination of AWS Identity and Access Management (IAM) along with native Kubernetes Role-Based Access Control (RBAC). Enabling private access to the Kubernetes API server ensures that all communication between your nodes and the API server stays within your VPC. You can limit the IP addresses that have access to your API server from the internet, or you can completely disallow internet access to the API server.

Remediation for rule failure

Set the value of the `EndpointPublicAccess` parameter to false and the value of the `EndpointPrivateAccess` parameter to true.

The examples that follow show how to implement this remediation.

Amazon EKS Cluster - Example

An Amazon EKS cluster configured with public access disabled to the cluster's Kubernetes API server endpoint. The example is shown in JSON and in YAML.

JSON example

```
{
    "EKSCluster": {
        "Type": "AWS::EKS::Cluster",
        "Properties": {
            "RoleArn": {
                "Fn::GetAtt": "EKSClusterRole.Arn"
            },
            "ResourcesVpcConfig": {
                "SubnetIds": [
                    {
                        "Ref": "SubnetOne"
                    },
                    {
                        "Ref": "SubnetTwo"
                    }
                ],
                "EndpointPublicAccess": false,
                "EndpointPrivateAccess": true
            }
        }
    }
}
```

YAML example

```
EKSCluster:
```

```
  Type: AWS::EKS::Cluster
  Properties:
    RoleArn: !GetAtt 'EKSClusterRole.Arn'
    ResourcesVpcConfig:
      SubnetIds:
        - !Ref 'SubnetOne'
        - !Ref 'SubnetTwo'
      EndpointPublicAccess: false
      EndpointPrivateAccess: true
```

CT.EKS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   eks_endpoint_no_public_access_check
#
# Description:
#   This control checks whether an Amazon Elastic Kubernetes Service (EKS) cluster
 endpoint disallows public access to the cluster Kubernetes API server endpoint.
#
# Reports on:
#   AWS::EKS::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EKS cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
```

```
#       And: 'EndpointPublicAccess' in 'ResourcesVpcConfig' has not been provided
#       And: 'EndpointPrivateAccess' in 'ResourcesVpcConfig' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EndpointPrivateAccess' in 'ResourcesVpcConfig' has not been provided
#       And: 'EndpointPublicAccess' in 'ResourcesVpcConfig' has not been provided or
 has been provided and set to a
#            value other than bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EndpointPublicAccess' in 'ResourcesVpcConfig' has not been provided or
 has been provided and set to a
#            value other than bool(false)
#       And: 'EndpointPrivateAccess' in 'ResourcesVpcConfig' has been provided and set
 a value other than bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EndpointPublicAccess' in 'ResourcesVpcConfig' has been provided and set
 to bool(false)
#       And: 'EndpointPrivateAccess' in 'ResourcesVpcConfig' has been provided and set
 to bool(true)
#      Then: PASS

#
# Constants
#
let EKS_CLUSTER_TYPE = "AWS::EKS::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let eks_clusters = Resources.*[ Type == %EKS_CLUSTER_TYPE ]

#
# Primary Rules
#
rule eks_endpoint_no_public_access_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                %eks_clusters not empty {
    check(%eks_clusters.Properties)
        <<
        [CT.EKS.PR.1]: Require an Amazon EKS cluster to be configured with public
 access disabled to the cluster Kubernetes API server endpoint.
        [FIX]: Set the value of the 'EndpointPublicAccess' parameter to false and the
 value of the 'EndpointPrivateAccess' parameter to true.
        >>
}

rule eks_endpoint_no_public_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EKS_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%EKS_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.EKS.PR.1]: Require an Amazon EKS cluster to be configured with public
 access disabled to the cluster Kubernetes API server endpoint.
        [FIX]: Set the value of the 'EndpointPublicAccess' parameter to false and the
 value of the 'EndpointPrivateAccess' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(eks_cluster) {
    %eks_cluster {
        ResourcesVpcConfig exists
        ResourcesVpcConfig is_struct
        ResourcesVpcConfig {
            # Scenarios 2, 3, 4 and 5
            EndpointPublicAccess exists
            EndpointPublicAccess == false

            EndpointPrivateAccess exists
            EndpointPrivateAccess == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
```

```
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EKS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
```

```
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  EKSClusterRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: eks.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
  EKSCluster:
    Type: AWS::EKS::Cluster
    Properties:
      RoleArn:
        Fn::GetAtt: EKSClusterRole.Arn
      ResourcesVpcConfig:
        SubnetIds:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        EndpointPublicAccess: false
        EndpointPrivateAccess: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  EKSClusterRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: eks.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
  EKSCluster:
    Type: AWS::EKS::Cluster
    Properties:
      RoleArn:
        Fn::GetAtt: EKSClusterRole.Arn
      ResourcesVpcConfig:
        SubnetIds:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        EndpointPublicAccess: true
        EndpointPrivateAccess: false
```

[CT.EKS.PR.2] Require an Amazon EKS cluster to be configured with secret encryption using AWS Key Management Service (KMS) keys

This control checks whether Amazon Elastic Kubernetes Service (Amazon EKS) clusters are configured to use Kubernetes secrets encrypted with AWS Key Management Service (KMS) keys.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EKS::Cluster`
- **CloudFormation guard rule:** CT.EKS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EKS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EKS.PR.2 example templates

Explanation

Kubernetes secrets store sensitive information, such as user certificates, passwords, or API keys. Encrypting Kubernetes secrets at rest bolsters the security of your EKS clusters.

Usage considerations

- For a cluster that uses KMS Envelope Encryption, `kms:CreateGrant` permissions are required. The condition `kms:GrantIsForAWSResource` is not supported for the CreateCluster action, and this condition should not be given in KMS policies to control `kms:CreateGrant` permissions for users performing CreateCluster operations.

Remediation for rule failure

Provide an `EncryptionConfig` configuration with a list of `Resources` that contains `secrets` and a `Provider` configuration containing a `KeyArn`.

The examples that follow show how to implement this remediation.

Amazon EKS cluster - Example

Amazon EKS cluster configured to have Kubernetes secrets encrypted using Amazon Elastic Kubernetes Service (KMS) keys. The example is shown in JSON and in YAML.

JSON example

```
{
    "EKSCluster": {
        "Type": "AWS::EKS::Cluster",
        "Properties": {
            "RoleArn": {
                "Fn::GetAtt": [
                    "EKSClusterRole",
                    "Arn"
                ]
            },
            "ResourcesVpcConfig": {
                "SubnetIds": [
                    {
                        "Ref": "SubnetOne"
                    },
                    {
                        "Ref": "SubnetTwo"
                    }
                ],
                "EndpointPublicAccess": false,
                "EndpointPrivateAccess": true
            },
            "Logging": {
                "ClusterLogging": {
                    "EnabledTypes": [
                        {
                            "Type": "api"
                        },
```

```
                        {
                            "Type": "audit"
                        },
                        {
                            "Type": "authenticator"
                        },
                        {
                            "Type": "controllerManager"
                        },
                        {
                            "Type": "scheduler"
                        }
                    ]
                }
            },
            "EncryptionConfig": [
                {
                    "Resources": [
                        "secrets"
                    ],
                    "Provider": {
                        "KeyArn": {
                            "Fn::GetAtt": [
                                "KMSKey",
                                "Arn"
                            ]
                        }
                    }
                }
            ]
        }
    }
}
```

YAML example

```
EKSCluster:
  Type: AWS::EKS::Cluster
  Properties:
    RoleArn: !GetAtt 'EKSClusterRole.Arn'
    ResourcesVpcConfig:
```

```
      SubnetIds:
        - !Ref 'SubnetOne'
        - !Ref 'SubnetTwo'
      EndpointPublicAccess: false
      EndpointPrivateAccess: true
    Logging:
      ClusterLogging:
        EnabledTypes:
          - Type: api
          - Type: audit
          - Type: authenticator
          - Type: controllerManager
          - Type: scheduler
    EncryptionConfig:
      - Resources:
          - secrets
        Provider:
          KeyArn: !GetAtt 'KMSKey.Arn'
```

CT.EKS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   eks_secrets_encrypted_check
#
# Description:
#   This control checks whether Amazon Elastic Kubernetes Service (Amazon EKS) clusters
 are configured to use Kubernetes secrets encrypted with AWS Key Management Service
 (KMS) keys.
#
# Reports on:
#   AWS::EKS::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon EKS cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EncryptionConfig' has not been provided or provided as an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EncryptionConfig' has been provided as a non-empty list
#       And: There are no entries in 'EncryptionConfig' where 'Resources' has been
 provided
#            as a non-empty list with at least one value equal to 'secrets'
#       And: For the same entry in 'EncryptionConfig', where 'KeyArn' in 'Provider' has
 been
#            provided as a non-empty string or valid local reference to a KMS key or
 key alias
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon EKS cluster resource
#       And: 'EncryptionConfig' has been provided as a non-empty list
#       And: For at least one entry in 'EncryptionConfig', 'Resources' has been
 provided as a
#            non-empty list with at least one value equal to 'secrets'
#       And: For the same entry in 'EncryptionConfig', 'KeyArn' in 'Provider' has been
 provided
#            as a non-empty string or valid local reference to a KMS key or key alias
#      Then: PASS

#
# Constants
#
let EKS_CLUSTER_TYPE = "AWS::EKS::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
```

```
#
let eks_clusters = Resources.*[ Type == %EKS_CLUSTER_TYPE ]

#
# Primary Rules
#
rule eks_secrets_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                      %eks_clusters not empty {
    check(%eks_clusters.Properties)
        <<
        [CT.EKS.PR.2]: Require an Amazon EKS cluster to be configured with secret
 encryption using AWS Key Management Service (KMS) keys
        [FIX]: Provide an 'EncryptionConfig' configuration with a list of 'Resources'
 that contains 'secrets' and a 'Provider' configuration containing a 'KeyArn'.
        >>
}

rule eks_secrets_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT, %EKS_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%EKS_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.EKS.PR.2]: Require an Amazon EKS cluster to be configured with secret
 encryption using AWS Key Management Service (KMS) keys
        [FIX]: Provide an 'EncryptionConfig' configuration with a list of 'Resources'
 that contains 'secrets' and a 'Provider' configuration containing a 'KeyArn'.
        >>
}

#
# Parameterized Rules
#
rule check(eks_cluster) {
    %eks_cluster {
        # Scenario 2
        EncryptionConfig exists
        EncryptionConfig is_list
        EncryptionConfig not empty

        # Scenario 3 and 4
        some EncryptionConfig[*] {
            Resources exists
            Resources is_list
            Resources not empty
            some Resources[*] == "secrets"
```

```
            Provider exists
            Provider is_struct
            Provider {
                KeyArn exists
                check_is_string_and_not_empty(KeyArn) or
                check_local_references(%INPUT_DOCUMENT, KeyArn, "AWS::KMS::Key") or
                check_local_references(%INPUT_DOCUMENT, KeyArn, "AWS::KMS::Alias")
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}
```

```
rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.EKS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
```

```
      - 1
      - Fn::GetAZs: ''
  EKSClusterRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: eks.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      PendingWindowInDays: 7
      KeyPolicy:
        Version: 2012-10-17
        Id: example-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeySpec: SYMMETRIC_DEFAULT
  EKSCluster:
    Type: AWS::EKS::Cluster
    Properties:
      RoleArn:
        Fn::GetAtt:
        - EKSClusterRole
        - Arn
      ResourcesVpcConfig:
        SubnetIds:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        EndpointPublicAccess: false
        EndpointPrivateAccess: true
```

```
      Logging:
        ClusterLogging:
          EnabledTypes:
          - Type: api
          - Type: audit
          - Type: authenticator
          - Type: controllerManager
          - Type: scheduler
      EncryptionConfig:
      - Resources:
        - secrets
        Provider:
          KeyArn:
            Fn::GetAtt: [KMSKey, Arn]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
```

```
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  EKSClusterRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: eks.amazonaws.com
          Action: sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
  EKSCluster:
    Type: AWS::EKS::Cluster
    Properties:
      RoleArn:
        Fn::GetAtt:
        - EKSClusterRole
        - Arn
      ResourcesVpcConfig:
        SubnetIds:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        EndpointPublicAccess: false
        EndpointPrivateAccess: true
      Logging:
        ClusterLogging:
          EnabledTypes:
          - Type: api
          - Type: audit
          - Type: authenticator
          - Type: controllerManager
          - Type: scheduler
```

Elastic Load Balancing controls

Topics

- [CT.ELASTICLOADBALANCING.PR.1] Require any application load balancer listener default actions to redirect all HTTP requests to HTTPS
- [CT.ELASTICLOADBALANCING.PR.2] Require any Amazon ELB application or network load balancer to have an AWS Certificate Manager certificate
- [CT.ELASTICLOADBALANCING.PR.3] Require any application load balancer to have defensive or strictest desync mitigation mode activated
- [CT.ELASTICLOADBALANCING.PR.4] Require that any application load balancer must be configured to drop HTTP headers
- [CT.ELASTICLOADBALANCING.PR.5] Require that application load balancer deletion protection is activated
- [CT.ELASTICLOADBALANCING.PR.6] Require that application and network load balancer access logging is activated
- [CT.ELASTICLOADBALANCING.PR.7] Require any classic load balancer to have multiple Availability Zones configured
- [CT.ELASTICLOADBALANCING.PR.8] Require any classic load balancer SSL/HTTPS listener to have a certificate provided by AWS Certificate Manager
- [CT.ELASTICLOADBALANCING.PR.9] Require that an AWS ELB Application or Classic Load Balancer listener is configured with HTTPS or TLS termination
- [CT.ELASTICLOADBALANCING.PR.10] Require an ELB application or classic load balancer to have logging activated
- [CT.ELASTICLOADBALANCING.PR.11] Require any ELB classic load balancer to have connection draining activated
- [CT.ELASTICLOADBALANCING.PR.12] Require any ELB classic load balancer SSL/HTTPS listener to have a predefined security policy with a strong configuration
- [CT.ELASTICLOADBALANCING.PR.13] Require any ELB classic load balancer to have cross-zone load balancing activated
- [CT.ELASTICLOADBALANCING.PR.14] Require a Network Load Balancer to have cross-zone load balancing activated
- [CT.ELASTICLOADBALANCING.PR.15] Require that an Elastic Load Balancing v2 target group does not explicitly disable cross-zone load balancing

[CT.ELASTICLOADBALANCING.PR.1] Require any application load balancer listener default actions to redirect all HTTP requests to HTTPS

This control checks whether HTTP to HTTPS redirection is configured as a default action on HTTP listeners of Application Load Balancers.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::Listener`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.1 example templates

Explanation

Before you start to use your Application Load Balancer, you must add one or more listeners. A listener is a process that uses the configured protocol and port to check for connection requests. Listeners support HTTP and HTTPS protocols. You can use an HTTPS listener to offload the work of encryption and decryption to your Application Load Balancer. You should utilize redirect actions with Application Load Balancer to redirect any client HTTP request to an HTTPS request on port 443, to enforce encryption in transit.

Usage considerations

- This control evaluates only the default actions on Application Load Balancer listeners.

Remediation for rule failure

Configure a default HTTPS redirect action on Application Load Balancer HTTP listeners.

The examples that follow show how to implement this remediation.

Application Load Balancer Listener - Example

Application load balancer listener configured with a default action that redirects HTTP requests on port 80 to HTTPS requests on port 443, retaining the original host name, path, and query string. The example is shown in JSON and in YAML.

JSON example

```
{
    "Listener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "Properties": {
            "LoadBalancerArn": {
                "Ref": "ApplicationLoadBalancer"
            },
            "Port": 80,
            "Protocol": "HTTP",
            "DefaultActions": [
                {
                    "Type": "redirect",
                    "RedirectConfig": {
                        "Protocol": "HTTPS",
                        "Port": 443,
                        "Host": "#{host}",
                        "Path": "/#{path}",
                        "Query": "#{query}",
                        "StatusCode": "HTTP_301"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
Listener:
```

```
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    LoadBalancerArn: !Ref 'ApplicationLoadBalancer'
    Port: 80
    Protocol: HTTP
    DefaultActions:
      - Type: redirect
        RedirectConfig:
          Protocol: HTTPS
          Port: 443
          Host: '#{host}'
          Path: /#{path}
          Query: '#{query}'
          StatusCode: HTTP_301
```

CT.ELASTICLOADBALANCING.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   alb_http_to_https_redirection_check
#
# Description:
#   This control checks whether HTTP to HTTPS redirection is configured as a default
 action on HTTP listeners of Application Load Balancers.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::Listener
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any ElasticLoadBalancingV2 listener
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener
#       And: 'Protocol' is set to a value other than 'HTTP'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener
#       And: 'Protocol' is set to 'HTTP'
#       And: 'DefaultActions' is missing or is provided and an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener
#       And: 'Protocol' is set to 'HTTP'
#       And: 'DefaultActions' contains an action with 'Type' set to a value other than
 'redirect'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener
#       And: 'Protocol' is set to 'HTTP'
#       And: 'DefaultActions' contains an action with 'Type' set to a value of
 'redirect'
#       And: 'RedirectConfig.Protocol' is missing or set to a value other than 'HTTPS'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener
#       And: 'Protocol' is set to 'HTTP'
#       And: All 'DefaultActions' have an action with 'Type' set to a value of
 'redirect' and
#            'Protocol.RedirectConfig' set to the value 'HTTPS'
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE = "AWS::ElasticLoadBalancingV2::Listener"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let elb_v2_listeners = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE ]

#
# Primary Rules
#
rule alb_http_to_https_redirection_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %elb_v2_listeners not empty {
    check(%elb_v2_listeners.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.1]: Require any application load balancer listener
 default actions to redirect all HTTP requests to HTTPS
        [FIX]: Configure a default HTTPS redirect action on application load balancer
 HTTP listeners.
        >>
}

rule alb_http_to_https_redirection_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.1]: Require any application load balancer listener
 default actions to redirect all HTTP requests to HTTPS
        [FIX]: Configure a default HTTPS redirect action on application load balancer
 HTTP listeners.
        >>
}

#
# Parameterized Rules
#
rule check(elbv2_listener) {
    %elbv2_listener [
        # Scenario 2
        Protocol in [ "HTTP" ]
    ] {
        # Scenarios 3
        DefaultActions exists
        DefaultActions is_list
        DefaultActions not empty

        # Scenario 4 and 5
```

```
        DefaultActions[*] {
            Type == "redirect"

            RedirectConfig exists
            RedirectConfig is_struct
            RedirectConfig {
                Protocol exists
                Protocol == "HTTPS"
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
```

```
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
  Listener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn:
        Ref: ApplicationLoadBalancer
      Port: 80
      Protocol: HTTP
      DefaultActions:
      - Type: redirect
        RedirectConfig:
          Protocol: HTTPS
          Port: 443
          Host: "#{host}"
          Path: "/#{path}"
          Query: "#{query}"
          StatusCode: "HTTP_301"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
  Listener:
```

```
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn:
        Ref: ApplicationLoadBalancer
      Port: 80
      Protocol: HTTP
      DefaultActions:
      - Type: redirect
        RedirectConfig:
          Protocol: HTTP
          Port: 8080
          Host: "#{host}"
          Path: "/#{path}"
          Query: "#{query}"
          StatusCode: "HTTP_301"
```

[CT.ELASTICLOADBALANCING.PR.2] Require any Amazon ELB application or network load balancer to have an AWS Certificate Manager certificate

This control checks whether your Elastic Load Balancing (ELB) application and network load balancers use certificates provided by AWS Certificate Manager (ACM).

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::Listener`, `AWS::ElasticLoadBalancingV2::ListenerCertificate`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.2 example templates

Explanation

To create a certificate, use AWS Certificate Manager (ACM) or another tool that supports the SSL and TLS protocols, such as OpenSSL. AWS Control Tower recommends that you use AWS Certificate Manager to create or import certificates for your load balancer.

AWS Certificate Manager integrates with Amazon ELB application load balancers and network load balancers, so that you can deploy the certificate on your load balancer. We also recommend that you automatically renew these certificates.

Usage considerations

- This control applies only to HTTPS and TLS Amazon ELB listeners and ELB listener certificate resources that have one or more certificates configured.

Remediation for rule failure

Configure the `Certificates` property to use certificates provided by AWS Certificate Manager.

The examples that follow show how to implement this remediation.

Amazon ELB Listener - Example

Amazon ELB HTTPS listener configured with an AWS Certificate Manager SSL certificate. The example is shown in JSON and in YAML.

JSON example

```
{
    "ELBListener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "Properties": {
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": {
                        "Ref": "TargetGroup"
                    }
                }
```

```
            ],
            "LoadBalancerArn": {
                "Ref": "ApplicationLoadBalancer"
            },
            "Protocol": "HTTPS",
            "Certificates": [
                {
                    "CertificateArn": {
                        "Ref": "ACMCertificate"
                    }
                }
            ],
            "Port": 443
        }
    }
}
```

YAML example

```
ELBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref 'TargetGroup'
    LoadBalancerArn: !Ref 'ApplicationLoadBalancer'
    Protocol: HTTPS
    Certificates:
      - CertificateArn: !Ref 'ACMCertificate'
    Port: 443
```

The examples that follow show how to implement this remediation.

Amazon ELB Listener Certificate - Example

Amazon ELB listener certificate configured with an AWS Certificate Manager SSL certificate. The example is shown in JSON and in YAML.

JSON example

```
{
    "ELBListenerCertificate": {
        "Type": "AWS::ElasticLoadBalancingV2::ListenerCertificate",
        "Properties": {
            "ListenerArn": {
                "Ref": "Listener"
            },
            "Certificates": [
                {
                    "CertificateArn": {
                        "Ref": "ACMCertificate"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ELBListenerCertificate:
  Type: AWS::ElasticLoadBalancingV2::ListenerCertificate
  Properties:
    ListenerArn: !Ref 'Listener'
    Certificates:
      - CertificateArn: !Ref 'ACMCertificate'
```

CT.ELASTICLOADBALANCING.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   elbv2_acm_certificate_required_check
#
```

```
# Description:
#   This control checks whether your Elastic Load Balancing (ELB) application and
 network load balancers use certificates provided by AWS Certificate Manager (ACM).
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::Listener,
 AWS::ElasticLoadBalancingV2::ListenerCertificate
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElasticLoadBalancingV2 listener or
 listener certificate resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener resource
#       And: 'Protocol' is set to a value other than 'HTTPS' or 'TLS'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener certificate
 resource
#       And: 'Certificates' has not been provided or has been provided as an empty list
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener resource
#       And: 'Protocol' is set to 'HTTPS' or 'TLS'
#       And: 'Certificates' has not been provided or has been provided as an empty list
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener resource
#       And: 'Protocol' is set to 'HTTPS' or 'TLS'
#       And: One or more items in 'Certificates' do not match an ACM certificate ARN
#      Then: FAIL
#   Scenario: 6
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener certificate
 resource
#       And: One or more items in 'Certificates' do not match an ACM certificate ARN
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener resource
#       And: 'Protocol' is set to 'HTTPS' or 'TLS'
#       And: All items in 'Certificates' match an ACM certificate ARN
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 listener certificate
 resource
#       And: All items in 'Certificates' match an ACM certificate ARN
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE = "AWS::ElasticLoadBalancingV2::Listener"
let ELASTIC_LOAD_BALANCER_V2_CERTIFICATE_TYPE =
 "AWS::ElasticLoadBalancingV2::ListenerCertificate"
let ACM_CERTIFICATE_ARN_PATTERN = /arn:aws[a-z0-9\-]*:acm:[a-z0-9\-]+:
\d{12}:certificate\/[\w\-]{1,64}/
let INPUT_DOCUMENT = this

#
# Assignments
#
let elb_v2_listeners = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE ]
let elb_v2_certificates = Resources.*[ Type ==
 %ELASTIC_LOAD_BALANCER_V2_CERTIFICATE_TYPE ]

#
# Primary Rules
#
rule elbv2_acm_certificate_required_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %elb_v2_listeners not empty {
    check_listener(%elb_v2_listeners.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.2]: Require any Amazon ELB application or network
 load balancer to have an AWS Certificate Manager certificate
```

```
            [FIX]: Configure the 'Certificates' property to use certificates provided
 by AWS Certificate Manager.
        >>
}

rule elbv2_acm_certificate_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE) {
    check_listener(%INPUT_DOCUMENT.
%ELASTIC_LOAD_BALANCER_V2_LISTENER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.2]: Require any Amazon ELB application or network
 load balancer to have an AWS Certificate Manager certificate
            [FIX]: Configure the 'Certificates' property to use certificates provided
 by AWS Certificate Manager.
        >>
}

rule elbv2_acm_certificate_required_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %elb_v2_certificates not empty {
    check_elbv2_listener_certificate(%elb_v2_certificates.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.2]: Require any Amazon ELB application or network
 load balancer to have an AWS Certificate Manager certificate
            [FIX]: Configure the 'Certificates' property to use certificates provided
 by AWS Certificate Manager.
        >>
}

rule elbv2_acm_certificate_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_CERTIFICATE_TYPE) {
    check_elbv2_listener_certificate(%INPUT_DOCUMENT.
%ELASTIC_LOAD_BALANCER_V2_CERTIFICATE_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.2]: Require any Amazon ELB application or network
 load balancer to have an AWS Certificate Manager certificate
            [FIX]: Configure the 'Certificates' property to use certificates provided
 by AWS Certificate Manager.
        >>
}

#
# Parameterized Rules
#
rule check_listener(elbv2_listener) {
```

```
    %elbv2_listener[
        # Scenario 2
        Protocol in ["HTTPS", "TLS"]
    ] {
        # Scenarios 3 and 5
        Certificates exists
        Certificates is_list
        Certificates not empty
        Certificates[*] {
            CertificateArn exists
            check_is_acm_certificate(CertificateArn)
        }
    }
}

rule check_elbv2_listener_certificate(listener_certificate) {
    %listener_certificate[
        Certificates exists
        Certificates is_list
        Certificates not empty
    ] {
        # Scenarios 4 and 6
        Certificates[*] {
            CertificateArn exists
            check_is_acm_certificate(CertificateArn)
        }
    }
}

rule check_is_acm_certificate(certificate) {
    %certificate {
        this == %ACM_CERTIFICATE_ARN_PATTERN or
        check_local_references(%INPUT_DOCUMENT, this,
 "AWS::CertificateManager::Certificate")
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICLOADBALANCING.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
```

```
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Protocol: HTTP
      Port: 80
      VpcId:
        Ref: VPC
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
  Listener:
```

```
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: TargetGroup
      LoadBalancerArn:
        Ref: ApplicationLoadBalancer
      Protocol: HTTPS
      Certificates:
      - CertificateArn:
          Ref: ACMCertificate
      Port: 443
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Protocol: HTTP
      Port: 80
      VpcId:
        Ref: VPC
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
  Listener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: TargetGroup
      LoadBalancerArn:
        Ref: ApplicationLoadBalancer
      Protocol: HTTPS
      Certificates:
      - CertificateArn: arn:aws:iam::123456789012:server-certificate/example-
certificate
      Port: 443
```

[CT.ELASTICLOADBALANCING.PR.3] Require any application load balancer to have defensive or strictest desync mitigation mode activated

This control checks to ensure that an Application Load Balancer is configured with `defensive` or `strictest` desync mitigation mode.

- **Control objective:** Protect data integrity

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.3 example templates

Explanation

HTTP desynchronization (desync) issues can lead to request smuggling and make applications vulnerable to request queue or cache poisoning. In turn, these vulnerabilities can lead to credential stuffing or execution of unauthorized commands. When configured with defensive or strictest desync mitigation mode, Application Load Balancers can protect your application from security issues that may be caused by HTTP desync.

Remediation for rule failure

Omit the load balancer attribute `routing.http.desync_mitigation_mode` or set the attribute to one of `defensive` or `strictest`.

The examples that follow show how to implement this remediation.

Application Load Balancer - Example

Application Load Balancer configured with `defensive` desync mitigation mode, by means of AWS CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApplicationLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
```

```
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "Type": "application"
        }
    }
}
```

YAML example

```
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    Type: application
```

The examples that follow show how to implement this remediation.

Application Load Balancer - Example

Application Load Balancer configured with `strictest` desync mitigation mode, by meand of the `routing.http.desync_mitigation_mode` load balancer attribute. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApplicationLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "Type": "application",
            "LoadBalancerAttributes": [
                {
                    "Key": "routing.http.desync_mitigation_mode",
                    "Value": "strictest"
                }
            ]
        }
    }
}
```

YAML example

```
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    Type: application
    LoadBalancerAttributes:
      - Key: routing.http.desync_mitigation_mode
        Value: strictest
```

CT.ELASTICLOADBALANCING.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   alb_desync_mode_check
#
# Description:
#   This control checks to ensure that an Application Load Balancer is configured with
 'defensive' or 'strictest' desync mitigation mode.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ELBv2 load balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' is set to a value other than 'application'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: The 'LoadBalancerAttribute' 'routing.http.desync_mitigation_mode' has been
 provided
#            and is not one of 'defensive' or 'strictest'
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have not been specified on the ELBv2 load
 balancer resource or specified
#            as an empty list
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: 'routing.http.desync_mitigation_mode' has not been provided as a
 'LoadBalancerAttribute'
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: The 'LoadBalancerAttribute' 'routing.http.desync_mitigation_mode' has been
 provided
#            and is one of 'defensive' or 'strictest'
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_TYPE = "AWS::ElasticLoadBalancingV2::LoadBalancer"
let ALLOWED_DESYNC_MODES = [ "defensive", "strictest" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_TYPE ]

#
# Primary Rules
#
rule alb_desync_mode_check when is_cfn_template(%INPUT_DOCUMENT)
                               %elastic_load_balancers not empty {
```

```
    check(%elastic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.3]: Require any application load balancer to have
 defensive or strictest desync mitigation mode activated
        [FIX]: Omit the load balancer attribute 'routing.http.desync_mitigation_mode'
 or set the attribute to one of 'defensive' or 'strictest'.
        >>

}

rule alb_desync_mode_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.3]: Require any application load balancer to have
 defensive or strictest desync mitigation mode activated
        [FIX]: Omit the load balancer attribute 'routing.http.desync_mitigation_mode'
 or set the attribute to one of 'defensive' or 'strictest'.
        >>
}

#
# Parameterized Rules
#
rule check(elastic_load_balancer) {
    %elastic_load_balancer[
        # Scenario 2
        Type not exists or
        Type == "application"
    ] {
        # Scenario 4
        LoadBalancerAttributes not exists or
        check_application_load_balancer_attributes(this)
    }
}

rule check_application_load_balancer_attributes(application_load_balancer) {
    %application_load_balancer {
        LoadBalancerAttributes is_list
        LoadBalancerAttributes[
            # Scenario 5
            Key exists
```

```
            Key == "routing.http.desync_mitigation_mode"
        ] {
            # Scenarios 3 and 6
            Value exists
            Value in %ALLOWED_DESYNC_MODES
        }

    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      Type: application
      LoadBalancerAttributes:
      - Key: routing.http.desync_mitigation_mode
        Value: strictest
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
```

```
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      Type: application
      LoadBalancerAttributes:
      - Key: routing.http.desync_mitigation_mode
        Value: monitor
```

[CT.ELASTICLOADBALANCING.PR.4] Require that any application load balancer must be configured to drop HTTP headers

This control checks whether Application Load Balancers are configured to drop non-valid HTTP headers.

- **Control objective:** Protect configurations

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.4 example templates

Explanation

By default, Application Load Balancers are not configured to drop non-valid HTTP header values. Removing these header values prevents HTTP desync attacks.

Remediation for rule failure

Set the load balancer attribute `routing.http.drop_invalid_header_fields.enabled` to `true`.

The examples that follow show how to implement this remediation.

Application Load Balancer - Example

Application Load Balancer configured to drop non-valid HTTP headers. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApplicationLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Type": "application",
            "Subnets": [
                {
```

```
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "LoadBalancerAttributes": [
                {
                    "Key": "routing.http.drop_invalid_header_fields.enabled",
                    "Value": "true"
                }
            ]
        }
    }
}
```

YAML example

```
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Type: application
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    LoadBalancerAttributes:
      - Key: routing.http.drop_invalid_header_fields.enabled
        Value: 'true'
```

CT.ELASTICLOADBALANCING.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
```

```
# Rule Identifier:
#   alb_http_drop_invalid_header_enabled_check
#
# Description:
#   This control checks whether Application Load Balancers are configured to drop non-
valid HTTP headers.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ELBv2 load balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' is set to a value other than 'application', or is not set
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have not been specified on the ELBv2 load
 balancer resource
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: 'routing.http.drop_invalid_header_fields.enabled' has not been provided as
 a 'LoadBalancerAttribute'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: The 'LoadBalancerAttribute'
 'routing.http.drop_invalid_header_fields.enabled' has been provided
#            and is set to bool(false) or string(false)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 load balancer resource
#       And: 'Type' has not been provided or has been provided and set to 'application'
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 load balancer
 resource
#       And: The 'LoadBalancerAttribute'
 'routing.http.drop_invalid_header_fields.enabled' has been provided and
#            is set to bool(true) or string(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_TYPE = "AWS::ElasticLoadBalancingV2::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_TYPE ]

#
# Primary Rules
#
rule alb_http_drop_invalid_header_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %elastic_load_balancers not empty
 {

    check(%elastic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.4]: Require that any application load balancer must
 be configured to drop HTTP headers
        [FIX]: Set the load balancer attribute
 'routing.http.drop_invalid_header_fields.enabled' to 'true'.
        >>
```

```
}

rule alb_http_drop_invalid_header_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.4]: Require that any application load balancer must
 be configured to drop HTTP headers
        [FIX]: Set the load balancer attribute
 'routing.http.drop_invalid_header_fields.enabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(elastic_load_balancer) {
    %elastic_load_balancer[ Type not exists or Type == "application" ] {
        # Scenario 2
        LoadBalancerAttributes exists
        LoadBalancerAttributes is_list
        LoadBalancerAttributes not empty

        # Scenario 3, 4 and 5
        some LoadBalancerAttributes[*] {
            Key exists
            Value exists

            Key == "routing.http.drop_invalid_header_fields.enabled"
            Value in [ true, "true" ]
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
```

```
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: application
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      LoadBalancerAttributes:
      - Key: routing.http.drop_invalid_header_fields.enabled
        Value: "true"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
```

```
      - 1
      - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: application
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      LoadBalancerAttributes:
      - Key: routing.http.drop_invalid_header_fields.enabled
        Value: "false"
```

[CT.ELASTICLOADBALANCING.PR.5] Require that application load balancer deletion protection is activated

Checks whether Elastic Load Balancing (ELB) has deletion protection activated.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.5 example templates

Explanation

Activate deletion protection to protect your Application Load Balancer from deletion.

Remediation for rule failure

Set the load balancer attribute `deletion_protection.enabled` to `true`.

The examples that follow show how to implement this remediation.

Application Load Balancer - Example

Application Load Balancer configured with deletion protection active. The example is shown in JSON and in YAML.

JSON example

```
{
    "Elb": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Type": "application",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "LoadBalancerAttributes": [
                {
                    "Key": "deletion_protection.enabled",
                    "Value": "true"
                }
            ]
        }
    }
}
```

YAML example

```
Elb:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Type: application
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    LoadBalancerAttributes:
      - Key: deletion_protection.enabled
        Value: 'true'
```

CT.ELASTICLOADBALANCING.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elbv2_deletion_protection_enabled_check
#
# Description:
#   Checks whether Elastic Load Balancing (ELB) has deletion protection activated.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ELBv2 LoadBalancer resource
#      Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 LoadBalancer resource
#       And: 'LoadBalancerAttributes' have not been specified or is an empty list on
 the ELBv2 resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 LoadBalancer resource
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 LoadBalancer
 resource
#       And: 'deletion_protection.enabled' has not been provided as a
 'LoadBalancerAttribute'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 LoadBalancer resource
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 LoadBalancer
 resource
#       And: The 'LoadBalancerAttribute' 'deletion_protection.enabled' has been
 provided and is set to bool(false) or
#            string(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 LoadBalancer Resource
#       And: 'LoadBalancerAttributes' have been specified on the ELBv2 LoadBalancer
 resource
#       And: The 'LoadBalancerAttribute' 'deletion_protection.enabled' has been
 provided and is set to bool(true) or
#            string(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_TYPE = "AWS::ElasticLoadBalancingV2::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_TYPE ]

#
```

```
# Primary Rules
#
rule elbv2_deletion_protection_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %elastic_load_balancers not empty {
    check(%elastic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.5]: Require that application load balancer deletion
 protection is activated
        [FIX]: Set the load balancer attribute 'deletion_protection.enabled' to 'true'.
        >>
}

rule elbv2_deletion_protection_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.5]: Require that application load balancer deletion
 protection is activated
        [FIX]: Set the load balancer attribute 'deletion_protection.enabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(elastic_load_balancer) {
    %elastic_load_balancer {
        # Scenario 2
        LoadBalancerAttributes exists
        LoadBalancerAttributes is_list
        LoadBalancerAttributes not empty

        # Scenario 3, 4 and 5
        some LoadBalancerAttributes[*] {
            Key exists
            Value exists

            Key == "deletion_protection.enabled"
            Value in [ true, "true" ]
        }
    }
}

#
```

```
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: application
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      LoadBalancerAttributes:
      - Key: deletion_protection.enabled
        Value: "true"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
```

```
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: application
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      LoadBalancerAttributes:
      - Key: deletion_protection.enabled
        Value: "false"
```

[CT.ELASTICLOADBALANCING.PR.6] Require that application and network load balancer access logging is activated

This control checks whether your Elastic Load Balancing (ELB) application and network load balancers have logging activated.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.6 example templates

Explanation

Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and to troubleshoot issues.

Usage considerations

- This control applies only to ELB load balancer types of `application` and `network`.

Remediation for rule failure

Set the load balancer attribute `access_logs.s3.enabled` to `true`, and set `access_logs.s3.bucket` to reach an S3 bucket that's configured to receive application load balancer or network load balancer access logs.

The examples that follow show how to implement this remediation.

Application Load Balancer - Example

Application Load Balancer configured with access logging activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "ApplicationLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
```

```
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "Type": "application",
            "LoadBalancerAttributes": [
                {
                    "Key": "access_logs.s3.enabled",
                    "Value": true
                },
                {
                    "Key": "access_logs.s3.bucket",
                    "Value": {
                        "Ref": "LoggingBucket"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    Type: application
    LoadBalancerAttributes:
      - Key: access_logs.s3.enabled
        Value: true
      - Key: access_logs.s3.bucket
        Value: !Ref 'LoggingBucket'
```

The examples that follow show how to implement this remediation.

Network Load Balancer - Example

Network Load Balancer configured with access logging activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "NetworkLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "Type": "network",
            "LoadBalancerAttributes": [
                {
                    "Key": "access_logs.s3.enabled",
                    "Value": true
                },
                {
                    "Key": "access_logs.s3.bucket",
                    "Value": {
                        "Ref": "LoggingBucket"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
NetworkLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    Type: network
    LoadBalancerAttributes:
      - Key: access_logs.s3.enabled
        Value: true
      - Key: access_logs.s3.bucket
        Value: !Ref 'LoggingBucket'
```

CT.ELASTICLOADBALANCING.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elbv2_logging_enabled_check
#
# Description:
#   This control checks whether your Elastic Load Balancing (ELB) application and
 network load balancers have logging activated.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ElasticLoadBalancingV2
 LoadBalancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
#       And: 'Type' is set to a value other than 'application' or 'network'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
#       And: The LoadBalancer is of type 'application' or 'network', or the type has
 not been provided
#       And: 'LoadBalancerAttributes' has not been provided or is an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
#       And: The LoadBalancer is of type 'application' or 'network', or the type has
 not been provided
#       And: A 'LoadBalancerAttributes' with Key 'access_logs.s3.enabled' and
 'access_logs.s3.bucket'
#            has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
#       And: The LoadBalancer is of type 'application' or 'network', or the type has
 not been provided
#       And: A 'LoadBalancerAttributes' with Key 'access_logs.s3.enabled' and
 'access_logs.s3.bucket' has been provided
#       And: 'access_logs.s3.enabled' is set to bool(false) or string(false)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
```

```
#       And: The LoadBalancer is of type 'application' or 'network', or the type has
 not been provided
#       And: A 'LoadBalancerAttributes' with Key 'access_logs.s3.enabled' and
 'access_logs.s3.bucket' has been provided
#       And: 'access_logs.s3.enabled' is set to bool(true) or string(true)
#       And: 'access_logs.s3.bucket' is missing or an empty string value
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancingV2 LoadBalancer
 resource
#       And: The LoadBalancer is of type 'application', or the type has not been
 provided
#       And: A 'LoadBalancerAttributes' with Key 'access_logs.s3.enabled' has been
 provided
#       And: 'access_logs.s3.enabled' is set to bool(true) or string(true)
#       And: 'access_logs.s3.bucket' is provided and a non-empty string value or valid
 local reference
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_TYPE = "AWS::ElasticLoadBalancingV2::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elastic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_TYPE ]

#
# Primary Rules
#
rule elbv2_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %elastic_load_balancers not empty {

    check(%elastic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.6]: Require that application and network load
 balancer access logging is activated
        [FIX]: Set the load balancer attribute 'access_logs.s3.enabled' to 'true', and
 set 'access_logs.s3.bucket' to reach an Amazon S3 bucket that's configured to receive
 application load balancer or network load balancer access logs.
```

```
        >>
}

rule elbv2_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.6]: Require that application and network load
 balancer access logging is activated
        [FIX]: Set the load balancer attribute 'access_logs.s3.enabled' to 'true', and
 set 'access_logs.s3.bucket' to reach an Amazon S3 bucket that's configured to receive
 application load balancer or network load balancer access logs.
        >>
}

#
# Parameterized Rules
#
rule check(elastic_load_balancer) {
    %elastic_load_balancer[ Type not exists or Type in ["application", "network"] ] {
        # Scenario 3
        LoadBalancerAttributes exists
        LoadBalancerAttributes is_list
        LoadBalancerAttributes not empty

        # Scenario 4, 5, 6 and 7
        some LoadBalancerAttributes[*] {
            Key exists
            Value exists

            Key == "access_logs.s3.enabled"
            Value in [ true, "true" ]
        }

        some LoadBalancerAttributes[*] {
            Key exists
            Value exists

            Key == "access_logs.s3.bucket"
            check_is_string_and_not_empty(Value) or
            check_local_references(%INPUT_DOCUMENT, Value, "AWS::S3::Bucket")
        }
    }
```

```
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICLOADBALANCING.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Mappings:
  RegionToELBAccountId:
    us-east-1:
      AccountId: '127311923021'
    us-west-1:
      AccountId: '027434742980'
    us-west-2:
      AccountId: '797873946194'
    ca-central-1:
      AccountId: '985666609251'
    eu-west-1:
      AccountId: '156460612806'
    ap-northeast-1:
      AccountId: '582318560864'
    ap-northeast-2:
      AccountId: '600734575887'
    ap-southeast-1:
      AccountId: '114774131450'
    ap-southeast-2:
      AccountId: '783225319266'
    ap-south-1:
      AccountId: '718504428378'
    us-east-2:
      AccountId: '033677994240'
    sa-east-1:
      AccountId: '507241528517'
    eu-central-1:
      AccountId: '054676820928'
    af-south-1:
      AccountId: '098369216593'
    ap-east-1:
      AccountId: '754344448648'
    ap-southeast-3:
      AccountId: '589379963580'
```

```
    ap-northeast-3:
      AccountId: '383597477331'
    eu-west-2:
      AccountId: '652711504416'
    eu-south-1:
      AccountId: '635631232127'
    eu-west-3:
      AccountId: '009996457667'
    eu-north-1:
      AccountId: '897822967062'
    me-south-1:
      AccountId: '076674570225'
    us-gov-west-1:
      AccountId: '048591011584'
    us-gov-east-1:
      AccountId: '190560391635'
  RegionToARNPrefix:
    us-east-1:
      ARNPrefix: 'arn:aws:'
    us-west-1:
      ARNPrefix: 'arn:aws:'
    us-west-2:
      ARNPrefix: 'arn:aws:'
    ca-central-1:
      ARNPrefix: 'arn:aws:'
    eu-west-1:
      ARNPrefix: 'arn:aws:'
    ap-northeast-1:
      ARNPrefix: 'arn:aws:'
    ap-northeast-2:
      ARNPrefix: 'arn:aws:'
    ap-southeast-1:
      ARNPrefix: 'arn:aws:'
    ap-southeast-2:
      ARNPrefix: 'arn:aws:'
    ap-south-1:
      ARNPrefix: 'arn:aws:'
    us-east-2:
      ARNPrefix: 'arn:aws:'
    sa-east-1:
      ARNPrefix: 'arn:aws:'
    eu-central-1:
      ARNPrefix: 'arn:aws:'
    af-south-1:
```

```
      ARNPrefix: 'arn:aws:'
    ap-east-1:
      ARNPrefix: 'arn:aws:'
    ap-southeast-3:
      ARNPrefix: 'arn:aws:'
    ap-northeast-3:
      ARNPrefix: 'arn:aws:'
    eu-west-2:
      ARNPrefix: 'arn:aws:'
    eu-south-1:
      ARNPrefix: 'arn:aws:'
    eu-west-3:
      ARNPrefix: 'arn:aws:'
    eu-north-1:
      ARNPrefix: 'arn:aws:'
    me-south-1:
      ARNPrefix: 'arn:aws:'
    us-gov-west-1:
      ARNPrefix: 'arn:aws-us-gov:'
    us-gov-east-1:
      ARNPrefix: 'arn:aws-us-gov:'
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - Fn::FindInMap: [RegionToARNPrefix, !Ref 'AWS::Region', ARNPrefix]
                  - 's3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              AWS:
                Fn::FindInMap: [RegionToELBAccountId, !Ref 'AWS::Region', AccountId]
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      Type: application
      LoadBalancerAttributes:
      - Key: access_logs.s3.enabled
        Value: true
      - Key: access_logs.s3.bucket
        Value:
          Ref: LoggingBucket
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      Type: application
```

[CT.ELASTICLOADBALANCING.PR.7] Require any classic load balancer to have multiple Availability Zones configured

This control checks whether an Elastic Load Balancing (ELB) classic load balancer has been configured with multiple Availability Zones.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.7 example templates

Explanation

A Classic Load Balancer can be set up to distribute incoming requests across Amazon EC2 instances in a single Availability Zone or multiple Availability Zones. A Classic Load Balancer that does not span multiple Availability Zones is unable to redirect traffic to targets in another Availability Zone, in case the sole configured Availability Zone becomes unavailable.

Remediation for rule failure

Configure Classic Load Balancers with two or more subnets or Availability Zones.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example One

Classic Load Balancer configured with two Availability Zones. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internet-facing",
            "Listeners": [
                {
                    "InstancePort": "80",
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": "443",
                    "Protocol": "HTTPS",
                    "PolicyNames": [
                        "Sample-SSLNegotiation-Policy"
                    ],
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    }
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Sample-SSLNegotiation-Policy",
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "AvailabilityZones": [
                {
                    "Fn::Select": [
                        0,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                {
```

```
                    "Fn::Select": [
                        1,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internet-facing
    Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
          - Sample-SSLNegotiation-Policy
        SSLCertificateId: !Ref 'ACMCertificate'
    Policies:
      - PolicyName: Sample-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    AvailabilityZones:
      - !Select
        - 0
        - !GetAZs ''
      - !Select
        - 1
        - !GetAZs ''
```

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example Two

Classic Load Balancer configured with two subnets. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internet-facing",
            "Listeners": [
                {
                    "InstancePort": "80",
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": "443",
                    "Protocol": "HTTPS",
                    "PolicyNames": [
                        "Sample-SSLNegotiation-Policy"
                    ],
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    }
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Sample-SSLNegotiation-Policy",
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "AvailabilityZones": [
                {
                    "Fn::Select": [
                        0,
```

```
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                {
                    "Fn::Select": [
                        1,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internet-facing
    Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
          - Sample-SSLNegotiation-Policy
        SSLCertificateId: !Ref 'ACMCertificate'
    Policies:
      - PolicyName: Sample-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    AvailabilityZones:
      - !Select
        - 0
```

```
        - !GetAZs ''
      - !Select
        - 1
        - !GetAZs ''
```

CT.ELASTICLOADBALANCING.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_multiple_az_check
#
# Description:
#   This control checks whether an Elastic Load Balancing (ELB) Classic Load Balancer
 has been configured with multiple Availability Zones.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing load
 balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: Neither 'AvailabilityZones' or 'Subnets' have been specified
#      Then: FAIL
#   Scenario: 3
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'AvailabilityZones' been specified on the Elastic Load Balancing load
 balancer resource
#       And: The number of entries in 'AvailabilityZones' is < 2 or the number of
#            unique 'AvailabilityZones' provided is less than 2 (< 2)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'Subnets' been specified on the Elastic Load Balancing load balancer
 resource
#       And: The number of entries in 'Subnets' is < 2
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'AvailabilityZones' been specified on the Elastic Load Balancing load
 balancer resource
#       And: The number of entries in 'AvailabilityZones' is >= 2
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'Subnets' been specified on the Elastic Load Balancing load balancer
 resource
#       And: The number of entries in 'Subnets' is >= 2
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]
```

```
#
# Primary Rules
#
rule elb_multiple_az_check when is_cfn_template(%INPUT_DOCUMENT)
                               %classic_load_balancers not empty {

    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.7]: Require any classic load balancer to have
 multiple Availability Zones configured
        [FIX]: Configure Classic Load Balancers with two or more subnets or
 Availability Zones.
        >>
}

rule elb_multiple_az_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.7]: Require any classic load balancer to have
 multiple Availability Zones configured
        [FIX]: Configure Classic Load Balancers with two or more subnets or
 Availability Zones.
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
    %classic_load_balancer {
        # Scenario 2
        AvailabilityZones exists or
        Subnets exists

        when AvailabilityZones exists {
            # Scenarios 3 and 5
            two_or_more_entries(AvailabilityZones)
            AvailabilityZones[0] not in AvailabilityZones[1]
        }

        when Subnets exists {
            # Scenarios 4 and 6
```

```
            two_or_more_entries(Subnets)
        }
    }
}

rule two_or_more_entries(list_property) {
    %list_property {
        this is_list
        this not empty
        this[0] exists
        this[1] exists
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
```

```
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internet-facing
      Listeners:
      - Protocol: HTTPS
        InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        PolicyNames:
        - Example-SSLNegotiation-Policy
        SSLCertificateId:
          Ref: ACMCertificate
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      AvailabilityZones:
      - Fn::Select:
        - 0
        - Fn::GetAZs: ''
      - Fn::Select:
        - 1
        - Fn::GetAZs: ''
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
```

```
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
      Subnets:
      - Ref: Subnet
```

[CT.ELASTICLOADBALANCING.PR.8] Require any classic load balancer SSL/HTTPS listener to have a certificate provided by AWS Certificate Manager

This control checks whether classic load balancers use HTTPS/SSL certificates provided by AWS Certificate Manager.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.8 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.8 example templates

Explanation

To create a certificate, you can use either ACM or a tool that supports the SSL and TLS protocols, such as OpenSSL. Security Hub CSPM recommends that you use ACM to create or import certificates for your load balancer.

ACM integrates with Classic Load Balancers, so that you can deploy the certificate on your load balancer. You also should renew these certificates automatically.

Usage considerations

- This control applies only to Classic Load Balancers configured with HTTPS or SSL listeners.

Remediation for rule failure

Configure Classic Load Balancers to use certificates provided by AWS Certificate Manager (ACM).

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example

Classic Load Balancer configured with an HTTPS listener and AWS Certificate Manager SSL certificate. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
```

```
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Example-SSLNegotiation-Policy",
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "Listeners": [
                {
                    "InstancePort": "80",
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": "443",
                    "Protocol": "HTTPS",
                    "PolicyNames": [
                        "Example-SSLNegotiation-Policy"
                    ],
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
```

```
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
          - Example-SSLNegotiation-Policy
        SSLCertificateId: !Ref 'ACMCertificate'
```

CT.ELASTICLOADBALANCING.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   elb_acm_certificate_required_check
#
# Description:
#   This control checks whether Classic Load Balancers use HTTPS/SSL certificates
 provided by AWS Certificate Manager.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing load
 balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: There are no HTTPS or SSL 'Listeners' configured on the load balancer
 resource
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: There are one or more HTTPS or SSL 'Listeners' configured on the load
 balancer resource
#       And: 'SSLCertificateId' on load balancer HTTPS or SSL 'Listeners' is missing or
 not a valid ACM certificate ARN
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: There are one or more HTTPS or SSL 'Listeners' configured on the load
 balancer resource
#       And: 'SSLCertificateId' matches an ACM certificate ARN for all 'HTTPS' and
 'SSL' 'Listeners'
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let ACM_CERTIFICATE_ARN_PATTERN = /arn:aws[a-z0-9\-]*:acm:[a-z0-9\-]+:
\d{12}:certificate\/[\w\-]{1,64}/
let SECURE_LISTENER_PROTOCOLS = ["HTTPS", "SSL"]
let INPUT_DOCUMENT = this

#
# Assignments
#
```

```
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
rule elb_acm_certificate_required_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %classic_load_balancers not empty {

    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.8]: Require any classic load balancer SSL/HTTPS
 listener to have a certificate provided by AWS Certificate Manager
        [FIX]: Configure Classic Load Balancers to use certificates provided by AWS
 Certificate Manager (ACM).
        >>
}

rule elb_acm_certificate_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.8]: Require any classic load balancer SSL/HTTPS
 listener to have a certificate provided by AWS Certificate Manager
        [FIX]: Configure Classic Load Balancers to use certificates provided by AWS
 Certificate Manager (ACM).
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
    %classic_load_balancer [
        filter_load_balancer_with_listeners(this)
    ] {
        Listeners [
            filter_secure_listeners(this)
        ] {
            # Scenarios 3 and 4
            SSLCertificateId exists
            SSLCertificateId == %ACM_CERTIFICATE_ARN_PATTERN or
            check_local_references(%INPUT_DOCUMENT, SSLCertificateId,
 "AWS::CertificateManager::Certificate")
```

```
        }
    }
}

rule filter_load_balancer_with_listeners(classic_load_balancer) {
    %classic_load_balancer {
        Listeners exists
        Listeners is_list
        Listeners not empty
    }
}

rule filter_secure_listeners(listener) {
    %listener {
        Protocol exists
        Protocol in %SECURE_LISTENER_PROTOCOLS
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}
```

```
rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICLOADBALANCING.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
```

```
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
        - Example-SSLNegotiation-Policy
        SSLCertificateId:
          Ref: ACMCertificate
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
```

```
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
        - Example-SSLNegotiation-Policy
```

```
        SSLCertificateId: arn:aws:iam::123456789012:server-certificate/example-
certificate
```

[CT.ELASTICLOADBALANCING.PR.9] Require that an AWS ELB Application or Classic Load Balancer listener is configured with HTTPS or TLS termination

This control checks whether your Elastic Load Balancing (ELB) Classic Load Balancer front-end listeners are configured with HTTPS or SSL protocols.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.9 example templates

Explanation

Before you start to use a load balancer, you must add one or more listeners. A listener is a process that uses the configured protocol and port to check for connection requests. Listeners can support HTTP and HTTPS/TLS protocols. You should always use an HTTPS or TLS listener, so that the load balancer does the work of encryption and decryption in transit.

Remediation for rule failure

Configure Classic Load Balancer front-end listeners with HTTPS or SSL protocols.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example One

Classic Load Balancer configured with an HTTPS Listener. The example is shown in JSON and in YAML.

JSON example

```
{
    "LoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "Subnet"
                }
            ],
            "Listeners": [
                {
                    "Protocol": "HTTPS",
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    },
                    "InstancePort": 80,
                    "LoadBalancerPort": 443
                }
            ]
        }
    }
}
```

YAML example

```
LoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'Subnet'
    Listeners:
```

```
      - Protocol: HTTPS
        SSLCertificateId: !Ref 'ACMCertificate'
        InstancePort: 80
        LoadBalancerPort: 443
```

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example Two

Classic Load Balancer configured with an SSL Listener. The example is shown in JSON and in YAML.

JSON example

```
{
    "LoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "Subnet"
                }
            ],
            "Listeners": [
                {
                    "Protocol": "SSL",
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    },
                    "InstancePort": 80,
                    "LoadBalancerPort": 443
                }
            ]
        }
    }
}
```

YAML example

```
LoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'Subnet'
    Listeners:
      - Protocol: SSL
        SSLCertificateId: !Ref 'ACMCertificate'
        InstancePort: 80
        LoadBalancerPort: 443
```

CT.ELASTICLOADBALANCING.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_tls_https_listeners_only_check
#
# Description:
#   Checks whether Classic Load Balancer front-end listeners are configured with HTTPS
 or SSL protocols.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing
 LoadBalancer resources
```

```
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancing LoadBalancer resource
#       And: 'Listeners' has not been provided or is provided with a value of an empty
 list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancing LoadBalancer resource
#       And: 'Protocol' on LoadBalancer 'Listeners' is not set to 'HTTPS' or 'SSL'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ElasticLoadBalancing LoadBalancer resource
#       And: 'Protocol' is set to 'HTTPS' or 'SSL' for all 'Listeners'
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
rule elb_tls_https_listeners_only_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %classic_load_balancers not empty {

    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.9]: Require that an AWS ELB Application or Classic
 Load Balancer listener is configured with HTTPS or TLS termination
        [FIX]: Configure Classic Load Balancer front-end listeners with HTTPS or SSL
 protocols.
        >>
}
```

```
rule elb_tls_https_listeners_only_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.9]: Require that an AWS ELB Application or Classic
 Load Balancer listener is configured with HTTPS or TLS termination
        [FIX]: Configure Classic Load Balancer front-end listeners with HTTPS or SSL
 protocols.
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
    %classic_load_balancer {
        # Scenario 2
        Listeners exists
        Listeners is_list
        Listeners not empty

        # Scenarios 3 and 4
        Listeners[*] {
           Protocol exists
           Protocol in ["HTTPS", "SSL"]
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
```

```
      - Ref: SubnetTwo
      Listeners:
      - Protocol: HTTPS
        SSLCertificateId: arn:aws:acm:us-
east-1:123456789012:certificate/12345678-12ab-34cd-56ef-12345678
        InstancePort: 80
        LoadBalancerPort: 443
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
```

```
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
```

[CT.ELASTICLOADBALANCING.PR.10] Require an ELB application or classic load balancer to have logging activated

This control checks whether Classic Load Balancers have logging enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.10 example templates

Explanation

Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and to troubleshoot issues.

Remediation for rule failure

Set an `AccessLoggingPolicy` and provide an `S3BucketName` with an Amazon S3 bucket configured to receive classic load balancer access logs.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example

Classic Load Balancer configured with an HTTPS listener and access logging. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Listeners": [
                {
                    "Protocol": "HTTPS",
                    "InstancePort": 80,
                    "LoadBalancerPort": 443
                }
            ],
            "Subnets": [
                {
                    "Ref": "Subnet"
                }
            ],
            "AccessLoggingPolicy": {
                "Enabled": true,
                "S3BucketName": {
                    "Ref": "LoggingBucket"
                }
            }
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Listeners:
      - Protocol: HTTPS
        InstancePort: 80
        LoadBalancerPort: 443
    Subnets:
      - !Ref 'Subnet'
    AccessLoggingPolicy:
      Enabled: true
      S3BucketName: !Ref 'LoggingBucket'
```

CT.ELASTICLOADBALANCING.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_logging_enabled_check
#
# Description:
#   This control checks whether Classic Load Balancers have logging enabled.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any Elastic Load Balancing load
 balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'AccessLoggingPolicy' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'AccessLoggingPolicy' has been provided
#       And: 'Enabled' in 'AccessLoggingPolicy' is missing or has been set to
 bool(false) or 'S3BucketName' is missing
#             or empty string value
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'AccessLoggingPolicy' has been provided
#       And: 'Enabled' has been provided in 'AccessLoggingPolicy' and has been set to
 bool(true)
#       And: 'S3BucketName' has been provided in 'AccessLoggingPolicy' as a non-empty
 string value or
#            valid local reference
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
```

```
rule elb_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %classic_load_balancers not empty {
    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.10]: Require an ELB application or classic load
 balancer to have logging activated
        [FIX]: Set an 'AccessLoggingPolicy' and provide an 'S3BucketName' with an
 Amazon S3 bucket configured to receive classic load balancer access logs.
        >>

}

rule elb_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.10]: Require an ELB application or classic load
 balancer to have logging activated
        [FIX]: Set an 'AccessLoggingPolicy' and provide an 'S3BucketName' with an
 Amazon S3 bucket configured to receive classic load balancer access logs.
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
    %classic_load_balancer {
        # Scenario 2
        AccessLoggingPolicy exists
        AccessLoggingPolicy is_struct

        AccessLoggingPolicy {
            # Scenario 3 and 4
            Enabled exists
            Enabled == true

            S3BucketName exists
            check_is_string_and_not_empty(S3BucketName) or
            check_local_references(%INPUT_DOCUMENT, S3BucketName, "AWS::S3::Bucket")
        }
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.ELASTICLOADBALANCING.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Mappings:
  RegionToELBAccountId:
    us-east-1:
      AccountId: '127311923021'
    us-west-1:
      AccountId: '027434742980'
    us-west-2:
      AccountId: '797873946194'
    ca-central-1:
      AccountId: '985666609251'
    eu-west-1:
      AccountId: '156460612806'
    ap-northeast-1:
      AccountId: '582318560864'
    ap-northeast-2:
      AccountId: '600734575887'
    ap-southeast-1:
      AccountId: '114774131450'
    ap-southeast-2:
      AccountId: '783225319266'
    ap-south-1:
      AccountId: '718504428378'
    us-east-2:
      AccountId: '033677994240'
    sa-east-1:
      AccountId: '507241528517'
    eu-central-1:
      AccountId: '054676820928'
    af-south-1:
      AccountId: '098369216593'
    ap-east-1:
      AccountId: '754344448648'
    ap-southeast-3:
      AccountId: '589379963580'
    ap-northeast-3:
      AccountId: '383597477331'
```

```
    eu-west-2:
      AccountId: '652711504416'
    eu-south-1:
      AccountId: '635631232127'
    eu-west-3:
      AccountId: '009996457667'
    eu-north-1:
      AccountId: '897822967062'
    me-south-1:
      AccountId: '076674570225'
    us-gov-west-1:
      AccountId: '048591011584'
    us-gov-east-1:
      AccountId: '190560391635'
  RegionToARNPrefix:
    us-east-1:
      ARNPrefix: 'arn:aws:'
    us-west-1:
      ARNPrefix: 'arn:aws:'
    us-west-2:
      ARNPrefix: 'arn:aws:'
    ca-central-1:
      ARNPrefix: 'arn:aws:'
    eu-west-1:
      ARNPrefix: 'arn:aws:'
    ap-northeast-1:
      ARNPrefix: 'arn:aws:'
    ap-northeast-2:
      ARNPrefix: 'arn:aws:'
    ap-southeast-1:
      ARNPrefix: 'arn:aws:'
    ap-southeast-2:
      ARNPrefix: 'arn:aws:'
    ap-south-1:
      ARNPrefix: 'arn:aws:'
    us-east-2:
      ARNPrefix: 'arn:aws:'
    sa-east-1:
      ARNPrefix: 'arn:aws:'
    eu-central-1:
      ARNPrefix: 'arn:aws:'
    af-south-1:
      ARNPrefix: 'arn:aws:'
    ap-east-1:
```

```
      ARNPrefix: 'arn:aws:'
    ap-southeast-3:
      ARNPrefix: 'arn:aws:'
    ap-northeast-3:
      ARNPrefix: 'arn:aws:'
    eu-west-2:
      ARNPrefix: 'arn:aws:'
    eu-south-1:
      ARNPrefix: 'arn:aws:'
    eu-west-3:
      ARNPrefix: 'arn:aws:'
    eu-north-1:
      ARNPrefix: 'arn:aws:'
    me-south-1:
      ARNPrefix: 'arn:aws:'
    us-gov-west-1:
      ARNPrefix: 'arn:aws-us-gov:'
    us-gov-east-1:
      ARNPrefix: 'arn:aws-us-gov:'
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
```

```
      - 1
      - Fn::GetAZs: ''
  LoggingBucket:
    Type: AWS::S3::Bucket
  LoggingBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: LoggingBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 's3:PutObject'
            Effect: Allow
            Resource:
              Fn::Join:
                - ''
                - - Fn::FindInMap: [RegionToARNPrefix, !Ref 'AWS::Region', ARNPrefix]
                  - 's3:::'
                  - Ref: LoggingBucket
                  - /AWSLogs/
                  - Ref: AWS::AccountId
                  - /*
            Principal:
              AWS:
                Fn::FindInMap: [RegionToELBAccountId, !Ref 'AWS::Region', AccountId]
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      AccessLoggingPolicy:
        Enabled: true
        S3BucketName:
          Ref: LoggingBucket
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
      Subnets:
      - Ref: SubnetOne
```

```
      - Ref: SubnetTwo
```

[CT.ELASTICLOADBALANCING.PR.11] Require any ELB classic load balancer to have connection draining activated

This control checks whether Elastic Load Balancing (ELB) Classic Load Balancers have connection draining configured.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.11 example templates

Explanation

Activating connection draining on Classic Load Balancers ensures that the load balancer stops sending requests to instances that are de-registering or unhealthy. It keeps the existing connections open. This configuration is particularly useful for instances in Auto Scaling groups, to ensure that connections aren't severed abruptly.

Remediation for rule failure

Configure a `ConnectionDrainingPolicy` on Elastic Load Balancing Classic Load Balancers.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example

Classic Load Balancer configured with connection draining active. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Listeners": [
                {
                    "InstancePort": "80",
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": "443",
                    "Protocol": "HTTPS",
                    "PolicyNames": [
                        "Example-SSLNegotiation-Policy"
                    ],
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    }
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Example-SSLNegotiation-Policy",
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
```

```
                    "Ref": "SubnetTwo"
                }
            ],
            "ConnectionDrainingPolicy": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
          - Example-SSLNegotiation-Policy
        SSLCertificateId: !Ref 'ACMCertificate'
    Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    ConnectionDrainingPolicy:
      Enabled: true
```

CT.ELASTICLOADBALANCING.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_connection_draining_enabled_check
#
# Description:
#   This control checks whether Elastic Load Balancing (ELB) Classic Load Balancers
 have connection draining configured.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing load
 balancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'ConnectionDrainingPolicy' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
#       And: 'ConnectionDrainingPolicy' has been specified
#       And: 'Enabled' in 'ConnectionDrainingPolicy' is missing or has been set to
 bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing load balancer
 resource
```

```
#       And: 'ConnectionDrainingPolicy' has been specified
#       And: 'Enabled' in 'ConnectionDrainingPolicy' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let INPUT_DOCUMENT = this

#
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
rule elb_connection_draining_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %classic_load_balancers not empty {

    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.11]: Require any ELB classic load balancer to have
 connection draining activated
        [FIX]: Configure a 'ConnectionDrainingPolicy' on Elastic Load Balancing Classic
 Load Balancers.
        >>
}

rule elb_connection_draining_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.11]: Require any ELB classic load balancer to have
 connection draining activated
        [FIX]: Configure a 'ConnectionDrainingPolicy' on Elastic Load Balancing Classic
 Load Balancers.
        >>
}

#
# Parameterized Rules
#
```

```
rule check(classic_load_balancer) {
    %classic_load_balancer {
        # Scenario 2
        ConnectionDrainingPolicy exists
        ConnectionDrainingPolicy is_struct

        ConnectionDrainingPolicy {
            # Scenario 3 and 4
            Enabled exists
            Enabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
```

```
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
        - Example-SSLNegotiation-Policy
        SSLCertificateId:
          Ref: ACMCertificate
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
```

```
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      ConnectionDrainingPolicy:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
```

```
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
```

[CT.ELASTICLOADBALANCING.PR.12] Require any ELB classic load balancer SSL/HTTPS listener to have a predefined security policy with a strong configuration

This control checks whether Elastic Load Balancing (ELB) Classic Load Balancer HTTPS/SSL listeners use the predefined security policy `ELBSecurityPolicy-TLS-1-2-2017-01`.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.12 example templates

Explanation

A security policy is a combination of SSL protocols, ciphers, and the server order preference option. Predefined policies control the ciphers, protocols, and preference orders that provide support during SSL negotiations between a client and load balancer.

Using `ELBSecurityPolicy-TLS-1-2-2017-01` can help you to meet compliance and security standards that require you to turn off specific versions of SSL and TLS.

Usage considerations

- This control applies only to Elastic Load Balancing Classic Load Balancers configured with HTTPS or SSL listeners.

Remediation for rule failure

Configure Classic Load Balancer HTTPS/SSL listeners to use the predefined security policy called `ELBSecurityPolicy-TLS-1-2-2017-01`.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example

Classic Load Balancer configured with an HTTPS listener and SSL negotiation policy that references the `ELBSecurityPolicy-TLS-1-2-2017-01` predefined security policy for classic load balancers. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Example-SSLNegotiation-Policy",
```

```
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "Listeners": [
                {
                    "InstancePort": 80,
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": 443,
                    "Protocol": "HTTPS",
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    },
                    "PolicyNames": [
                        "Example-SSLNegotiation-Policy"
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
```

```
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    Listeners:
      - InstancePort: 80
        InstanceProtocol: HTTP
        LoadBalancerPort: 443
        Protocol: HTTPS
        SSLCertificateId: !Ref 'ACMCertificate'
        PolicyNames:
          - Example-SSLNegotiation-Policy
```

CT.ELASTICLOADBALANCING.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_predefined_security_policy_ssl_check
#
# Description:
#   This control checks whether Elastic Load Balancing (ELB) Classic Load Balancer
 HTTPS/SSL listeners use the predefined security policy 'ELBSecurityPolicy-
TLS-1-2-2017-01'.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing
 LoadBalancer resources
#      Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: There are no HTTPS or SSL 'Listeners' configured on the Elastic Load
 Balancing LoadBalancer resource
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'Policies' does not contain a policy with 'PolicyType' equal to
 'SSLNegotiationPolicyType'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'Policies' contains a policy with 'PolicyType' equal to
 'SSLNegotiationPolicyType'
#       And: 'Policies' is missing a 'Reference-Security-Policy' with a value of
#            'ELBSecurityPolicy-TLS-1-2-2017-01'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'Policies' contains a policy with 'PolicyType' equal to
 'SSLNegotiationPolicyType'
#       And: 'Policies' contains a 'Reference-Security-Policy' with a value of
#            'ELBSecurityPolicy-TLS-1-2-2017-01'
#       And: A 'HTTPS' or 'SSL' Listener on the LoadBalancer resource does not
 reference the secure policy
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'Policies' contains a policy with 'PolicyType' equal to
 'SSLNegotiationPolicyType'
#       And: 'Policies' contains a 'Reference-Security-Policy' with a value of
#            'ELBSecurityPolicy-TLS-1-2-2017-01'
#       And: All 'HTTPS' and 'SSL' Listeners on the LoadBalancer resource reference the
 secure policy
#      Then: PASS
```

```
#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let VALID_REFERENCE_SECURITY_POLICIES = [ "ELBSecurityPolicy-TLS-1-2-2017-01" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
rule elb_predefined_security_policy_ssl_check when is_cfn_template(%INPUT_DOCUMENT)
                                                   %classic_load_balancers not empty {

    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.12]: Require any ELB classic load balancer SSL/
HTTPS listener to have a predefined security policy with a strong configuration
        [FIX]: Configure classic load balancer HTTPS/SSL listeners to use the
 predefined security policy called ELBSecurityPolicy-TLS-1-2-2017-01.
        >>
}

rule elb_predefined_security_policy_ssl_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.12]: Require any ELB classic load balancer SSL/
HTTPS listener to have a predefined security policy with a strong configuration
        [FIX]: Configure classic load balancer HTTPS/SSL listeners to use the
 predefined security policy called ELBSecurityPolicy-TLS-1-2-2017-01.
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
```

```
    %classic_load_balancer {
        let elb = this

        # Scenario 2
        Listeners[ Protocol in ["HTTPS", "SSL"] ] {
            %elb.Policies exists
            %elb.Policies is_list
            %elb.Policies not empty

            let secure_policies = %elb.Policies[
                PolicyType == "SSLNegotiationPolicyType"
                some Attributes[*] {
                    Name == "Reference-Security-Policy"
                    Value in %VALID_REFERENCE_SECURITY_POLICIES
                }
            ].PolicyName

            # Scenarios 3 and 4
            %secure_policies not empty

            # Scenarios 5 and 6
            PolicyNames exists
            PolicyNames is_list
            PolicyNames not empty
            some PolicyNames.* in %secure_policies
        }

    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
```

```
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      Listeners:
      - InstancePort: 80
        InstanceProtocol: HTTP
        LoadBalancerPort: 443
        Protocol: HTTPS
        SSLCertificateId:
          Ref: ACMCertificate
        PolicyNames:
        - Example-SSLNegotiation-Policy
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
```

```
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-2016-08
      Listeners:
      - InstancePort: 80
        InstanceProtocol: HTTP
        LoadBalancerPort: 443
        Protocol: HTTPS
        SSLCertificateId: arn:aws:iam::123456789012:server-certificate/example-
certificate
        PolicyNames:
        - Example-SSLNegotiation-Policy
```

[CT.ELASTICLOADBALANCING.PR.13] Require any ELB classic load balancer to have cross-zone load balancing activated

This control checks whether cross-zone load balancing is configured for your Classic Load Balancer.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancing::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.13 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.13 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.13 example templates

Explanation

A load balancer node distributes traffic across the registered targets in its Availability Zone. When cross-zone load balancing is turned off, each load balancer node distributes traffic only across the registered targets in its own Availability Zone. If the number of registered targets is not same across the Availability Zones, traffic is not distributed evenly, so the instances in one zone may become over-utilized, when compared to the instances in another zone. With cross-zone load balancing activated, each load balancer node for your classic load balancer distributes requests evenly across the registered instances in all enabled Availability Zones.

Remediation for rule failure

Set `CrossZone` to `true` on Classic Load Balancers.

The examples that follow show how to implement this remediation.

Classic Load Balancer - Example

Classic Load Balancer configured with cross-zone load balancing active. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClassicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Listeners": [
                {
                    "InstancePort": "80",
                    "InstanceProtocol": "HTTP",
                    "LoadBalancerPort": "443",
                    "Protocol": "HTTPS",
                    "PolicyNames": [
                        "Sample-SSLNegotiation-Policy"
                    ],
                    "SSLCertificateId": {
                        "Ref": "ACMCertificate"
                    }
                }
            ],
            "Policies": [
                {
                    "PolicyName": "Sample-SSLNegotiation-Policy",
                    "PolicyType": "SSLNegotiationPolicyType",
                    "Attributes": [
                        {
                            "Name": "Reference-Security-Policy",
                            "Value": "ELBSecurityPolicy-TLS-1-2-2017-01"
                        }
                    ]
                }
            ],
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "CrossZone": true
        }
```

```
    }
}
```

YAML example

```
ClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    Scheme: internal
    Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
          - Sample-SSLNegotiation-Policy
        SSLCertificateId: !Ref 'ACMCertificate'
    Policies:
      - PolicyName: Sample-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
          - Name: Reference-Security-Policy
            Value: ELBSecurityPolicy-TLS-1-2-2017-01
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    CrossZone: true
```

CT.ELASTICLOADBALANCING.PR.13 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elb_cross_zone_load_balancing_enabled_check
#
# Description:
```

```
#   This control checks whether cross-zone load balancing is configured for your
 Classic Load Balancer.
#
# Reports on:
#   AWS::ElasticLoadBalancing::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing
 LoadBalancer resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'CrossZone' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'CrossZone' has been specified and set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing LoadBalancer
 resource
#       And: 'CrossZone' has been specified and set to bool(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_TYPE = "AWS::ElasticLoadBalancing::LoadBalancer"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let classic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_TYPE ]

#
# Primary Rules
#
rule elb_cross_zone_load_balancing_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %classic_load_balancers not empty
 {
    check(%classic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.13]: Require any ELB classic load balancer to have
 cross-zone load balancing activated
        [FIX]: Set 'CrossZone' to 'true' on Classic Load Balancers.
        >>
}

rule elb_cross_zone_load_balancing_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.13]: Require any ELB classic load balancer to have
 cross-zone load balancing activated
        [FIX]: Set 'CrossZone' to 'true' on Classic Load Balancers.
        >>
}

#
# Parameterized Rules
#
rule check(classic_load_balancer) {
    %classic_load_balancer {
        # Scenario 2
        CrossZone exists

        # Scenario 3 and 4
        CrossZone == true
    }
}

#
# Utility Rules
#
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.13 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
```

```
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ACMCertificate:
    Type: "AWS::CertificateManager::Certificate"
    Properties:
      DomainName: example.com
      ValidationMethod: DNS
      DomainValidationOptions:
        - DomainName: www.example.com
          HostedZoneId: ZZZHHHHWWWWAAA
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - InstancePort: '80'
        InstanceProtocol: HTTP
        LoadBalancerPort: '443'
        Protocol: HTTPS
        PolicyNames:
        - Example-SSLNegotiation-Policy
        SSLCertificateId:
          Ref: ACMCertificate
      Policies:
      - PolicyName: Example-SSLNegotiation-Policy
        PolicyType: SSLNegotiationPolicyType
        Attributes:
        - Name: Reference-Security-Policy
          Value: ELBSecurityPolicy-TLS-1-2-2017-01
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      CrossZone: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  ClassicLoadBalancer:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      Scheme: internal
      Listeners:
      - Protocol: HTTP
        InstancePort: 80
        LoadBalancerPort: 80
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      CrossZone: false
```

[CT.ELASTICLOADBALANCING.PR.14] Require a Network Load Balancer to have cross-zone load balancing activated

This control checks whether a Network Load Balancer (NLB) is configured with cross-zone load balancing.

- **Control objective:** Improve resiliency, Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::LoadBalancer`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.14 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.14 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.14 example templates

Explanation

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is not enabled, each load balancer node distributes traffic only across the registered targets in its own Availability Zone.

Usage considerations

- This control applies only to a Network Load Balancer (`Type` of `network`).
- With a Network Load Balancer, cross-zone load balancing is off by default at the load-balancer level. You can turn it on at any time. For target groups, the default is to use the load balancer setting, but you can override the default by turning cross-zone load balancing on or off explicitly, at the target group level. To ensure that cross-zone load balancing is configured on target groups, use this control in conjunction with `CT.ELASTICLOADBALANCING.PR.15`.

Remediation for rule failure

Set the load balancer attribute `load_balancing.cross_zone.enabled` to `true`.

The examples that follow show how to implement this remediation.

Network Load Balancer - Example

Network Load Balancer configured with cross-zone load balancing enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "NetworkLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internal",
            "Type": "network",
            "Subnets": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "IpAddressType": "ipv4",
            "LoadBalancerAttributes": [
                {
                    "Key": "load_balancing.cross_zone.enabled",
                    "Value": true
                }
            ]
        }
    }
}
```

YAML example

```
NetworkLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Scheme: internal
    Type: network
    Subnets:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    IpAddressType: ipv4
    LoadBalancerAttributes:
      - Key: load_balancing.cross_zone.enabled
        Value: true
```

CT.ELASTICLOADBALANCING.PR.14 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   nlb_cross_zone_load_balancing_enabled_check
#
# Description:
#   This control checks whether a Network Load Balancer (NLB) is configured with cross-
zone load balancing.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::LoadBalancer
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any ELBv2 Load Balancer resources
#      Then: SKIP
```

```
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 Load Balancer resource
#       And: 'Type' has been provided and is set to a value other than 'network'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 Load Balancer resource
#       And: 'Type' has been provided and set to 'network'
#       And: 'LoadBalancerAttributes' has not been provided or has been provided as an
 empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 Load Balancer resource
#       And: 'Type' has been provided and set to 'network'
#       And: 'LoadBalancerAttributes' has been provided as a non-empty list
#       And: 'LoadBalancerAttributes' does not contain an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 Load Balancer resource
#       And: 'Type' has been provided and set to 'network'
#       And: 'LoadBalancerAttributes' has been provided as a non-empty list
#       And: 'LoadBalancerAttributes' contains an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled' and
#            'Value' equal to a value other than bool(true) or string(true)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an ELBv2 Load Balancer resource
#       And: 'Type' has been provided and set to 'network'
#       And: 'LoadBalancerAttributes' has been provided as a non-empty list
#       And: 'LoadBalancerAttributes' contains an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled' and
#            'Value' equal to bool(true) or string(true)
#      Then: PASS

#
# Constants
#
let ELASTIC_LOAD_BALANCER_V2_TYPE = "AWS::ElasticLoadBalancingV2::LoadBalancer"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let elastic_load_balancers = Resources.*[ Type == %ELASTIC_LOAD_BALANCER_V2_TYPE ]

#
# Primary Rules
#
rule nlb_cross_zone_load_balancing_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %elastic_load_balancers not empty
 {

    check(%elastic_load_balancers.Properties)
        <<
        [CT.ELASTICLOADBALANCING.PR.14]: Require a Network Load Balancer to have cross-
zone load balancing activated
        [FIX]: Set the load balancer attribute 'load_balancing.cross_zone.enabled' to
 'true'.
        >>
}

rule nlb_cross_zone_load_balancing_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTIC_LOAD_BALANCER_V2_TYPE) {

    check(%INPUT_DOCUMENT.%ELASTIC_LOAD_BALANCER_V2_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.14]: Require a Network Load Balancer to have cross-
zone load balancing activated
        [FIX]: Set the load balancer attribute 'load_balancing.cross_zone.enabled' to
 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(elastic_load_balancer) {
    %elastic_load_balancer[ Type == "network" ] {
        # Scenario 2
        LoadBalancerAttributes exists
        LoadBalancerAttributes is_list
        LoadBalancerAttributes not empty
```

```
        # Scenarios 3, 4 and 5
        some LoadBalancerAttributes[*] {
            Key exists
            Value exists

            Key == "load_balancing.cross_zone.enabled"
            Value in [ true, "true" ]
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.14 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
```

```
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: Vpc
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: Vpc
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  NetworkLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: network
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
      LoadBalancerAttributes:
      - Key: load_balancing.cross_zone.enabled
        Value: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
```

```
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: Vpc
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: Vpc
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  NetworkLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internal
      Type: network
      Subnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      IpAddressType: ipv4
```

[CT.ELASTICLOADBALANCING.PR.15] Require that an Elastic Load Balancing v2 target group does not explicitly disable cross-zone load balancing

This control checks whether an Elastic Load Balancing v2 target group is configured so that it does not explicitly turn off cross-zone load balancing.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::ElasticLoadBalancingV2::TargetGroup`
- **CloudFormation guard rule:** CT.ELASTICLOADBALANCING.PR.15 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.ELASTICLOADBALANCING.PR.15 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.ELASTICLOADBALANCING.PR.15 example templates

Explanation

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is not enabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

The target group's cross-zone load balancing setting determines the load balancing behavior for the entire target group. For example, if cross-zone load balancing is enabled at the load balancer level, but not enabled at the target group level, traffic sent to the target group is not routed across Availability Zones.

Usage considerations

- This control checks only to ensure that cross-zone load balancing has not been explicitly turned off on an ELB target group. To ensure that cross-zone load balancing is enabled, be sure to enable this control in conjunction with related proactive controls that check load balancers directly.
- If you turn on cross-zone load balancing, you can't start a zonal shift. For more information, see [Resources supported for zonal shifts] (https://docs.aws.amazon.com/r53recovery/latest/dg/arc-zonal-shift.resource-types.html) in the Amazon Route 53 Application Recovery Controller Developer Guide.

Remediation for rule failure

Do not set the load balancer attribute `load_balancing.cross_zone.enabled` to adopt the default value of `use_load_balancer_configuration`. Do not explicitly set the attribute to true, nor to the value `use_load_balancer_configuration`.

The examples that follow show how to implement this remediation.

Elastic Load Balancer target group - Example

Elastic Load Balancer target group configured to enable cross-zone load balancing. The example is shown in JSON and in YAML.

JSON example

```
{
    "TargetGroup": {
        "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
        "Properties": {
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": {
                "Ref": "VPC"
            },
            "TargetGroupAttributes": [
                {
                    "Key": "load_balancing.cross_zone.enabled",
                    "Value": true
                }
            ]
        }
    }
}
```

YAML example

```
TargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
```

```
    Protocol: HTTP
    Port: 80
    VpcId: !Ref 'VPC'
    TargetGroupAttributes:
      - Key: load_balancing.cross_zone.enabled
        Value: true
```

CT.ELASTICLOADBALANCING.PR.15 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elbv2_target_group_cross_zone_check
#
# Description:
#   This control checks whether an Elastic Load Balancing v2 target group is configured
 so that it does not explicitly turn off cross-zone load balancing.
#
# Reports on:
#   AWS::ElasticLoadBalancingV2::TargetGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elastic Load Balancing v2 target
 group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing v2 target group
 resource
#       And: 'TargetGroupAttributes' has been provided as a non-empty list
```

```
#       And: 'TargetGroupAttributes' contain an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled'
#            and 'Value' equal to a value other than bool(true), 'true' or
 'use_load_balancer_configuration'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing v2 target group
 resource
#       And: 'TargetGroupAttributes' has not been provided or has been provided as a
 list that
#            does not contain an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled'
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elastic Load Balancing v2 target group
 resource
#       And: 'TargetGroupAttributes' has been provided as a non-empty list
#       And: 'TargetGroupAttributes' contains an entry with a 'Key' equal to
 'load_balancing.cross_zone.enabled' and
#            'Value' equal to bool(true), string(true) or
 'use_load_balancer_configuration'
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let ELBV2_TARGET_GROUP_TYPE = "AWS::ElasticLoadBalancingV2::TargetGroup"
let ALLOWED_CROSS_ZONE_VALUES = ["true", true, "use_load_balancer_configuration"]

#
# Assignments
#
let elbv2_target_groups = Resources.*[ Type == %ELBV2_TARGET_GROUP_TYPE ]

#
# Primary Rules
#
rule elbv2_target_group_cross_zone_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %elbv2_target_groups not empty {

    check(%elbv2_target_groups.Properties)
```

```
        <<
        [CT.ELASTICLOADBALANCING.PR.15]: Require that an Elastic Load Balancing v2
 target group does not explicitly disable cross-zone load balancing
        [FIX]: Do not set the load balancer attribute
 'load_balancing.cross_zone.enabled' to adopt the default value of
 'use_load_balancer_configuration'. Do not explicitly set the attribute to true, nor to
 the value 'use_load_balancer_configuration'.
        >>
}

rule elbv2_target_group_cross_zone_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELBV2_TARGET_GROUP_TYPE) {

    check(%INPUT_DOCUMENT.%ELBV2_TARGET_GROUP_TYPE.resourceProperties)
        <<
        [CT.ELASTICLOADBALANCING.PR.15]: Require that an Elastic Load Balancing v2
 target group does not explicitly disable cross-zone load balancing
        [FIX]: Do not set the load balancer attribute
 'load_balancing.cross_zone.enabled' to adopt the default value of
 'use_load_balancer_configuration'. Do not explicitly set the attribute to true, nor to
 the value 'use_load_balancer_configuration'.
        >>
}

#
# Parameterized Rules
#
rule check(elbv2_target_group) {
    %elbv2_target_group {
        # Scenarios 2 and 3
        TargetGroupAttributes not exists or
        check_target_group_with_attributes(this)
    }
}

rule check_target_group_with_attributes(target_group) {
    %target_group {
        TargetGroupAttributes exists
        TargetGroupAttributes is_list
        TargetGroupAttributes[
            Key exists
            Key == "load_balancing.cross_zone.enabled"
        ] {
            # Scenario 4
```

```
            Value exists
            Value in %ALLOWED_CROSS_ZONE_VALUES
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.ELASTICLOADBALANCING.PR.15 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Protocol: HTTP
      Port: 80
      VpcId:
```

```
        Ref: VPC
      TargetGroupAttributes:
      - Key: load_balancing.cross_zone.enabled
        Value: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Protocol: HTTP
      Port: 80
      VpcId:
        Ref: VPC
      TargetGroupAttributes:
      - Key: load_balancing.cross_zone.enabled
        Value: false
```

Amazon Elastic Map Reduce (Amazon EMR) controls

Topics

- [CT.EMR.PR.1] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data at rest in Amazon S3
- [CT.EMR.PR.2] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data at rest in Amazon S3 with an AWS KMS key
- [CT.EMR.PR.3] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured with EBS volume local disk encryption using an AWS KMS key

- [CT.EMR.PR.4] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data in transit

[CT.EMR.PR.1] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data at rest in Amazon S3

This control checks whether an Amazon EMR security configuration is configured to encrypt EMR File System (EMRFS) objects at rest in Amazon S3.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EMR::SecurityConfiguration`
- **CloudFormation guard rule:** CT.EMR.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EMR.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EMR.PR.1 example templates

Explanation

Amazon S3 encryption works with EMR File System (EMRFS) objects that are read from and written to Amazon S3. When you enable encryption at rest, you specify Amazon S3 server-side encryption (SSE) or client-side encryption (CSE) as the default encryption mode. Optionally, you can specify different encryption methods for individual buckets by using per bucket encryption overrides.

Remediation for rule failure

In the `EncryptionConfiguration` parameter, set the value of `EnableAtRestEncryption` to true, and provide an `AtRestEncryptionConfiguration` configuration.

The examples that follow show how to implement this remediation.

Amazon EMR security configuration - Example

An Amazon EMR security configuration configured to encrypt EMR File System (EMRFS) objects at rest in Amazon S3. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityConfiguration": {
        "Type": "AWS::EMR::SecurityConfiguration",
        "Properties": {
            "SecurityConfiguration": {
                "EncryptionConfiguration": {
                    "EnableInTransitEncryption": false,
                    "EnableAtRestEncryption": true,
                    "AtRestEncryptionConfiguration": {
                        "S3EncryptionConfiguration": {
                            "EncryptionMode": "SSE-S3"
                        }
                    }
                }
            }
        }
    }
}
```

YAML example

```
SecurityConfiguration:
  Type: AWS::EMR::SecurityConfiguration
  Properties:
    SecurityConfiguration:
      EncryptionConfiguration:
        EnableInTransitEncryption: false
        EnableAtRestEncryption: true
        AtRestEncryptionConfiguration:
          S3EncryptionConfiguration:
            EncryptionMode: SSE-S3
```

CT.EMR.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   emr_sec_config_encryption_at_rest_s3_check
#
# Description:
#   This control checks whether an Amazon EMR security configuration is configured to
 encrypt EMR File System (EMRFS) objects at rest in Amazon S3.
#
# Reports on:
#   AWS::EMR::SecurityConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1`
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EMR security configuration
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has not been
 provided
#            or has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EncryptionMode' in
 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
#            has not been provided or has been provided as an empty string
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EncryptionMode' in
 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
#            has been provided as a non-empty string
#      Then: PASS

#
# Constants
#
let EMR_SECURITY_CONFIGURATION_TYPE = "AWS::EMR::SecurityConfiguration"
let INPUT_DOCUMENT = this

#
# Assignments
```

```
#
let emr_security_configurations = Resources.*[ Type ==
 %EMR_SECURITY_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule emr_sec_config_encryption_at_rest_s3_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %emr_security_configurations not
 empty {
    check(%emr_security_configurations.Properties)
        <<
        [CT.EMR.PR.1]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data at rest in Amazon S3
        [FIX]: In the 'EncryptionConfiguration' parameter, set the value of
 'EnableAtRestEncryption' to true, and provide an 'AtRestEncryptionConfiguration'
 configuration.
        >>
}

rule emr_sec_config_encryption_at_rest_s3_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EMR_SECURITY_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%EMR_SECURITY_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.EMR.PR.1]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data at rest in Amazon S3
        [FIX]: In the 'EncryptionConfiguration' parameter, set the value of
 'EnableAtRestEncryption' to true, and provide an 'AtRestEncryptionConfiguration'
 configuration.
        >>
}

#
# Parameterized Rules
#
rule check(emr_security_configuration) {
    %emr_security_configuration {
        SecurityConfiguration exists
        SecurityConfiguration is_struct

        SecurityConfiguration {
            # Scenario 2
            EncryptionConfiguration exists
            EncryptionConfiguration is_struct
```

```
            EncryptionConfiguration {
                # Scenario 3
                EnableAtRestEncryption exists
                EnableAtRestEncryption == true

                # Scenario 4
                AtRestEncryptionConfiguration exists
                AtRestEncryptionConfiguration is_struct

                # Scenarios 5 and 6
                AtRestEncryptionConfiguration {
                    S3EncryptionConfiguration exists
                    S3EncryptionConfiguration is_struct

                    S3EncryptionConfiguration {
                        EncryptionMode exists
                        check_is_string_and_not_empty(EncryptionMode)
                    }
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
```

```
}
```

CT.EMR.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: true
          AtRestEncryptionConfiguration:
            S3EncryptionConfiguration:
              EncryptionMode: SSE-S3
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableAtRestEncryption: false
          EnableInTransitEncryption: false
```

[CT.EMR.PR.2] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data at rest in Amazon S3 with an AWS KMS key

This control checks whether an Amazon EMR security configuration is configured to encrypt EMR File System (EMRFS) objects at rest in Amazon S3 with an AWS KMS key.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EMR::SecurityConfiguration`
- **CloudFormation guard rule:** CT.EMR.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EMR.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EMR.PR.2 example templates

Explanation

Amazon S3 encryption works with EMR File System (EMRFS) objects that are read from and written to Amazon S3. When you enable encyption at rest, you specify Amazon S3 server-side encryption (SSE) or client-side encryption (CSE) as the default encryption mode. Optionally, you can specify different encryption methods for individual buckets using per bucket encryption overrides.

Remediation for rule failure

In the `EncryptionConfiguration` parameter, set `EnableAtRestEncryption` to true, and provide an `AtRestEncryptionConfiguration` configuration, with `EncryptionMode` set to `SSE-KMS` or `CSE-KMS`.

The examples that follow show how to implement this remediation.

Amazon EMR security configuration - Example

An Amazon EMR security configuration configured to encrypt EMR File System (EMRFS) objects at rest in Amazon S3 with AWS KMS. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityConfiguration": {
        "Type": "AWS::EMR::SecurityConfiguration",
        "Properties": {
            "SecurityConfiguration": {
                "EncryptionConfiguration": {
                    "EnableInTransitEncryption": false,
                    "EnableAtRestEncryption": true,
                    "AtRestEncryptionConfiguration": {
                        "S3EncryptionConfiguration": {
                            "EncryptionMode": "SSE-KMS",
                            "AwsKmsKey": {
                                "Fn::GetAtt": [
                                    "KmsKey",
                                    "Arn"
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
}
```

YAML example

```
SecurityConfiguration:
  Type: AWS::EMR::SecurityConfiguration
  Properties:
    SecurityConfiguration:
      EncryptionConfiguration:
        EnableInTransitEncryption: false
```

```
        EnableAtRestEncryption: true
        AtRestEncryptionConfiguration:
          S3EncryptionConfiguration:
            EncryptionMode: SSE-KMS
            AwsKmsKey: !GetAtt 'KmsKey.Arn'
```

CT.EMR.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   emr_sec_config_encryption_at_rest_s3_kms_check
#
# Description:
#   This control checks whether an Amazon EMR security configuration is configured to
 encrypt EMR File System (EMRFS) objects at rest in Amazon S3 with an AWS KMS key.
#
# Reports on:
#   AWS::EMR::SecurityConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EMR security configuration
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has not been
 provided
#            or has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EncryptionMode' in
 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
#            has not been provided or has been provided and set to a value other than
#            a KMS-based encryption mode ('SSE-KMS', 'CSE-KMS')
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EncryptionMode' in
 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
```

```
#            has been provided and set to a KMS-based encryption mode ('SSE-KMS', 'CSE-
KMS')
#       And: 'Overrides' in 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
 has been
#            provided as a non-empty list where one or more entries does not contain
 'EncryptionMode',
#            or contains 'EncryptionMode' set to a value other than a KMS-based
 encryption mode
#            ('SSE-KMS', 'CSE-KMS')
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EncryptionMode' in
 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
#            has been provided and set to a KMS-based encryption mode ('SSE-KMS', 'CSE-
KMS')
#       And: 'Overrides' in 'AtRestEncryptionConfiguration.S3EncryptionConfiguration'
 has not
#            been provided, or has been provided as an empty list, or list where every
 entry
#            contains 'EncryptionMode' set to a KMS-based encryption mode ('SSE-KMS',
 'CSE-KMS')
#      Then: PASS

#
# Constants
#
let EMR_SECURITY_CONFIGURATION_TYPE = "AWS::EMR::SecurityConfiguration"
let S3_KMS_ENCRYPTION_MODES = [ "SSE-KMS", "CSE-KMS" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let emr_security_configurations = Resources.*[ Type ==
 %EMR_SECURITY_CONFIGURATION_TYPE ]
```

```
#
# Primary Rules
#
rule emr_sec_config_encryption_at_rest_s3_kms_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %emr_security_configurations
 not empty {
    check(%emr_security_configurations.Properties)
        <<
        [CT.EMR.PR.2]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data at rest in Amazon S3 with an AWS KMS key
        [FIX]: In the 'EncryptionConfiguration' parameter, set 'EnableAtRestEncryption'
 to true, and provide an 'AtRestEncryptionConfiguration' configuration, with
 'EncryptionMode' set to 'SSE-KMS' or 'CSE-KMS'.
        >>
}

rule emr_sec_config_encryption_at_rest_s3_kms_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EMR_SECURITY_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%EMR_SECURITY_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.EMR.PR.2]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data at rest in Amazon S3 with an AWS KMS key
        [FIX]: In the 'EncryptionConfiguration' parameter, set 'EnableAtRestEncryption'
 to true, and provide an 'AtRestEncryptionConfiguration' configuration, with
 'EncryptionMode' set to 'SSE-KMS' or 'CSE-KMS'.
        >>
}

#
# Parameterized Rules
#
rule check(emr_security_configuration) {
    %emr_security_configuration {
        SecurityConfiguration exists
        SecurityConfiguration is_struct

        SecurityConfiguration {
            # Scenario 2
            EncryptionConfiguration exists
            EncryptionConfiguration is_struct

            EncryptionConfiguration {
                # Scenario 3
```

```
                EnableAtRestEncryption exists
                EnableAtRestEncryption == true

                # Scenario 4
                AtRestEncryptionConfiguration exists
                AtRestEncryptionConfiguration is_struct

                # Scenarios 5, 6 and 7
                AtRestEncryptionConfiguration {
                    S3EncryptionConfiguration exists
                    S3EncryptionConfiguration is_struct

                    let s3_encryption_configuration = S3EncryptionConfiguration

                    %s3_encryption_configuration {
                        check_kms_key_configuration(this)
                    }

                    %s3_encryption_configuration [
                        Overrides exists
                        Overrides is_list
                        Overrides not empty
                    ] {
                        Overrides[*] {
                            check_kms_key_configuration(this)
                        }
                    }
                }
            }
        }
    }
}

rule check_kms_key_configuration(s3_encryption_config) {
    %s3_encryption_config {
        EncryptionMode exists
        EncryptionMode in %S3_KMS_ENCRYPTION_MODES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
```

```
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EMR.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KmsKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: "*"
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: true
          AtRestEncryptionConfiguration:
```

```
            S3EncryptionConfiguration:
              EncryptionMode: SSE-KMS
              AwsKmsKey:
                Fn::GetAtt:
                - KmsKey
                - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: false
```

[CT.EMR.PR.3] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured with EBS volume local disk encryption using an AWS KMS key

This control checks whether Amazon EMR security configurations are configured with local disk encryption enabled, using EBS volume encryption and AWS KMS.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EMR::SecurityConfiguration`
- **CloudFormation guard rule:** CT.EMR.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EMR.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EMR.PR.3 example templates

Explanation

For data at rest, EMR provides the option to encrypt local disk storage. The local storage consists of EC2 instance store volumes and the attached Amazon Elastic Block Store (EBS) storage, which are provisioned with your cluster. The EBS encryption option encrypts the EBS root device volume and its attached storage volumes. The EBS encryption option is available only when you specify AWS Key Management Service as your key provider. AWS Control Tower recommends using EBS encryption.

Usage considerations

- If you create a cluster in a Region where Amazon EC2 encryption of EBS volumes is enabled by default for your account, an EBS volume is encrypted even when local disk encryption is not enabled.
- With local disk encryption enabled in a security configuration, the Amazon EMR settings take precedence over the Amazon EC2 encryption-by-default settings for cluster EC2 instances.

Remediation for rule failure

In the `EncryptionConfiguration` parameter, set the value of `EnableAtRestEncryption` to true, and provide an `AtRestEncryptionConfiguration` configuration, containing an `LocalDiskEncryptionConfiguration` configuration that sets `EnableEbsEncryption` to true.

The examples that follow show how to implement this remediation.

Amazon EMR security configuration - Example

An Amazon EMR security configuration configured with EBS encryption using AWS KMS. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityConfiguration": {
        "Type": "AWS::EMR::SecurityConfiguration",
        "Properties": {
            "SecurityConfiguration": {
                "EncryptionConfiguration": {
                    "EnableInTransitEncryption": false,
                    "EnableAtRestEncryption": true,
                    "AtRestEncryptionConfiguration": {
                        "LocalDiskEncryptionConfiguration": {
                            "EnableEbsEncryption": true,
                            "EncryptionKeyProviderType": "AwsKms",
                            "AwsKmsKey": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
                        }
                    }
                }
            }
        }
    }
}
```

YAML example

```
SecurityConfiguration:
  Type: AWS::EMR::SecurityConfiguration
  Properties:
    SecurityConfiguration:
      EncryptionConfiguration:
        EnableInTransitEncryption: false
        EnableAtRestEncryption: true
        AtRestEncryptionConfiguration:
          LocalDiskEncryptionConfiguration:
            EnableEbsEncryption: true
            EncryptionKeyProviderType: AwsKms
            AwsKmsKey: arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

CT.EMR.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   emr_sec_config_ebs_encryption_check
#
# Description:
#   This control checks whether Amazon EMR security configurations are configured with
 local disk encryption enabled, using EBS volume encryption and AWS KMS.
#
# Reports on:
#   AWS::EMR::SecurityConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EMR security configuration
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has not been
 provided
#            or has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EnableEbsEncryption' in
 'AtRestEncryptionConfiguration.LocalDiskEncryptionConfiguration'
#            has not been provided or has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableAtRestEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'AtRestEncryptionConfiguration' has been provided as a struct
#       And: 'EnableEbsEncryption' in
 'AtRestEncryptionConfiguration.LocalDiskEncryptionConfiguration'
#            has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let EMR_SECURITY_CONFIGURATION_TYPE = "AWS::EMR::SecurityConfiguration"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let emr_security_configurations = Resources.*[ Type ==
 %EMR_SECURITY_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule emr_sec_config_ebs_encryption_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %emr_security_configurations not empty {
    check(%emr_security_configurations.Properties)
        <<
        [CT.EMR.PR.3]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured with EBS volume local disk encryption using an AWS KMS key
        [FIX]: In the 'EncryptionConfiguration' parameter, set the value of
 'EnableAtRestEncryption' to true, and provide an 'AtRestEncryptionConfiguration'
 configuration, containing an 'LocalDiskEncryptionConfiguration' configuration that
 sets 'EnableEbsEncryption' to true.
        >>
}

rule emr_sec_config_ebs_encryption_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EMR_SECURITY_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%EMR_SECURITY_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.EMR.PR.3]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured with EBS volume local disk encryption using an AWS KMS key
        [FIX]: In the 'EncryptionConfiguration' parameter, set the value of
 'EnableAtRestEncryption' to true, and provide an 'AtRestEncryptionConfiguration'
 configuration, containing an 'LocalDiskEncryptionConfiguration' configuration that
 sets 'EnableEbsEncryption' to true.
        >>
}

#
# Parameterized Rules
#
rule check(emr_security_configuration) {
    %emr_security_configuration {
        SecurityConfiguration exists
        SecurityConfiguration is_struct

        SecurityConfiguration {
            # Scenario 2
```

```
            EncryptionConfiguration exists
            EncryptionConfiguration is_struct

            EncryptionConfiguration {
                # Scenario 3
                EnableAtRestEncryption exists
                EnableAtRestEncryption == true

                # Scenario 4
                AtRestEncryptionConfiguration exists
                AtRestEncryptionConfiguration is_struct

                # Scenarios 5 and 6
                AtRestEncryptionConfiguration {
                    LocalDiskEncryptionConfiguration exists
                    LocalDiskEncryptionConfiguration is_struct

                    LocalDiskEncryptionConfiguration {
                        EnableEbsEncryption exists
                        EnableEbsEncryption == true
                    }
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.EMR.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KmsKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: "*"
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: true
          AtRestEncryptionConfiguration:
            LocalDiskEncryptionConfiguration:
              EnableEbsEncryption: true
              EncryptionKeyProviderType: AwsKms
              AwsKmsKey:
                Fn::GetAtt:
                - KmsKey
                - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: false
```

[CT.EMR.PR.4] Require that an Amazon Elastic MapReduce (EMR) security configuration is configured to encrypt data in transit

This control checks whether an Amazon EMR security configuration is configured to require encryption in transit.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::EMR::SecurityConfiguration`
- **CloudFormation guard rule:** CT.EMR.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.EMR.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.EMR.PR.4 example templates

Explanation

For data in transit, EMR security configurations provide you two options. You can create PEM certificates, zip them in a file, and reference them from Amazon S3, or you can implement a certificate custom provider in Java and specify the S3 path to the JAR. In either case, EMR downloads artifacts to each node in the cluster automatically, and later uses them to implement

open-source, in-transit encryption features. For more information on how these certificates are used with different big data technologies, see Amazon EMR documentation.

Usage considerations

- Several encryption mechanisms are associated with in-transit encryption. These mechanisms are open-source features, they are application-specific, and they may vary by Amazon EMR release. For more information, see Encryption in transit in the *Amazon EMR Management Guide*.

Remediation for rule failure

In the `EncryptionConfiguration` parameter, set the `EnableInTransitEncryption` parameter to true, and provide an `InTransitEncryptionConfiguration` configuration.

The examples that follow show how to implement this remediation.

Amazon EMR security configuration - Example

An Amazon EMR security configuration configured to require encryption of data in transit. The example is shown in JSON and in YAML.

JSON example

```
{
    "SecurityConfiguration": {
        "Type": "AWS::EMR::SecurityConfiguration",
        "Properties": {
            "SecurityConfiguration": {
                "EncryptionConfiguration": {
                    "EnableAtRestEncryption": false,
                    "EnableInTransitEncryption": true,
                    "InTransitEncryptionConfiguration": {
                        "TLSCertificateConfiguration": {
                            "CertificateProviderType": "PEM",
                            "S3Object": "s3://MyConfigStore/artifacts/MyCerts.zip"
                        }
                    }
                }
            }
```

```
        }
    }
}
```

YAML example

```
SecurityConfiguration:
  Type: AWS::EMR::SecurityConfiguration
  Properties:
    SecurityConfiguration:
      EncryptionConfiguration:
        EnableAtRestEncryption: false
        EnableInTransitEncryption: true
        InTransitEncryptionConfiguration:
          TLSCertificateConfiguration:
            CertificateProviderType: PEM
            S3Object: s3://MyConfigStore/artifacts/MyCerts.zip
```

CT.EMR.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   emr_sec_config_encryption_in_transit_check
#
# Description:
#   This control checks whether an Amazon EMR security configuration is configured to
 require encryption in transit.
#
# Reports on:
#   AWS::EMR::SecurityConfiguration
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any EMR security configuration
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableInTransitEncryption' in 'EncryptionConfiguration' has not been
 provided
#            or has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableInTransitEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'InTransitEncryptionConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableInTransitEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'InTransitEncryptionConfiguration' has been provided as a struct
#       And: 'CertificateProviderType' in
 'InTransitEncryptionConfiguration.TLSCertificateConfiguration'
#            has not been provided or has been provided and set to an empty string
```

```
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an EMR security configuration resource
#       And: 'EncryptionConfiguration' in 'SecurityConfiguration' has been provided as
 a struct
#       And: 'EnableInTransitEncryption' in 'EncryptionConfiguration' has been provided
 and
#            set to bool(true)
#       And: 'InTransitEncryptionConfiguration' has been provided as a struct
#       And: 'CertificateProviderType' in
 'InTransitEncryptionConfiguration.TLSCertificateConfiguration'
#            has been provided and set to a non-empty string
#      Then: PASS

#
# Constants
#
let EMR_SECURITY_CONFIGURATION_TYPE = "AWS::EMR::SecurityConfiguration"
let INPUT_DOCUMENT = this

#
# Assignments
#
let emr_security_configurations = Resources.*[ Type ==
 %EMR_SECURITY_CONFIGURATION_TYPE ]

#
# Primary Rules
#
rule emr_sec_config_encryption_in_transit_check when is_cfn_template(%INPUT_DOCUMENT)
                                                     %emr_security_configurations not
 empty {
    check(%emr_security_configurations.Properties)
        <<
        [CT.EMR.PR.4]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data in transit
        [FIX]: In the 'EncryptionConfiguration' parameter, set
 the 'EnableInTransitEncryption' parameter to true, and provide an
 'InTransitEncryptionConfiguration' configuration.
        >>
}
```

```
rule emr_sec_config_encryption_in_transit_check when is_cfn_hook(%INPUT_DOCUMENT,
 %EMR_SECURITY_CONFIGURATION_TYPE) {
    check(%INPUT_DOCUMENT.%EMR_SECURITY_CONFIGURATION_TYPE.resourceProperties)
        <<
        [CT.EMR.PR.4]: Require that an Amazon Elastic MapReduce (EMR) security
 configuration is configured to encrypt data in transit
        [FIX]: In the 'EncryptionConfiguration' parameter, set
 the 'EnableInTransitEncryption' parameter to true, and provide an
 'InTransitEncryptionConfiguration' configuration.
        >>
}

#
# Parameterized Rules
#
rule check(emr_security_configuration) {
    %emr_security_configuration {
        SecurityConfiguration exists
        SecurityConfiguration is_struct

        SecurityConfiguration {
            # Scenario 2
            EncryptionConfiguration exists
            EncryptionConfiguration is_struct

            EncryptionConfiguration {
                # Scenario 3
                EnableInTransitEncryption exists
                EnableInTransitEncryption == true

                # Scenario 4
                InTransitEncryptionConfiguration exists
                InTransitEncryptionConfiguration is_struct

                # Scenarios 5 and 6
                InTransitEncryptionConfiguration {
                    TLSCertificateConfiguration exists
                    TLSCertificateConfiguration is_struct

                    TLSCertificateConfiguration {
                        CertificateProviderType exists
                        check_is_string_and_not_empty(CertificateProviderType)
                    }
                }
```

```
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.EMR.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableAtRestEncryption: false
```

```
          EnableInTransitEncryption: true
          InTransitEncryptionConfiguration:
            TLSCertificateConfiguration:
              CertificateProviderType: PEM
              S3Object: s3://MyConfigStore/artifacts/MyCerts.zip
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SecurityConfiguration:
    Type: AWS::EMR::SecurityConfiguration
    Properties:
      SecurityConfiguration:
        EncryptionConfiguration:
          EnableInTransitEncryption: false
          EnableAtRestEncryption: false
```

AWS Glue controls

Topics

- [CT.GLUE.PR.1] Require an AWS Glue job to have an associated security configuration

[CT.GLUE.PR.1] Require an AWS Glue job to have an associated security configuration

This control checks whether an AWS Glue job has an associated security configuration.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Glue::Job`
- **CloudFormation guard rule:** CT.GLUE.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.GLUE.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.GLUE.PR.1 example templates

Explanation

In AWS Glue, a security configuration contains the properties that are needed when you write encrypted data. Security configurations for an AWS Glue job must be configured to specify how the data is encrypted at the Amazon S3 target. Encryption helps protect the data from unauthorized access and exposure.

Remediation for rule failure

Set the `SecurityConfiguration` parameter to the name of an AWS Glue security configuration.

The examples that follow show how to implement this remediation.

AWS Glue job - Example

An AWS Glue job configured with an associated security configuration. The example is shown in JSON and in YAML.

JSON example

```
{
    "GlueJob": {
        "Type": "AWS::Glue::Job",
        "Properties": {
            "Command": {
                "Name": "glueetl",
                "ScriptLocation": "s3://example-glue-script-bucket/scripts"
            },
            "Name": "sample-glue-job",
            "Role": {
                "Ref": "IAMRole"
            },
            "GlueVersion": "2.0",
            "SecurityConfiguration": {
```

```
                "Ref": "GlueSecurityConfig"
            }
        }
    }
}
```

YAML example

```
GlueJob:
  Type: AWS::Glue::Job
  Properties:
    Command:
      Name: glueetl
      ScriptLocation: s3://example-glue-script-bucket/scripts
    Name: sample-glue-job
    Role: !Ref 'IAMRole'
    GlueVersion: '2.0'
    SecurityConfiguration: !Ref 'GlueSecurityConfig'
```

CT.GLUE.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   glue_job_security_config_check
#
# Description:
#   This control checks whether an AWS Glue job has an associated security
 configuration.
#
# Reports on:
#   AWS::Glue::Job
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
```

```
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS Glue job resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS Glue job resource
#       And: 'SecurityConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS Glue job resource
#       And: 'SecurityConfiguration 'has been provided as an empty string or invalid
 local
#            reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation Hook Document
#       And: The input document contains an AWS Glue job resource
#       And: 'SecurityConfiguration' has been provided as a non-empty string or valid
#            local reference to an AWS Glue security configuration resource
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let GLUE_JOB_TYPE = "AWS::Glue::Job"

#
# Assignments
#
let glue_jobs = Resources.*[ Type == %GLUE_JOB_TYPE ]

#
# Primary Rules
#
rule glue_job_security_config_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %glue_jobs not empty {
    check(%glue_jobs.Properties)
```

```
        <<
        [CT.GLUE.PR.1]: Require an AWS Glue job to have an associated security
 configuration
        [FIX]: Set the 'SecurityConfiguration' parameter to the name of an AWS Glue
 security configuration.
        >>
}

rule glue_job_security_config_check when is_cfn_hook(%INPUT_DOCUMENT, %GLUE_JOB_TYPE) {
    check(%INPUT_DOCUMENT.%GLUE_JOB_TYPE.resourceProperties)
        <<
        [CT.GLUE.PR.1]: Require an AWS Glue job to have an associated security
 configuration
        [FIX]: Set the 'SecurityConfiguration' parameter to the name of an AWS Glue
 security configuration.
        >>
}

#
# Parameterized Rules
#
rule check(glue_job) {
    %glue_job{
        # Scenario 2
        SecurityConfiguration exists
         # Scenario 3 and 4
        check_is_string_and_not_empty(SecurityConfiguration) or
        check_local_references(%INPUT_DOCUMENT, SecurityConfiguration,
 "AWS::Glue::SecurityConfiguration")
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
```

```
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_RESOURCE_TYPE) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_RESOURCE_TYPE) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_RESOURCE_TYPE
    }
}
```

CT.GLUE.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "glue.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
  Key:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeySpec: SYMMETRIC_DEFAULT
      EnableKeyRotation: true
  GlueSecurityConfig:
    Type: AWS::Glue::SecurityConfiguration
    Properties:
      Name:
        Fn::Sub: ${AWS::StackName}-example
      EncryptionConfiguration:
        S3Encryptions:
        - KmsKeyArn:
            Fn::GetAtt: [Key, Arn]
          S3EncryptionMode: SSE-KMS
  GlueJob:
    Type: AWS::Glue::Job
    Properties:
      Command:
        Name: glueetl
        ScriptLocation: "s3://example-glue-script-bucket/scripts"
      Name:
        Fn::Sub: ${AWS::StackName}-example
      Role:
```

```
        Ref: IAMRole
      GlueVersion: "2.0"
      SecurityConfiguration:
        Ref: GlueSecurityConfig
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "glue.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
  GlueJob:
    Type: AWS::Glue::Job
    Properties:
      Command:
        Name: glueetl
        ScriptLocation: "s3://example-glue-script-bucket/scripts"
      Name:
        Fn::Sub: ${AWS::StackName}-example
      Role:
        Ref: IAMRole
      GlueVersion: "2.0"
```

Amazon GuardDuty controls

Topics

- [CT.GUARDDUTY.PR.1] Require an Amazon GuardDuty detector to have Amazon S3 protection activated

[CT.GUARDDUTY.PR.1] Require an Amazon GuardDuty detector to have Amazon S3 protection activated

This control checks whether Amazon S3 protection is enabled on an Amazon GuardDuty detector.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::GuardDuty::Detector`
- **CloudFormation guard rule:** CT.GUARDDUTY.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.GUARDDUTY.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.GUARDDUTY.PR.1 example templates

Explanation

Amazon GuardDuty monitors threats against your Amazon S3 resources by analyzing AWS CloudTrail management events and CloudTrail S3 data events. These data sources monitor different kinds of activity. For example, CloudTrail management events for S3 include operations that list or configure S3 buckets, such as `ListBuckets`, `DeleteBuckets`, and `PutBucketReplication`. Examples of data events for S3 include object-level API operations, such as `GetObject`, `ListObjects`, `DeleteObject`, and `PutObject`.

Amazon GuardDuty monitoring of AWS CloudTrail management events is on by default for all accounts that have enabled GuardDuty, and it is not configurable. Amazon S3 data event logs are a configurable data source in GuardDuty.

AWS Control Tower recommends that you enable Amazon S3 protection in GuardDuty. If the feature is not enabled, GuardDuty cannot fully monitor or generate findings for suspicious access to data stored in your Amazon S3 buckets.

Remediation for rule failure

Set `DataSources.S3Logs` to true.

The examples that follow show how to implement this remediation.

GuardDuty Detector - Example

Amazon GuardDuty detector with Amazon S3 protection enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "GuardDutyDetector": {
        "Type": "AWS::GuardDuty::Detector",
        "Properties": {
            "Enable": true,
            "DataSources": {
                "S3Logs": {
                    "Enable": true
                }
            }
        }
    }
}
```

YAML example

```
GuardDutyDetector:
  Type: AWS::GuardDuty::Detector
  Properties:
    Enable: true
    DataSources:
      S3Logs:
        Enable: true
```

CT.GUARDDUTY.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   guardduty_s3_protection_enabled_check
#
# Description:
#   Checks if Amazon S3 protection is enabled on an Amazon GuardDuty detector.
#
# Reports on:
#   AWS::GuardDuty::Detector
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any Amazon GuardDuty detector
 resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains a GuardDuty detector resource
#        And: 'Enable' has not been specified or specified with value is bool(false)
#       Then: FAIL
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains a GuardDuty detector resource
#        And: 'Enable' is specified with a value of bool(true)
#        And: 'DataSources.S3Logs' has not been specified
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains a GuardDuty detector resource
#        And: 'Enable' is specified and value is bool(true)
#        And: 'DataSources.S3Logs' has been specified
```

```
#        And: 'Enable' has not been specified within 'S3Logs' or has been specified
 with a value of bool(false)
#       Then: FAIL
#   Scenario: 5
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains a GuardDuty detector resource
#        And: 'Enable' is specified and value is bool(true)
#        And: 'DataSources.S3Logs' has been specified
#        And: 'Enable' has been specified within 'S3Logs' with a value of bool(true)
#       Then: PASS

#
# Constants
#
let GUARDDUTY_DETECTOR_TYPE = "AWS::GuardDuty::Detector"
let INPUT_DOCUMENT = this

#
# Assignments
#
let guardduty_detectors = Resources.*[ Type == %GUARDDUTY_DETECTOR_TYPE ]

#
# Primary Rules
#
rule guardduty_s3_protection_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %guardduty_detectors not empty {
    check(%guardduty_detectors.Properties)
        <<
        [CT.GUARDDUTY.PR.1]: Require an Amazon GuardDuty detector to have Amazon S3
 protection activated
        [FIX]: Set 'DataSources.S3Logs' to true.
        >>
}

rule guardduty_s3_protection_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %GUARDDUTY_DETECTOR_TYPE) {
    check(%INPUT_DOCUMENT.%GUARDDUTY_DETECTOR_TYPE.resourceProperties)
        <<
        [CT.GUARDDUTY.PR.1]: Require an Amazon GuardDuty detector to have Amazon S3
 protection activated
        [FIX]: Set 'DataSources.S3Logs' to true.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(guardduty_detector) {
    %guardduty_detector {
        # Scenario: 2
        Enable exists
        Enable == true
        # Scenario: 3
        DataSources exists
        DataSources is_struct
        DataSources {
            # Scenario: 4 and 5
            S3Logs exists
            S3Logs is_struct
            S3Logs {
                Enable exists
                Enable == true
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.GUARDDUTY.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  GuardDutyDetector:
    Type: AWS::GuardDuty::Detector
    Properties:
      Enable: true
      DataSources:
        S3Logs:
          Enable: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  GuardDutyDetector:
    Type: AWS::GuardDuty::Detector
    Properties:
      Enable: true
      DataSources:
        S3Logs:
          Enable: false
```

AWS Identity and Access Management (IAM) controls

Topics

- [CT.IAM.PR.1] Require that an AWS Identity and Access Management (IAM) inline policy does not have a statement that includes "*" in the Action and Resource elements
- [CT.IAM.PR.2] Require that AWS Identity and Access Management (IAM) customer-managed policies do not contain a statement that includes "*" in the Action and Resource elements

- [CT.IAM.PR.3] Require that AWS Identity and Access Management (IAM) customer-managed policies do not have wildcard service actions
- [CT.IAM.PR.4] Require that an AWS Identity and Access Management(IAM) user does not have an inline or managed policy attached
- [CT.IAM.PR.5] Require that AWS Identity and Access Management (IAM) inline policies do not have wildcard service actions

[CT.IAM.PR.1] Require that an AWS Identity and Access Management (IAM) inline policy does not have a statement that includes "*" in the Action and Resource elements

This control checks that AWS Identity and Access Management (IAM) inline policies do not include `Effect`: `Allow` with `Action`: `*` over `Resource`: `*`.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::IAM::Policy`, `AWS::IAM::Role`, `AWS::IAM::User`, `AWS::IAM::Group`
- **CloudFormation guard rule:** CT.IAM.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.IAM.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.IAM.PR.1 example templates

Explanation

IAM policies define a set of privileges that are granted to users, groups, or roles. In alignment with industry-standard security advice, AWS recommends that your policies grant least privilege, which means to grant only the permissions that are required to perform a task. When you provide full administrative privileges instead of the minimum set of permissions that the user requires, you may expose the resources to unwanted actions.

Instead of allowing full administrative privileges, determine the specific actions that your users must carry out, and then craft policies that let the users perform only those tasks. It is more secure to start with a minimum set of permissions and grant additional permissions when necessary. Do not start with lenient permissions and try to tighten them later.

Remove IAM policies that have a statement with `Effect`: `Allow` that permit `Action`: `*` over `Resource`: `*`.

Usage considerations

- This control applies only to IAM inline policies with statements that contain an `Effect` of `Allow` and that contain both the `Action` and the `Resource` element.

Remediation for rule failure

Remove IAM inline policy statements with `Effect`: `Allow` that permit `Action`: `*` over `Resource`: `*`.

The examples that follow show how to implement this remediation.

IAM Policy - Example One

IAM inline policy configured to allow retrieval of objects from an Amazon S3 bucket. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMPolicy": {
        "Type": "AWS::IAM::Policy",
        "Properties": {
            "PolicyName": "sample-inline-policy",
            "Roles": [
                {
                    "Ref": "IAMRole"
                }
            ],
            "PolicyDocument": {
                "Version": "2012-10-17",
```

```
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "s3:GetObject"
                        ],
                        "Resource": [
                            "arn:aws:s3:::amzn-s3-demo-bucket/*"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML example

```
IAMPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: sample-inline-policy
    Roles:
      - !Ref 'IAMRole'
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
          Resource:
            - arn:aws:s3:::amzn-s3-demo-bucket/*
```

The examples that follow show how to implement this remediation.

IAM Role - Example Two

IAM role configured with an inline policy allowing retrieval of objects from an Amazon S3 bucket. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "AWS": {
                                "Ref": "AWS::AccountId"
                            }
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Policies": [
                {
                    "PolicyName": "sample-inline-policy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:GetObject"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                                ]
                            }
                        ]
                    }
                }
            ]
        }
    }
```

```
}
```

YAML example

```
IAMRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Principal:
            AWS: !Ref 'AWS::AccountId'
          Action:
            - sts:AssumeRole
    Policies:
      - PolicyName: sample-inline-policy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
            - Effect: Allow
              Action:
                - s3:GetObject
              Resource:
                - arn:aws:s3:::amzn-s3-demo-bucket/*
```

CT.IAM.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Name:
#   iam_inline_policy_no_statements_with_admin_access_check
#
# Description:
#   This control checks that AWS Identity and Access Management (IAM) inline policies
 do not include "Effect": "Allow" with "Action": "*" over "Resource": "*".
```

```
#
# Reports on:
#   AWS::IAM::Policy, AWS::IAM::Role, AWS::IAM::User, AWS::IAM::Group
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any IAM policy, IAM role, IAM user or
 IAM group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has no statements with 'Effect' set to 'Allow'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy does not have both Action and resource statements
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resources
#       And: 'Policies' is not provided or is an empty list
#      Then: SKIP
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resources
#       And: 'Policies' is provided as a non-empty list
#       And: All IAM policy documents in 'Policies' have no statements with 'Effect'
 set to 'Allow'
#      Then: SKIP
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
```

```
#       And: The policy statement has one or more Action statements and one or more
 Resource statements
#       And: At least one Action statement allows all actions (Action value of '*')
#       And: At least one Resource statement is a wildcard representing all resources
 (Resource value of '*')
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resources
#       And: 'Policies' is provided as a non-empty list
#       And: IAM policy document in 'Policies' has a statement with 'Effect' set to
 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: At least one Action statement allows all actions (Action value of '*')
#       And: At least one Resource statement is a wildcard representing all resources
 (Resource value of '*')
#      Then: FAIL
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more Action statements and one or more Resource
 statements
#       And: No Action statements allow administrator access (Action value of '*')
#       And: No Resources are wildcards representing all resources (Resource value of
 '*')
#      Then: PASS
#   Scenario: 9
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resources
#       And: 'Policies' is provided as a non-empty list
#       And: IAM policy document in 'Policies' has a statement with 'Effect' set to
 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: No Action statements allow administrator access (Action value of '*')
#       And: No Resources are wildcards representing all resources (Resource value of
 '*')
#      Then: PASS

#
# Constants
#
let AWS_IAM_POLICY_TYPE = "AWS::IAM::Policy"
let AWS_IAM_ROLE_TYPE = "AWS::IAM::Role"
```

```
let AWS_IAM_USER_TYPE = "AWS::IAM::User"
let AWS_IAM_GROUP_TYPE = "AWS::IAM::Group"
let INPUT_DOCUMENT = this

#
# Assignments
#
let iam_policies = Resources.*[ Type == %AWS_IAM_POLICY_TYPE ]
let iam_principals = Resources.*[
    Type == %AWS_IAM_ROLE_TYPE or
    Type == %AWS_IAM_USER_TYPE or
    Type == %AWS_IAM_GROUP_TYPE
]

#
# Primary Rules
#
rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                                %iam_policies not
 empty {
    check_policy(%iam_policies.Properties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_POLICY_TYPE) {
    check_policy(%INPUT_DOCUMENT.%AWS_IAM_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                                %iam_principals not
 empty {
    check_principal(%iam_principals.Properties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_ROLE_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_ROLE_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_USER_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_USER_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_inline_policy_no_statements_with_admin_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_GROUP_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_GROUP_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.1]: Require that an AWS Identity and Access Management (IAM) inline
 policy does not have a statement that includes "*" in the Action and Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}
```

```
#
# Parameterized Rules
#
rule check_policy(policy) {
    %policy [
        filter_policy_document_with_statement_provided(this)
    ] {
        PolicyDocument {
           check_statement(Statement)
       }
    }
}

rule check_principal(iam_principal) {
    %iam_principal [
        filter_iam_principal_with_inline_policy_provided(this)
    ] {
        Policies[*] {
            check_policy(this)
       }
    }
}

rule check_statement(statement) {
    %statement [
        filter_allow_on_action_and_resource(this)
    ] {
        Action exists
        Resource exists
        check_admin_access(Action, Resource)
    }
}

rule filter_allow_on_action_and_resource(statement) {
    %statement {
        Effect == "Allow"
        Action exists
        Resource exists
    }
}

rule filter_policy_document_with_statement_provided(policy) {
    %policy {
        PolicyDocument exists
```

```
        PolicyDocument is_struct
        PolicyDocument {
            Statement exists
            filter_statement_non_empty_list(Statement) or
            Statement is_struct
        }
    }
}

rule filter_iam_principal_with_inline_policy_provided(iam_principal) {
    %iam_principal {
        Policies exists
        Policies is_list
        Policies not empty
    }
}

rule filter_statement_non_empty_list(statement) {
    %statement {
        this is_list
        this not empty
    }
}

rule check_admin_access(actions, resources) {
    when some %actions[*] == "*" {
        %resources[*] != "*"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.IAM.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-inline-policy
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action:
          - s3:GetObject
          Resource:
          - arn:aws:s3:::amzn-s3-demo-bucket/*
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-inline-policy
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action: '*'
          Resource: '*'
```

[CT.IAM.PR.2] Require that AWS Identity and Access Management (IAM) customer-managed policies do not contain a statement that includes "*" in the Action and Resource elements

This control checks whether AWS Identity and Access Management (IAM) customer managed policies do not include `Effect`: `Allow` with `Action`: * over `Resource`: *.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::IAM::ManagedPolicy`

- **CloudFormation guard rule:** CT.IAM.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.IAM.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.IAM.PR.2 example templates

Explanation

IAM policies define a set of privileges that are granted to users, groups, or roles. In alignment with industry-standard security advice, AWS recommends that your policies grant least privilege, which means to grant only the permissions that are required to perform a task. When you provide full privileges instead of the minimum set of permissions that the user requires, you may expose the resources to unwanted actions.

Instead of allowing full privileges, determine the specific actions that your users must carry out, and then craft policies that let the users perform only those tasks. It is more secure to start with a minimum set of permissions and grant additional permissions when necessary. Do not start with lenient permissions and try to tighten them later.

Remove IAM policies that have a statement with `Effect`: `Allow` that permit `Action`: `*` over `Resource`: `*`.

Usage considerations

- This control checks IAM customer-managed policies only. It does not check inline and AWS-managed policies.
- This control applies only to IAM inline policies with statements that contain an `Effect` of `Allow` and that contain both the `Action` and the `Resource` element.

Remediation for rule failure

Remove IAM inline policy statements with `Effect`: `Allow` that permit `Action`: `*` over `Resource`: `*`.

The examples that follow show how to implement this remediation.

IAM Managed Policy - Example

IAM managed policy configured to allow retrieval of objects from an Amazon S3 bucket. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMManagedPolicy": {
        "Type": "AWS::IAM::ManagedPolicy",
        "Properties": {
            "Roles": [
                {
                    "Ref": "IAMRole"
                }
            ],
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "s3:GetObject"
                        ],
                        "Resource": [
                            "arn:aws:s3:::amzn-s3-demo-bucket/*"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML example

```
IAMManagedPolicy:
  Type: AWS::IAM::ManagedPolicy
```

```
  Properties:
    Roles:
      - !Ref 'IAMRole'
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
          Resource:
            - arn:aws:s3:::amzn-s3-demo-bucket/*
```

CT.IAM.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Name:
#   iam_managed_policy_no_statements_with_admin_access_check
#
# Description:
#   This control checks whether AWS Identity and Access Management (IAM) customer
 managed policies do not include "Effect": "Allow" with "Action": "*" over "Resource":
 "*".
#
# Reports on:
#   AWS::IAM::ManagedPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any IAM managed policy resources
#      Then: SKIP
```

```
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM managed policy resource
#       And: The policy has no statements with 'Effect' set to 'Allow'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM managed policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy does not have both Action and Resource statements
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM managed policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy statement has one or more Action statements and one or more
 Resource statements
#       And: Within a single policy statement at least one Action statement allows all
 actions (Action value of '*')
#       And: Within the same policy statement at least one Resource statement is a
 wildcard representing all
#            resources (Resource value of '*')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM managed policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more Action statements and one or more Resource
 statements
#       And: Within a single policy statement no Action statements allow all actions
 (Action value of '*')
#       And: Within the same policy statement no Resources are wildcards representing
 all
#            resources (Resource value of '*')
#      Then: PASS

#
# Constants
#
let AWS_IAM_MANAGED_POLICY_TYPE = "AWS::IAM::ManagedPolicy"
let INPUT_DOCUMENT = this

#
# Assignments
```

```
#
let iam_managed_policies = Resources.*[ Type == %AWS_IAM_MANAGED_POLICY_TYPE ]

#
# Primary Rules
#
rule iam_managed_policy_no_statements_with_admin_access_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %iam_managed_policies not empty {
    check(%iam_managed_policies.Properties)
        <<
        [CT.IAM.PR.2]: Require that AWS Identity and Access Management (IAM) customer-
managed policies do not contain a statement that includes "*" in the Action and
 Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

rule iam_managed_policy_no_statements_with_admin_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_MANAGED_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%AWS_IAM_MANAGED_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.2]: Require that AWS Identity and Access Management (IAM) customer-
managed policies do not contain a statement that includes "*" in the Action and
 Resource elements
            [FIX]: Remove IAM inline policy statements with "Effect": "Allow" that
 permit "Action": "*" over "Resource": "*".
        >>
}

#
# Parameterized Rules
#
rule check(policy) {
    %policy [
        filter_policy_document_with_statement_provided(this)
    ] {
        PolicyDocument {
           check_statement(Statement)
       }
    }
}
```

```
rule check_statement(statement) {
    %statement [
        filter_allow_on_action_and_resource(this)
    ] {
        Action exists
        Resource exists
        check_admin_access(Action, Resource)
    }
}

rule filter_allow_on_action_and_resource(statement) {
    %statement {
        Effect == "Allow"
        Action exists
        Resource exists
    }
}

rule filter_policy_document_with_statement_provided(policy) {
    %policy {
        PolicyDocument exists
        PolicyDocument is_struct
        PolicyDocument {
            Statement exists
            filter_statement_non_empty_list(Statement) or
            Statement is_struct
        }
    }
}

rule filter_statement_non_empty_list(statement) {
    %statement {
        this is_list
        this not empty
    }
}

rule check_admin_access(actions, resources) {
    when some %actions[*] == "*" {
        %resources[*] != "*"
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.IAM.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
      - Ref: IAMRole
      PolicyDocument:
```

```
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action:
          - s3:GetObject
          Resource:
          - arn:aws:s3:::amzn-s3-demo-bucket/*
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action: '*'
          Resource: '*'
```

[CT.IAM.PR.3] Require that AWS Identity and Access Management (IAM) customer-managed policies do not have wildcard service actions

This control checks that AWS Identity and Access Management (IAM) customer-managed policies do not contain statements of `Effect`: `Allow` with `Action`: `Service:*` (for example, s3:*) for individual AWS services, and that the policies do not use the combination of `NotAction` with an `Effect` of `Allow`.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::IAM::ManagedPolicy`
- **CloudFormation guard rule:** CT.IAM.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.IAM.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.IAM.PR.3 example templates

Explanation

When you assign permissions to AWS services, it is important to scope the allowed IAM actions in your IAM policies. We recommend that you provision least-privilege permissions by restricting IAM policies to required actions only. Overly permissive policies can lead to privilege escalation, if the policies are attached to an IAM principal that may not require the permission.

Usage considerations

- This control checks IAM customer-managed policies only. It does not check inline and AWS-managed policies.
- This control applies only to IAM customer-managed policies with statements that contain an `Effect` of `Allow` with an `Action` or `NotAction` element present.

Remediation for rule failure

Remove statements from IAM customer-managed policies with `Effect`: `Allow` and `Action`: `service:*` or `Effect`: `Allow` and `NotAction`.

The examples that follow show how to implement this remediation.

IAM Managed Policy - Example

IAM managed policy configured to allow the Amazon S3 `ListBucket` action. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMManagedPolicy": {
        "Type": "AWS::IAM::ManagedPolicy",
        "Properties": {
            "Roles": [
                {
                    "Ref": "IAMRole"
                }
            ],
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "s3:ListBucket"
                        ],
                        "Resource": [
                            "*"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML example

```
IAMManagedPolicy:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    Roles:
      - !Ref 'IAMRole'
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:ListBucket
          Resource:
            - '*'
```

CT.IAM.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Name:
#   iam_managed_policy_no_statements_with_full_access_check
#
# Description:
#   This control checks that AWS Identity and Access Management (IAM) customer-
managed policies do not contain statements of "Effect": "Allow" with "Action":
 "Service:*" (for example, s3:*) for individual AWS services, and that the policies do
 not use the combination of "NotAction" with an "Effect" of "Allow".
#
# Reports on:
#   AWS::IAM::ManagedPolicy
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any IAM Managed Policy resources
```

```
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM Managed Policy resource
#       And: The policy has no statements with 'Effect' set to 'Allow'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM Managed Policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: At least one 'Action' statement allows full access to a service ('Action'
 has a value 'service:*')
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM Managed Policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'NotAction' statements
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM Managed Policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: No 'Action' statements allow full access to a service ('Action' does not
 have a value 'service:*')
#      Then: PASS

#
# Constants
#
let AWS_IAM_MANAGED_POLICY_TYPE = "AWS::IAM::ManagedPolicy"
let WILDCARD_ACTION_PATTERN = /^[\w]*[:]*\*$/
let INPUT_DOCUMENT = this

#
# Assignments
#
let iam_managed_policies = Resources.*[ Type == %AWS_IAM_MANAGED_POLICY_TYPE ]

#
# Primary Rules
#
```

```
rule iam_managed_policy_no_statements_with_full_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                                %iam_managed_policies
 not empty {
    check(%iam_managed_policies.Properties)
        <<
        [CT.IAM.PR.3]: Require that AWS Identity and Access Management (IAM) customer-
managed policies do not have wildcard service actions
            [FIX]: Remove statements from IAM customer-managed policies with "Effect":
 "Allow" and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

rule iam_managed_policy_no_statements_with_full_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_MANAGED_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%AWS_IAM_MANAGED_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.3]: Require that AWS Identity and Access Management (IAM) customer-
managed policies do not have wildcard service actions
            [FIX]: Remove statements from IAM customer-managed policies with "Effect":
 "Allow" and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

#
# Parameterized Rules
#
rule check(policy) {
    %policy [
        filter_policy_document_with_statement_provided(this)
    ] {
        PolicyDocument {
            check_statement_no_wildcard_actions(Statement)
            check_statement_no_not_action(Statement)
       }
    }
}

rule check_statement_no_wildcard_actions(statement) {
    %statement [
        filter_allow_on_action(this)
    ] {
        Action exists
        check_no_wildcard_action(Action)
```

```
    }
}

rule check_statement_no_not_action(statement) {
    %statement [
        filter_allow(this)
    ] {
        NotAction not exists
    }
}

rule filter_allow_on_action(statement) {
    %statement {
        Effect == "Allow"
        Action exists
    }
}

rule filter_allow(statement) {
    %statement {
        Effect == "Allow"
    }
}

rule filter_policy_document_with_statement_provided(policy) {
    %policy {
        PolicyDocument exists
        PolicyDocument is_struct
        PolicyDocument {
            Statement exists
            filter_statement_non_empty_list(Statement) or
            Statement is_struct
        }
    }
}

rule filter_statement_non_empty_list(statement) {
    %statement {
        this is_list
        this not empty
    }
}

rule check_no_wildcard_action(actions) {
```

```
    %actions[*] {
        this != %WILDCARD_ACTION_PATTERN
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.IAM.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMManagedPolicy:
```

```
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action:
          - s3:ListBucket
          Resource:
          - '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action: s3:*
          Resource: '*'
```

[CT.IAM.PR.4] Require that an AWS Identity and Access Management(IAM) user does not have an inline or managed policy attached

This control checks whether your AWS Identity and Access Management (IAM) user has inline or managed (AWS and customer) policies directly attached. Instead, IAM users should inherit permissions from IAM groups or roles.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::IAM::User`, `AWS::IAM::Policy`, `AWS::IAM::ManagedPolicy`
- **CloudFormation guard rule:** CT.IAM.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.IAM.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.IAM.PR.4 example templates

Explanation

By default, IAM users, groups, and roles have no access to AWS resources. IAM policies grant privileges to users, groups, or roles. We recommend that you apply IAM policies directly to groups and roles, but not to users. As the number of users grows, assigning privileges at the group or role level reduces the complexity of access management. Reducing access management complexity may in turn reduce the opportunity for a principal to receive or retain excessive privileges inadvertently.

Remediation for rule failure

Configure IAM users to inherit permissions from IAM groups.

The examples that follow show how to implement this remediation.

IAM User - Example

IAM user configured with no IAM policy or managed policy attachments. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMUser": {
        "Type": "AWS::IAM::User"
    }
}
```

YAML example

```
IAMUser:
  Type: AWS::IAM::User
```

The examples that follow show how to implement this remediation.

IAM Policy - Example

IAM policy configured with no IAM user associations. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMPolicy": {
        "Type": "AWS::IAM::Policy",
        "Properties": {
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "cloudformation:DescribeStacks"
```

```
                        ],
                        "Resource": "*"
                    }
                ]
            },
            "PolicyName": "sample-policy",
            "Roles": [
                {
                    "Ref": "IAMRole"
                }
            ]
        }
    }
}
```

YAML example

```
IAMPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - cloudformation:DescribeStacks
          Resource: '*'
    PolicyName: sample-policy
    Roles:
      - !Ref 'IAMRole'
```

The examples that follow show how to implement this remediation.

IAM Managed Policy - Example

IAM managed policy configured with no IAM user associations. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMManagedPolicy": {
        "Type": "AWS::IAM::ManagedPolicy",
        "Properties": {
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "cloudformation:DescribeStacks"
                        ],
                        "Resource": "*"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
IAMManagedPolicy:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - cloudformation:DescribeStacks
          Resource: '*'
```

CT.IAM.PR.4 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   iam_user_no_policies_check
#
# Description:
#   This control checks whether your AWS Identity and Access Management (IAM) user has
 inline or managed (AWS and customer) policies directly attached. Instead, IAM users
 should inherit permissions from IAM groups or roles.
#
# Reports on:
#   AWS::IAM::User, AWS::IAM::Policy, AWS::IAM::ManagedPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any IAM user, policy or managed policy
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM user resource
#       And: 'Policies' or 'ManagedPolicyArns' have been specified as a non-empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy or managed policy resource
#       And: 'Users' has been specified and is a non-empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM user resource
#       And: 'Policies' has not been been specified or is an empty list
#       And: 'ManagedPolicyArns' has not been been specified or is an empty list
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an IAM policy or managed policy resource
#       And: 'Users' has not been specified or is an empty list
#      Then: PASS

#
# Constants
#
let IAM_USER_TYPE = "AWS::IAM::User"
let IAM_POLICY_TYPE = "AWS::IAM::Policy"
let IAM_MANAGED_POLICY_TYPE = "AWS::IAM::ManagedPolicy"
let INPUT_DOCUMENT = this

#
# Assignments
#
let iam_users = Resources.*[ Type == %IAM_USER_TYPE ]
let iam_policies = Resources.*[ Type == %IAM_POLICY_TYPE ]
let iam_managed_policies = Resources.*[ Type == %IAM_MANAGED_POLICY_TYPE ]

#
# Primary Rules
#
rule iam_user_no_policies_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %iam_users not empty {
    check_user(%iam_users.Properties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

rule iam_user_no_policies_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %iam_policies not empty {
    check_policy(%iam_policies.Properties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

rule iam_user_no_policies_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %iam_managed_policies not empty {
```

```
    check_policy(%iam_managed_policies.Properties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

rule iam_user_no_policies_check when is_cfn_hook(%INPUT_DOCUMENT, %IAM_USER_TYPE) {
    check_user(%INPUT_DOCUMENT.%IAM_USER_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

rule iam_user_no_policies_check when is_cfn_hook(%INPUT_DOCUMENT, %IAM_POLICY_TYPE) {
    check_policy(%INPUT_DOCUMENT.%IAM_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

rule iam_user_no_policies_check when is_cfn_hook(%INPUT_DOCUMENT,
 %IAM_MANAGED_POLICY_TYPE) {
    check_policy(%INPUT_DOCUMENT.%IAM_MANAGED_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.4]: Require that an AWS Identity and Access Management (IAM) user
 does not have an inline or managed policy attached attached
            [FIX]: Configure IAM users to inherit permissions from IAM groups.
        >>
}

#
# Parameterized Rules
#
rule check_user(iam_user) {
    %iam_user {
        # Scenario 2 and 4
        Policies not exists or
        exists_and_is_empty_list(Policies)
```

```
        ManagedPolicyArns not exists or
        exists_and_is_empty_list(ManagedPolicyArns)
    }
}

rule check_policy(policy) {
    %policy {
        # Scenario 3 and 4
        Users not exists or
        exists_and_is_empty_list(Users)
    }
}

rule exists_and_is_empty_list(list_property) {
    %list_property {
        this is_list
        this empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.IAM.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMUser:
    Type: AWS::IAM::User
    Properties: {}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMUser:
    Type: AWS::IAM::User
    Properties:
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AdministratorAccess
```

[CT.IAM.PR.5] Require that AWS Identity and Access Management (IAM) inline policies do not have wildcard service actions

This control checks whether AWS Identity and Access Management (IAM) inline policies do not include `Effect`: `Allow` with `Action`: `Service:*` (e.g. s3:*) for individual AWS services or use the combination of `NotAction` with an `Effect` of `Allow`.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::IAM::Policy`, `AWS::IAM::Role`, `AWS::IAM::User`, `AWS::IAM::Group`
- **CloudFormation guard rule:** CT.IAM.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.IAM.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.IAM.PR.5 example templates

Explanation

When you assign permissions to AWS services, it is important to scope the allowed IAM actions in your IAM policies. You should restrict IAM actions to only those actions that are needed. This helps you to provision least privilege permissions. Overly permissive policies might lead to privilege escalation if the policies are attached to an IAM principal that might not require the permission.

Usage considerations

- This control only applies to IAM inline policies with statements that contain an `Effect` of `Allow` with an `Action` or `NotAction` element present
- This control only applies to IAM role, user or group resources with one or more inline policies and IAM policy resources with one or more statements configured

Remediation for rule failure

Remove statements from IAM inline policies with `Effect`: `Allow` and `Action`: `service:*` or `Effect`: `Allow` and `NotAction`.

The examples that follow show how to implement this remediation.

IAM Inline Policy - Example One

IAM role configured with an inline policy allowing the S3 `ListBucket` action. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
```

```
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "AWS": {
                                "Ref": "AWS::AccountId"
                            }
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Policies": [
                {
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:ListBucket"
                                ],
                                "Resource": [
                                    "*"
                                ]
                            }
                        ]
                    },
                    "PolicyName": "sample-policy"
                }
            ]
        }
    }
}
```

YAML example

```
IAMRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Principal:
            AWS: !Ref 'AWS::AccountId'
          Action:
            - sts:AssumeRole
    Policies:
      - PolicyDocument:
          Version: '2012-10-17       '
          Statement:
            - Effect: Allow
              Action:
                - s3:ListBucket
              Resource:
                - '*'
        PolicyName: sample-policy
```

The examples that follow show how to implement this remediation.

IAM Inline Policy - Example Two

IAM user configured with an inline policy allowing the S3 `ListBucket` action. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMUser": {
        "Type": "AWS::IAM::User",
        "Properties": {
            "Policies": [
                {
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
```

```
                                "Effect": "Allow",
                                "Action": [
                                    "s3:ListBucket"
                                ],
                                "Resource": [
                                    "*"
                                ]
                            }
                        ]
                    },
                    "PolicyName": "sample-policy"
                }
            ]
        }
    }
}
```

YAML example

```
IAMUser:
  Type: AWS::IAM::User
  Properties:
    Policies:
      - PolicyDocument:
          Version: '2012-10-17       '
          Statement:
            - Effect: Allow
              Action:
                - s3:ListBucket
              Resource:
                - '*'
        PolicyName: sample-policy
```

The examples that follow show how to implement this remediation.

IAM Inline Policy - Example Three

IAM group configured with an inline policy allowing the S3 `ListBucket` action. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMGroup": {
        "Type": "AWS::IAM::Group",
        "Properties": {
            "Policies": [
                {
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:ListBucket"
                                ],
                                "Resource": [
                                    "*"
                                ]
                            }
                        ]
                    },
                    "PolicyName": "sample-policy"
                }
            ]
        }
    }
}
```

YAML example

```
IAMGroup:
  Type: AWS::IAM::Group
  Properties:
    Policies:
      - PolicyDocument:
          Version: '2012-10-17       '
          Statement:
            - Effect: Allow
              Action:
```

```
                - s3:ListBucket
              Resource:
                - '*'
        PolicyName: sample-policy
```

The examples that follow show how to implement this remediation.

IAM Inline Policy - Example Four

IAM policy associated with an IAM role as an inline policy and configured to allow the S3 `ListBucket` action. The example is shown in JSON and in YAML.

JSON example

```
{
    "IAMPolicy": {
        "Type": "AWS::IAM::Policy",
        "Properties": {
            "PolicyName": "sample-policy",
            "Roles": [
                {
                    "Ref": "IAMRole"
                }
            ],
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "s3:ListBucket"
                        ],
                        "Resource": [
                            "*"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML example

```
IAMPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: sample-policy
    Roles:
      - !Ref 'IAMRole'
    PolicyDocument:
      Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:ListBucket
          Resource:
            - '*'
```

CT.IAM.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Name:
#   iam_inline_policy_no_statements_with_full_access_check
#
# Description:
#   This control checks whether AWS Identity and Access Management (IAM) inline
 policies do not include "Effect": "Allow" with "Action": "Service:*" (e.g. s3:*) for
 individual AWS services or use the combination of "NotAction" with an "Effect" of
 "Allow".
#
# Reports on:
#   AWS::IAM::Policy, AWS::IAM::Role, AWS::IAM::User, AWS::IAM::Group
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any IAM policy, IAM role, IAM user or
 IAM group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has no statements with 'Effect' set to 'Allow'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resource
#       And: 'Policies' is not provided or is an empty list
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resource
#       And: 'Policies' is provided as a non-empty list
#       And: All IAM policy documents in 'Policies' have no statements with 'Effect'
 set to 'Allow'
#      Then: SKIP
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: 'Action' statement allows full access to a service ('Action' has a value
 'service:*')
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'NotAction' statements
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM group resource
#       And: 'Policies' is provided as a non-empty list
#       And: IAM policy document in 'Policies' has a statement with 'Effect' set to
 'Allow'
```

```
#       And: The policy has one or more 'Action' statements
#       And: 'Action' statement allows full access to a service ('Action' has a value
 'service:*')
#      Then: FAIL
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM user resource
#       And: 'Policies' is provided as a non-empty list
#       And: At least one IAM policy document in 'Policies' has a statement with
 'Effect' set to 'Allow'
#       And: The policy has one or more 'NotAction' statements
#      Then: FAIL
#   Scenario: 9
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM policy resource
#       And: The policy has a statement with 'Effect' set to 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: No 'Action' statements allow full access to a service ('Action' does not
 have a value 'service:*')
#      Then: PASS
#   Scenario: 10
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an IAM role, IAM user or IAM user resource
#       And: 'Policies' is provided as a non-empty list
#       And: At least one IAM policy document in 'Policies' has a statement with
 'Effect' set to 'Allow'
#       And: The policy has one or more 'Action' statements
#       And: No 'Action' statements allow full access to a service ('Action' does not
 have a value 'service:*')
#      Then: PASS

#
# Constants
#
let AWS_IAM_POLICY_TYPE = "AWS::IAM::Policy"
let AWS_IAM_ROLE_TYPE = "AWS::IAM::Role"
let AWS_IAM_USER_TYPE = "AWS::IAM::User"
let AWS_IAM_GROUP_TYPE = "AWS::IAM::Group"
let WILDCARD_ACTION_PATTERN = /^[\w]*[:]*\*$/
let INPUT_DOCUMENT = this

#
# Assignments
#
```

```
let iam_policies = Resources.*[ Type == %AWS_IAM_POLICY_TYPE ]
let iam_principals = Resources.*[
    Type == %AWS_IAM_ROLE_TYPE or
    Type == %AWS_IAM_USER_TYPE or
    Type == %AWS_IAM_GROUP_TYPE
]

#
# Primary Rules
#
rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                                %iam_policies not
 empty {
    check_policy(%iam_policies.Properties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_POLICY_TYPE) {
    check_policy(%INPUT_DOCUMENT.%AWS_IAM_POLICY_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                                %iam_principals not
 empty {
    check_principal(%iam_principals.Properties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
```

```
        >>
}

rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_ROLE_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_ROLE_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_USER_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_USER_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}

rule iam_inline_policy_no_statements_with_full_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %AWS_IAM_GROUP_TYPE) {
    check_principal(%INPUT_DOCUMENT.%AWS_IAM_GROUP_TYPE.resourceProperties)
        <<
        [CT.IAM.PR.5]: Require that AWS Identity and Access Management (IAM) inline
 policies do not have wildcard service actions
            [FIX]: Remove statements from IAM inline policies with "Effect": "Allow"
 and "Action": "service:*" or "Effect": "Allow" and "NotAction".
        >>
}



#
# Parameterized Rules
#
rule check_policy(policy) {
    %policy [
        filter_policy_document_with_statement_provided(this)
    ] {
```

```
        PolicyDocument {
            check_statement_no_wildcard_actions(Statement)
            check_statement_no_not_action(Statement)
       }
    }
}

rule check_principal(iam_principal) {
    %iam_principal [
        filter_iam_principal_with_inline_policy_provided(this)
    ] {
        Policies[*] {
            check_policy(this)
       }
    }
}

rule check_statement_no_wildcard_actions(statement) {
    %statement [
        filter_allow_on_action(this)
    ] {
        Action exists
        check_no_wildcard_action(Action)
    }
}

rule check_statement_no_not_action(statement) {
    %statement [
        filter_allow(this)
    ] {
        NotAction not exists
    }
}

rule filter_allow_on_action(statement) {
    %statement {
        Effect == "Allow"
        Action exists
    }
}

rule filter_allow(statement) {
    %statement {
        Effect == "Allow"
```

```
    }
}

rule filter_policy_document_with_statement_provided(policy) {
    %policy {
        PolicyDocument exists
        PolicyDocument is_struct
        PolicyDocument {
            Statement exists
            filter_statement_non_empty_list(Statement) or
            Statement is_struct
        }
    }
}

rule filter_iam_principal_with_inline_policy_provided(iam_principal) {
    %iam_principal {
        Policies exists
        Policies is_list
        Policies not empty
    }
}

rule filter_statement_non_empty_list(statement) {
    %statement {
        this is_list
        this not empty
    }
}

rule check_no_wildcard_action(actions) {
    %actions[*] {
        this != %WILDCARD_ACTION_PATTERN
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
```

```
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.IAM.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-example
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action:
          - s3:ListBucket
          Resource:
```

```
          - '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
          - sts:AssumeRole
  IAMPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-example
      Roles:
      - Ref: IAMRole
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Action: s3:*
          Resource: '*'
```

AWS Key Management Service (AWS KMS) controls

Topics

- [CT.KMS.PR.1] Require any AWS KMS key to have rotation configured

- [CT.KMS.PR.2] Require that an AWS Key Management Service asymmetric key with RSA key material used for encryption has a key length greater than 2048 bits
- [CT.KMS.PR.3] Require an AWS Key Management Service key policy to have a statement that limits creation of AWS KMS grants to AWS services

[CT.KMS.PR.1] Require any AWS KMS key to have rotation configured

This control checks whether key rotation is enabled for AWS KMS customer managed keys.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::KMS::Key`
- **CloudFormation guard rule:** CT.KMS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.KMS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.KMS.PR.1 example templates

Explanation

Key rotation minimizes the possibility of key exposure to malicious users. Cryptographic best practices discourage extensive reuse of encryption keys. Rotation of keys on regular basis helps you meet organizational security and compliance requirements.

Usage considerations

- This control applies only to AWS KMS symmetric-encryption, customer managed keys.

Remediation for rule failure

Set `EnableKeyRotation` to `true` for AWS KMS symmetric-encryption keys.

The examples that follow show how to implement this remediation.

AWS KMS key - Example

AWS KMS customer managed key configured with key rotation activated. The example is shown in JSON and in YAML.

JSON example

```
{
    "KMSKey": {
        "Type": "AWS::KMS::Key",
        "Properties": {
            "PendingWindowInDays": 7,
            "KeyPolicy": {
                "Version": "2012-10-17",
                "Id": "sample-policy",
                "Statement": [
                    {
                        "Sid": "Enable IAM User Permissions",
                        "Effect": "Allow",
                        "Principal": {
                            "AWS": {
                                "Fn::Sub": "arn:${AWS::Partition}:iam::
${AWS::AccountId}:root"
                            }
                        },
                        "Action": "kms:*",
                        "Resource": "*"
                    }
                ]
            },
            "EnableKeyRotation": true
        }
    }
}
```

YAML example

```
KMSKey:
```

```
  Type: AWS::KMS::Key
  Properties:
    PendingWindowInDays: 7
    KeyPolicy:
      Version: 2012-10-17
      Id: sample-policy
      Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS: !Sub 'arn:${AWS::Partition}:iam::${AWS::AccountId}:root'
          Action: kms:*
          Resource: '*'
    EnableKeyRotation: true
```

CT.KMS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   kms_key_rotation_enabled_check
#
# Description:
#   This control checks whether key rotation is enabled for AWS KMS customer managed
 keys.
#
# Reports on:
#   AWS::KMS::Key
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any KMS key resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeySpec' is provided and is a value other than 'SYMMETRIC_DEFAULT'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeySpec' is not provided or is provided and is set to 'SYMMETRIC_DEFAULT'
#       And: 'EnableKeyRotation' is not provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeySpec' is not provided or is provided and is set to 'SYMMETRIC_DEFAULT'
#       And: 'EnableKeyRotation' is provided and is set to bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeySpec' is not provided or is provided and is set to 'SYMMETRIC_DEFAULT'
#       And: 'EnableKeyRotation' is provided and is set to bool(true)
#      Then: PASS

#
# Constants
#
let KMS_KEY_TYPE = "AWS::KMS::Key"
let INPUT_DOCUMENT = this

#
# Assignments
#
let kms_keys = Resources.*[ Type == %KMS_KEY_TYPE ]

#
# Primary Rules
#
rule kms_key_rotation_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %kms_keys not empty {
    check(%kms_keys.Properties)
        <<
```

```
        [CT.KMS.PR.1]: Require any AWS KMS key to have rotation configured
        [FIX]: Set 'EnableKeyRotation' to 'true' for AWS KMS symmetric-encryption keys.
        >>
}

rule kms_key_rotation_enabled_check when is_cfn_hook(%INPUT_DOCUMENT, %KMS_KEY_TYPE) {
    check(%INPUT_DOCUMENT.%KMS_KEY_TYPE.resourceProperties)
        <<
        [CT.KMS.PR.1]: Require any AWS KMS key to have rotation configured
        [FIX]: Set 'EnableKeyRotation' to 'true' for AWS KMS symmetric-encryption keys.
        >>
}

#
# Parameterized Rules
#
rule check(kms_keys) {
    %kms_keys[
        # Scenario 2
        filter_is_kms_cmk_symmetric_key(this)
    ] {
        # Scenario 3, 4 and 5
        EnableKeyRotation exists
        EnableKeyRotation == true
    }
}

rule filter_is_kms_cmk_symmetric_key(kms_key) {
    %kms_key {
        KeySpec not exists or
        KeySpec == "SYMMETRIC_DEFAULT"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.KMS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
      PendingWindowInDays: 7
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      EnableKeyRotation: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
```

```
      PendingWindowInDays: 7
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      EnableKeyRotation: false
```

[CT.KMS.PR.2] Require that an AWS Key Management Service asymmetric key with RSA key material used for encryption has a key length greater than 2048 bits

This control checks whether an AWS KMS asymmetric key with RSA key material, which is used for encryption and decryption, to use a key spec with a key length greater than 2048 bits (that is, a key spec other than `RSA_2048`).

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::KMS::Key`
- **CloudFormation guard rule:** CT.KMS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.KMS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.KMS.PR.2 example templates

Explanation

AWS Control Tower recommends using an RSA key spec with a key length greater than 2048 bits, when you are using such keys for encryption and decryption. The key spec determines whether the KMS key is symmetric or asymmetric. It also determines the type of key material, and the algorithms it supports. AWS KMS supports asymmetric KMS keys that represent a mathematically-related RSA or elliptic curve (ECC) public and private key pair. A KMS key with an RSA key pair can be used for encryption and decryption, or for signing and verification (but not both). AWS KMS supports several key lengths for different security requirements.

Usage considerations

- This control applies only to a KMS key with an RSA key spec, which is configured for encryption and decryption.

Remediation for rule failure

For KMS keys with an RSA keyspec, which are configured for encryption and decryption (`KeyUsage` of `ENCRYPT_DECRYPT`), set the `KeySpec` parameter to a key spec other than `RSA_2048`.

The examples that follow show how to implement this remediation.

AWS KMS key - Example

An AWS KMS asymmetric key configured for encryption and decryption, with an `RSA_4096` key spec. The example is shown in JSON and in YAML.

JSON example

```
{
    "Key": {
        "Type": "AWS::KMS::Key",
        "Properties": {
            "KeyPolicy": {
                "Version": "2012-10-17",
                "Id": "example-policy",
                "Statement": [
                    {
                        "Sid": "Enable IAM User Permissions",
                        "Effect": "Allow",
                        "Principal": {
```

```
                            "AWS": {
                                "Fn::Sub": "arn:${AWS::Partition}:iam::
${AWS::AccountId}:root"
                            }
                        },
                        "Action": "kms:*",
                        "Resource": "*"
                    }
                ]
            },
            "KeyUsage": "ENCRYPT_DECRYPT",
            "KeySpec": "RSA_4096"
        }
    }
}
```

YAML example

```
Key:
  Type: AWS::KMS::Key
  Properties:
    KeyPolicy:
      Version: '2012-10-17       '
      Id: example-policy
      Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS: !Sub 'arn:${AWS::Partition}:iam::${AWS::AccountId}:root'
          Action: kms:*
          Resource: '*'
    KeyUsage: ENCRYPT_DECRYPT
    KeySpec: RSA_4096
```

CT.KMS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Name:
#   kms_asymmetric_rsa_keyspec_check
#
# Description:
#   This control checks whether an AWS KMS asymmetric key with RSA key material, which
 is used for encryption and decryption, to use a key spec with a key length greater
 than 2048 bits (that is, a key spec other than 'RSA_2048').
#
# Reports on:
#   AWS::KMS::Key
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any KMS key resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeyUsage' has been provided and is a value other than 'ENCRYPT_DECRYPT'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeyUsage' has not been provided or has been provided and is set to
 'ENCRYPT_DECRYPT'
#       And: 'KeySpec' has not been provided or has been provided and is set to a
#            key spec other than an RSA key spec (does not begin with 'RSA_')
#      Then: SKIP
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeyUsage' has not been provided or has been provided and is set to
 'ENCRYPT_DECRYPT'
#       And: 'KeySpec' has been provided and is set to an RSA key spec (begins with
 'RSA_')
```

```
#       And: 'KeySpec' has been set to a disallowed RSA key spec ('RSA_2048')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'KeyUsage' has not been provided or has been provided and is set to
 'ENCRYPT_DECRYPT'
#       And: 'KeySpec' has been provided and is set to an RSA key spec (begins with
 'RSA_')
#       And: 'KeySpec' has not been set to a disallowed RSA key spec ('RSA_2048')
#      Then: PASS

#
# Constants
#
let KMS_KEY_TYPE = "AWS::KMS::Key"
let RSA_KEYSPEC_PATTERN = /^RSA_/
let ENCRYPTION_KEY_USAGE = "ENCRYPT_DECRYPT"
let DISALLOWED_RSA_KEYSPECS = [ "RSA_2048" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let kms_keys = Resources.*[ Type == %KMS_KEY_TYPE ]

#
# Primary Rules
#
rule kms_asymmetric_rsa_keyspec_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %kms_keys not empty {
    check(%kms_keys.Properties)
        <<
        [CT.KMS.PR.2]: Require that an AWS KMS asymmetric key with RSA key material
 used for encryption has a key length greater than 2048 bits
        [FIX]: For KMS keys with an RSA keyspec, which are configured for encryption
 and decryption ('KeyUsage' of 'ENCRYPT_DECRYPT'), set the 'KeySpec' parameter to a key
 spec other than 'RSA_2048'.
        >>
}

rule kms_asymmetric_rsa_keyspec_check when is_cfn_hook(%INPUT_DOCUMENT, %KMS_KEY_TYPE)
 {
    check(%INPUT_DOCUMENT.%KMS_KEY_TYPE.resourceProperties)
```

```
        <<
        [CT.KMS.PR.2]: Require that an AWS KMS asymmetric key with RSA key material
 used for encryption has a key length greater than 2048 bits
        [FIX]: For KMS keys with an RSA keyspec, which are configured for encryption
 and decryption ('KeyUsage' of 'ENCRYPT_DECRYPT'), set the 'KeySpec' parameter to a key
 spec other than 'RSA_2048'.
        >>
}

#
# Parameterized Rules
#
rule check(kms_keys) {
    %kms_keys[
        # Scenarios 2 and 3
        filter_is_kms_rsa_asymmetric_encryption_key(this)
    ] {
        # Scenario 4 and 5
        KeySpec exists
        KeySpec not in %DISALLOWED_RSA_KEYSPECS
    }
}

rule filter_is_kms_rsa_asymmetric_encryption_key(kms_key) {
    %kms_key {
        KeyUsage not exists or
        KeyUsage == %ENCRYPTION_KEY_USAGE

        KeySpec exists
        KeySpec == %RSA_KEYSPEC_PATTERN
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.KMS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: '2012-10-17       '
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeyUsage: ENCRYPT_DECRYPT
      KeySpec: RSA_4096
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
```

```
        Version: '2012-10-17       '
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeyUsage: ENCRYPT_DECRYPT
      KeySpec: RSA_2048
```

[CT.KMS.PR.3] Require an AWS Key Management Service key policy to have a statement that limits creation of AWS KMS grants to AWS services

This control checks whether an AWS KMS key has an associated key policy statement that limits creation of AWS KMS grants to AWS services only.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::KMS::Key`
- **CloudFormation guard rule:** CT.KMS.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.KMS.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.KMS.PR.3 example templates

Explanation

Users with permission to create grants for a KMS key (`kms:CreateGrant`) can use a grant to allow other users and roles, including AWS services, to use the KMS key (grantee principals). Grantee

principals can be identities in your own AWS account, or identities from a different account or organization.

By denying creation of AWS KMS grants unless the request originates from an AWS service, you prevent grants from being assigned directly to principals other than AWS service principals, and you reduce the opportunities for grant misuse. The `kms:GrantIsForAWSResource` condition helps check whether the `CreateGrant` operation is being called by an AWS service integrated with AWS KMS, on behalf of another principal. The `aws:PrincipalIsAWSService` condition helps check whether the `CreateGrant` operation is being called directly by an AWS service principal.

Usage considerations

- If you must use additional conditions on your grants, or if you must issue AWS KMS grants directly to your IAM principals for a customer-managed key, **do not** enable this control. This control requires a policy statement that denies the creation of AWS KMS grants for your customer-managed KMS keys, if the request does not originate from an AWS service that's integrated with AWS KMS, or from an AWS service principal.

Remediation for rule failure

Configure an AWS KMS policy statement that denies access to the `kms:CreateGrant` operation for all principals when the `kms:GrantIsForAWSResource` and `aws:PrincipalIsAWSService` conditions are both false, using the `BoolIfExists` condition operator.

The examples that follow show how to implement this remediation.

AWS KMS key - Example

An AWS KMS key, configured to deny creation of AWS KMS grants where the `CreateGrant` request does not originate from an AWS service principal. The example is shown in JSON and in YAML.

JSON example

```
{
    "Key": {
        "Type": "AWS::KMS::Key",
        "Properties": {
            "KeyUsage": "ENCRYPT_DECRYPT",
```

```
            "KeySpec": "SYMMETRIC_DEFAULT",
            "KeyPolicy": {
                "Version": "2012-10-17",
                "Id": "sample-policy",
                "Statement": [
                    {
                        "Sid": "Enable IAM User Permissions",
                        "Effect": "Allow",
                        "Principal": {
                            "AWS": {
                                "Fn::Sub": "arn:${AWS::Partition}:iam::
${AWS::AccountId}:root"
                            }
                        },
                        "Action": "kms:*",
                        "Resource": "*"
                    },
                    {
                        "Effect": "Deny",
                        "Action": "kms:CreateGrant",
                        "Resource": "*",
                        "Principal": "*",
                        "Condition": {
                            "BoolIfExists": {
                                "kms:GrantIsForAWSResource": "false",
                                "aws:PrincipalIsAWSService": "false"
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
Key:
  Type: AWS::KMS::Key
  Properties:
    KeyUsage: ENCRYPT_DECRYPT
```

```
    KeySpec: SYMMETRIC_DEFAULT
    KeyPolicy:
      Version: '2012-10-17       '
      Id: sample-policy
      Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS: !Sub 'arn:${AWS::Partition}:iam::${AWS::AccountId}:root'
          Action: kms:*
          Resource: '*'
        - Effect: Deny
          Action: kms:CreateGrant
          Resource: '*'
          Principal: '*'
          Condition:
            BoolIfExists:
              kms:GrantIsForAWSResource: 'false'
              aws:PrincipalIsAWSService: 'false'
```

CT.KMS.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Name:
#   kms_create_grant_aws_service_check
#
# Description:
#   This control checks whether an AWS KMS key has an associated key policy statement
 that limits creation of AWS KMS grants to AWS services only.
#
# Reports on:
#   AWS::KMS::Key
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any KMS key resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'Statement' in 'KeyPolicy' has not been provided or has been provided as
 an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'Statement' in 'KeyPolicy' has been provided as a non-empty list
#       And: 'Statement' in 'KeyPolicy' does not include a statement that denies all
 Principals ('*', AWS: '*')
#            create grant permissions ('kms:CreateGrant') on the KMS key (resource of
 '*')
#            when the conditions 'kms:GrantIsForAWSResource' and
 'aws:PrincipalIsAWSService'
#            are both 'false' ('BoolIfExists' condition operator)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a KMS key resource
#       And: 'Statement' in 'KeyPolicy' has been provided as a non-empty list
#       And: 'Statement' in 'KeyPolicy' includes a statement that denies all Principals
 ('*', AWS: '*')
#            create grant permissions ('kms:CreateGrant') on the KMS key (resource of
 '*')
#            when the conditions 'kms:GrantIsForAWSResource' and
 'aws:PrincipalIsAWSService'
#            are both 'false' ('BoolIfExists' condition operator)
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let KMS_KEY_TYPE = "AWS::KMS::Key"
let KMS_GRANT_IS_FOR_AWS_RESOURCE_KEY_PATTERN = /^(?i)kms:GrantIsForAWSResource$/
```

```
let AWS_PRINCIPAL_IS_AWS_SERVICE_KEY_PATTERN = /^(?i)aws:PrincipalIsAWSService$/
let ALLOWED_KEY_PATTERNS = [/^(?i)kms:GrantIsForAWSResource$/, /^(?
i)aws:PrincipalIsAWSService$/]
#
# Assignments
#
let kms_keys = Resources.*[ Type == %KMS_KEY_TYPE ]

#
# Primary Rules
#
rule kms_create_grant_aws_service_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %kms_keys not empty {
    check(%kms_keys.Properties)
        <<
        [CT.KMS.PR.3]: Require an AWS KMS key policy to have a statement that limits
 creation of AWS KMS grants to AWS services
        [FIX]: Configure a KMS keys policy statement that denies access to the
 'kms:CreateGrant' operation for all principals when the 'kms:GrantIsForAWSResource'
 and 'aws:PrincipalIsAWSService' conditions are both false, using the 'BoolIfExists'
 condition operator.
        >>
}

rule kms_create_grant_aws_service_check when is_cfn_hook(%INPUT_DOCUMENT,
 %KMS_KEY_TYPE) {
    check(%INPUT_DOCUMENT.%KMS_KEY_TYPE.resourceProperties)
        <<
        [CT.KMS.PR.3]: Require an AWS KMS key policy to have a statement that limits
 creation of AWS KMS grants to AWS services
        [FIX]: Configure a KMS key policy statement that denies access to the
 'kms:CreateGrant' operation for all principals when the 'kms:GrantIsForAWSResource'
 and 'aws:PrincipalIsAWSService' conditions are both false, using the 'BoolIfExists'
 condition operator.
        >>
}

#
# Parameterized Rules
#
rule check(kms_keys) {
    %kms_keys {
        # Scenario 2
        KeyPolicy exists
```

```
        KeyPolicy is_struct

        KeyPolicy {
            Statement exists
            Statement is_list
            Statement not empty

            #Scenario 3 and 4
            some Statement[*] {
                check_statement_create_grant_aws_services_only(this)
            }
        }
    }
}

rule check_statement_create_grant_aws_services_only(statement) {
    %statement{
        check_all_required_statement_properties(this)

        Effect == "Deny"
        Action[*] in ["kms:CreateGrant"]

        Principal == "*" or
        Principal {
            AWS exists
            AWS == "*"
        }

        Resource[*] == "*"

        Condition is_struct
        struct_contains_only_allowed_keys(Condition, ["BoolIfExists"])

        Condition {
            BoolIfExists exists
            BoolIfExists is_struct

            struct_contains_only_allowed_keys(BoolIfExists, %ALLOWED_KEY_PATTERNS)
            struct_contains_key_with_value(BoolIfExists,
 %KMS_GRANT_IS_FOR_AWS_RESOURCE_KEY_PATTERN, "false")
            struct_contains_key_with_value(BoolIfExists,
 %AWS_PRINCIPAL_IS_AWS_SERVICE_KEY_PATTERN, "false")
        }
    }
```

```
}

rule check_all_required_statement_properties(statement) {
    %statement {
        Effect exists
        Action exists
        Principal exists
        Condition exists
        Resource exists
    }
}

rule struct_contains_only_allowed_keys(struct, allowed_keys) {
    let disallowed_keys = %struct[
        keys not in %allowed_keys
    ]
    %disallowed_keys empty
}

rule struct_contains_key_with_value(struct, key, value) {
    let key_present = %struct[
        keys == %key
    ]
    %key_present not empty
    %key_present == %value
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.KMS.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
    Properties:
      KeyUsage: ENCRYPT_DECRYPT
      KeySpec: SYMMETRIC_DEFAULT
      KeyPolicy:
        Version: '2012-10-17       '
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
        - Effect: Deny
          Action: kms:CreateGrant
          Resource: '*'
          Principal: '*'
          Condition:
            BoolIfExists:
              kms:GrantIsForAWSResource: 'false'
              aws:PrincipalIsAWSService: 'false'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Key:
    Type: AWS::KMS::Key
```

```
    Properties:
      KeyUsage: ENCRYPT_DECRYPT
      KeySpec: SYMMETRIC_DEFAULT
      KeyPolicy:
        Version: '2012-10-17       '
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
```

Amazon Kinesis controls

Topics

- [CT.KINESIS.PR.1] Require any Amazon Kinesis data stream to have encryption at rest configured

[CT.KINESIS.PR.1] Require any Amazon Kinesis data stream to have encryption at rest configured

This control checks whether Amazon Kinesis data streams are encrypted at rest with server-side encryption.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Kinesis::Stream`
- **CloudFormation guard rule:** CT.KINESIS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.KINESIS.PR.1 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.KINESIS.PR.1 example templates

Explanation

Server-side encryption is a feature in Amazon Kinesis data streams that encrypts data automatically, before the data is at rest, by using an AWS KMS key. Data is encrypted before it is written to the Kinesis stream storage layer, and decrypted after it is retrieved from storage. As a result, your data is encrypted at rest within the Amazon Kinesis data stream service.

Remediation for rule failure

Specify a `StreamEncryption` configuration, with `EncryptionType` set to `KMS` and `KeyId` set to an AWS KMS key identifier.

The examples that follow show how to implement this remediation.

Amazon Kinesis Data Stream - Example

Amazon Kinesis data stream configured to encrypt data at rest with server-side encryption, using an AWS KMS key. The example is shown in JSON and in YAML.

JSON example

```
{
    "KinesisStream": {
        "Type": "AWS::Kinesis::Stream",
        "Properties": {
            "RetentionPeriodHours": 168,
            "ShardCount": 3,
            "StreamEncryption": {
                "EncryptionType": "KMS",
                "KeyId": {
                    "Ref": "KMSKey"
                }
            }
        }
    }
}
```

YAML example

```
KinesisStream:
  Type: AWS::Kinesis::Stream
  Properties:
    RetentionPeriodHours: 168
    ShardCount: 3
    StreamEncryption:
      EncryptionType: KMS
      KeyId: !Ref 'KMSKey'
```

CT.KINESIS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   kinesis_stream_encrypted_check
#
# Description:
#   This control checks whether Amazon Kinesis data streams are encrypted at rest with
 server-side encryption.
#
# Reports on:
#   AWS::Kinesis::Stream
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Kinesis stream resources
#      Then: SKIP
#   Scenario: 2
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Kinesis stream resource
#       And: 'StreamEncryption' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Kinesis stream resource
#       And: 'StreamEncryption' has been provided
#       And: 'StreamEncryption.EncryptionType' has not been provided or provided as an
 empty string
#       And: 'StreamEncryption.KeyId' has not been provided or provided as an empty
 string or invalid local reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Kinesis stream resource
#       And: 'StreamEncryption' has been provided
#       And: 'StreamEncryption.EncryptionType' has been provided as a non-empty string
#       And: 'StreamEncryption.KeyId' has not been provided or provided as an empty
 string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Kinesis stream resource
#       And: 'StreamEncryption' has been provided
#       And: 'StreamEncryption.EncryptionType' has not been provided or provided as an
 empty string
#       And: 'StreamEncryption.KeyId' has been provided as a non-empty string or valid
 local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Kinesis stream resource
#       And: 'StreamEncryption' has been provided
#       And: 'StreamEncryption.EncryptionType' has been provided as a non-empty string
#       And: 'StreamEncryption.KeyId' has been provided as a non-empty string or valid
 local reference
#      Then: PASS

#
# Constants
#
let KINESIS_STREAM_TYPE = "AWS::Kinesis::Stream"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let kinesis_streams = Resources.*[ Type == %KINESIS_STREAM_TYPE ]

#
# Primary Rules
#
rule kinesis_stream_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %kinesis_streams not empty {
    check(%kinesis_streams.Properties)
        <<
        [CT.KINESIS.PR.1]: Require any Amazon Kinesis data stream to have encryption at
 rest configured
            [FIX]: Specify a 'StreamEncryption' configuration, with 'EncryptionType'
 set to 'KMS' and 'KeyId' set to an AWS KMS key identifier.
        >>
}

rule kinesis_stream_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %KINESIS_STREAM_TYPE) {
    check(%INPUT_DOCUMENT.%KINESIS_STREAM_TYPE.resourceProperties)
        <<
        [CT.KINESIS.PR.1]: Require any Amazon Kinesis data stream to have encryption at
 rest configured
            [FIX]: Specify a 'StreamEncryption' configuration, with 'EncryptionType'
 set to 'KMS' and 'KeyId' set to an AWS KMS key identifier.
        >>
}

#
# Parameterized Rules
#
rule check(kinesis_stream) {
    %kinesis_stream {
        # Scenario 2
        StreamEncryption exists
        StreamEncryption is_struct

        StreamEncryption {
            # Scenario 3
            EncryptionType exists
            KeyId exists
```

```
            # Scenario 4, 5 and 6
            check_is_string_and_not_empty(EncryptionType)
            check_is_string_and_not_empty(KeyId) or
            check_local_references(%INPUT_DOCUMENT, KeyId, "AWS::KMS::Key") or
            check_local_references(%INPUT_DOCUMENT, KeyId, "AWS::KMS::Alias")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
```

```
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.KINESIS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      PendingWindowInDays: 7
      KeyPolicy:
        Version: 2012-10-17
        Id: example-key-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeySpec: SYMMETRIC_DEFAULT
  KinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      RetentionPeriodHours: 168
      ShardCount: 3
      StreamEncryption:
        EncryptionType: KMS
        KeyId:
          Ref: KMSKey
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  KinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      RetentionPeriodHours: 168
      ShardCount: 3
```

AWS Lambda controls

Topics

- [CT.LAMBDA.PR.2] Require AWS Lambda function policies to prohibit public access
- [CT.LAMBDA.PR.3] Require an AWS Lambda function to be in a customer-managed Amazon Virtual Private Cloud (VPC)
- [CT.LAMBDA.PR.4] Require an AWS Lambda layer permission to grant access to an AWS organization or specific AWS account
- [CT.LAMBDA.PR.5] Require an AWS Lambda function URL to use AWS IAM-based authentication
- [CT.LAMBDA.PR.6] Require an AWS Lambda function URL CORS policy to restrict access to specific origins

[CT.LAMBDA.PR.2] Require AWS Lambda function policies to prohibit public access

This control checks whether an AWS Lambda function resource-based policy prohibits public access.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Lambda::Permission`

- **CloudFormation guard rule:** CT.LAMBDA.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.LAMBDA.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.LAMBDA.PR.2 example templates

Explanation

The Lambda function should not be publicly accessible, because it may permit unintended access to your code stored in the function.

Remediation for rule failure

When setting `Principal` to `*`, provide one of `SourceAccount`, `SourceArn`, or `PrincipalOrgID`. When setting `Principal` to a service principal (for example, s3.amazonaws.com), provide one of `SourceAccount` or `SourceArn`.

The examples that follow show how to implement this remediation.

AWS Lambda Function Policy - Example One

AWS Lambda function policy configured with an AWS account ID principal. The example is shown in JSON and in YAML.

JSON example

```
{
    "LambdaPermission": {
        "Type": "AWS::Lambda::Permission",
        "Properties": {
            "Action": "lambda:InvokeFunction",
            "FunctionName": {
                "Ref": "LambdaFunction"
            },
            "Principal": {
                "Ref": "AWS::AccountId"
            }
```

```
        }
    }
}
```

YAML example

```
LambdaPermission:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref 'LambdaFunction'
    Principal: !Ref 'AWS::AccountId'
```

The examples that follow show how to implement this remediation.

AWS Lambda Function Policy - Example Two

AWS Lambda function policy configured with a wildcard principal and source account condition. The example is shown in JSON and in YAML.

JSON example

```
{
    "LambdaPermission": {
        "Type": "AWS::Lambda::Permission",
        "Properties": {
            "Action": "lambda:InvokeFunction",
            "FunctionName": {
                "Ref": "LambdaFunction"
            },
            "Principal": "*",
            "SourceAccount": {
                "Ref": "AWS::AccountId"
            }
        }
    }
}
```

YAML example

```
LambdaPermission:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref 'LambdaFunction'
    Principal: '*'
    SourceAccount: !Ref 'AWS::AccountId'
```

The examples that follow show how to implement this remediation.

AWS Lambda Function Policy - Example Three

AWS Lambda function policy configured with a service principal and source ARN condition. The example is shown in JSON and in YAML.

JSON example

```
{
    "LambdaPermission": {
        "Type": "AWS::Lambda::Permission",
        "Properties": {
            "Action": "lambda:InvokeFunction",
            "FunctionName": {
                "Ref": "LambdaFunction"
            },
            "Principal": "s3.amazonaws.com",
            "SourceArn": {
                "Fn::GetAtt": [
                    "S3Bucket",
                    "Arn"
                ]
            }
        }
    }
}
```

YAML example

```
LambdaPermission:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref 'LambdaFunction'
    Principal: s3.amazonaws.com
    SourceArn: !GetAtt 'S3Bucket.Arn'
```

CT.LAMBDA.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   lambda_function_public_access_prohibited_check
#
# Description:
#   This control checks whether an AWS Lambda function resource-based policy prohibits
 public access.
#
# Reports on:
#   AWS::Lambda::Permission
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Lambda permission resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a Lambda permission resource
#       And: 'FunctionUrlAuthType' has been provided with a value of 'NONE'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda permission resource
#       And: 'Principal' has been provided with a wildcard value ('*')
#       And: 'SourceAccount' has not been provided or provided with an empty string
 value
#       And: 'SourceArn' has not been provided or provided with an empty string value
 or non-valid local reference
#       And: 'PrincipalOrgID' has not been provided or provided with an empty string
 value
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda permission resource
#       And: 'Principal' has been provided with value that does not match an AWS
 Account ID, AWS IAM ARN or
#            wildcard value ('*')
#       And: 'SourceAccount' has not been provided or provided with an empty string
 value
#       And: 'SourceArn' has not been provided or provided with an empty string value
 or non-valid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda permission resource
#       And: 'Principal' has been provided with an AWS Account ID or AWS IAM ARN value
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda permission resource
#       And: 'Principal' has been provided with a wildcard value ('*')
#       And: At least one of 'SourceAccount', 'SourceArn' or 'PrincipalOrgID' have been
 provided with non-empty string
#            values (or a valid local reference for 'SourceArn')
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda permission resource
#       And: 'Principal' has been provided with value that does not match an AWS
 Account ID or AWS IAM ARN
```

```
#       And: At least one of 'SourceAccount', 'SourceArn' have been provided with non-
empty string values (or a valid
#            local reference for 'SourceArn')
#      Then: PASS

#
# Constants
#
let LAMBDA_PERMISSION_TYPE = "AWS::Lambda::Permission"
let AWS_ACCOUNT_ID_PATTERN = /\d{12}/
let AWS_IAM_PRINCIPAL_PATTERN = /^arn:aws[a-z0-9\-]*:iam::\d{12}:.+/

let INPUT_DOCUMENT = this

#
# Assignments
#
let lambda_permissions = Resources.*[ Type == %LAMBDA_PERMISSION_TYPE ]

#
# Primary Rules
#
rule lambda_function_public_access_prohibited_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %lambda_permissions not empty
 {
    check(%lambda_permissions.Properties)
        <<
        [CT.LAMBDA.PR.2]: Require AWS Lambda function policies to prohibit public
 access
        [FIX]: When setting 'Principal' to '*', provide one of 'SourceAccount',
 'SourceArn', or 'PrincipalOrgID'. When setting 'Principal' to a service principal (for
 example, s3.amazonaws.com), provide one of 'SourceAccount' or 'SourceArn'.
        >>
}

rule lambda_function_public_access_prohibited_check when is_cfn_hook(%INPUT_DOCUMENT,
 %LAMBDA_PERMISSION_TYPE) {
    check(%INPUT_DOCUMENT.%LAMBDA_PERMISSION_TYPE.resourceProperties)
        <<
        [CT.LAMBDA.PR.2]: Require AWS Lambda function policies to prohibit public
 access
```

```
        [FIX]: When setting 'Principal' to '*', provide one of 'SourceAccount',
 'SourceArn', or 'PrincipalOrgID'. When setting 'Principal' to a service principal (for
 example, s3.amazonaws.com), provide one of 'SourceAccount' or 'SourceArn'.
        >>
}

#
# Parameterized Rules
#
rule check(lambda_permission) {
    %lambda_permission {
        # Scenario 2 and 5
        FunctionUrlAuthType not exists or
        FunctionUrlAuthType != "NONE"
    }

    %lambda_permission [
        Principal exists
        Principal == "*"
    ] {
        # Scenario 3 and 6
        SourceAccount exists or
        SourceArn exists or
        PrincipalOrgID exists

        check_is_string_and_not_empty(SourceAccount) or
        check_is_string_or_local_reference(SourceArn) or
        check_is_string_and_not_empty(PrincipalOrgID)
    }

    %lambda_permission [
        Principal exists
        Principal != "*"
        Principal != %AWS_ACCOUNT_ID_PATTERN
        Principal != %AWS_IAM_PRINCIPAL_PATTERN
    ] {
        # Scenario 4 and 7
        SourceAccount exists or
        SourceArn exists

        check_is_string_and_not_empty(SourceAccount) or
        check_is_string_or_local_reference(SourceArn)
    }
}
```

```
rule check_is_string_or_local_reference(value) {
    %value {
        check_is_string_and_not_empty(this) or
        check_local_references(%INPUT_DOCUMENT, this)
    }
}

rule check_local_references(doc, reference_properties) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0])
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
```

```
}
```

CT.LAMBDA.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
```

```
      Runtime: python3.9
      Code:
        ZipFile: "def handler(event, context):\n  print(\"hello\")\n"
      Description: TestS3EventFunction
  LambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:InvokeFunction
      FunctionName:
        Ref: LambdaFunction
      Principal:
        Ref: AWS::AccountId
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
```

```
          Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Runtime: python3.9
      Code:
        ZipFile: "def handler(event, context):\n  print(\"hello\")\n"
      Description: TestS3EventFunction
  LambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:InvokeFunction
      FunctionName:
        Ref: LambdaFunction
      Principal: '*'
```

[CT.LAMBDA.PR.3] Require an AWS Lambda function to be in a customer-managed Amazon Virtual Private Cloud (VPC)

This control checks whether an AWS Lambda function has been configured with access to resources in a customer-managed Amazon Virtual Private Cloud (VPC).

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Lambda::Function`
- **CloudFormation guard rule:** CT.LAMBDA.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.LAMBDA.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.LAMBDA.PR.3 example templates

Explanation

AWS Lambda functions can be linked to private subnets within a virtual private cloud (VPC) in your AWS account to connect to resources such as databases, cache instances, or internal services. Ensure that the subnets and security groups used allow access to the necessary resources.

Usage considerations

- This control does not evaluate the VPC subnet routing configuration to determine public reachability.
- This control does not support AWS Lambda@Edge Functions. Lambda@Edge does not support functions that are configured with access to resources inside your VPC.
- Lambda functions can't connect directly to a VPC with dedicated instance tenancy. To connect to resources in a dedicated VPC, peer it to a second VPC with default tenancy.

Remediation for rule failure

In `VpcConfig`, provide the `SubnetIds` property with one or more Subnet IDs, and provide the `SecurityGroupIds` property with one or more Security Group IDs.

The examples that follow show how to implement this remediation.

AWS Lambda Function - Example

AWS Lambda function configured to access resources in a VPC. The example is shown in JSON and in YAML.

JSON example

```
{
    "LambdaFunction": {
        "Type": "AWS::Lambda::Function",
        "Properties": {
            "Role": {
                "Fn::GetAtt": "LambdaFunctionRole.Arn"
            },
            "Handler": "index.handler",
            "Code": {
```

```
                "ZipFile": "def handler(event, context):\n    print(\"sample function
\")\n"
            },
            "Runtime": "python3.9",
            "VpcConfig": {
                "SubnetIds": [
                    {
                        "Fn::GetAtt": [
                            "SubnetOne",
                            "SubnetId"
                        ]
                    },
                    {
                        "Fn::GetAtt": [
                            "SubnetTwo",
                            "SubnetId"
                        ]
                    }
                ],
                "SecurityGroupIds": [
                    {
                        "Fn::GetAtt": [
                            "SecurityGroup",
                            "GroupId"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML example

```
LambdaFunction:
  Type: AWS::Lambda::Function
  Properties:
    Role: !GetAtt 'LambdaFunctionRole.Arn'
    Handler: index.handler
    Code:
      ZipFile: "def handler(event, context):\n    print(\"sample function\")\n"
```

```
    Runtime: python3.9
    VpcConfig:
      SubnetIds:
        - !GetAtt 'SubnetOne.SubnetId'
        - !GetAtt 'SubnetTwo.SubnetId'
      SecurityGroupIds:
        - !GetAtt 'SecurityGroup.GroupId'
```

CT.LAMBDA.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   lambda_inside_vpc_check
#
# Description:
#   This control checks whether an AWS Lambda function has been configured with access
 to resources in a customer-managed Amazon Virtual Private Cloud (VPC).
#
# Reports on:
#   AWS::Lambda::Function
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Lambda function resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function resource
#       And: 'VpcConfig' has not been provided
#      Then: FAIL
```

```
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function resource
#       And: 'VpcConfig' has been provided
#       And: 'SubnetIds' in 'VpcConfig' has been provided as a non-empty list that
 contains non-empty strings or valid
#            local references
#       And: 'SecurityGroupIds' in 'VpcConfig' has not been been provided or has been
 provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function resource
#       And: 'VpcConfig' has been provided
#       And: 'SecurityGroupIds' in 'VpcConfig' has been provided as a non-empty list
 that contains non-empty strings
#            or valid local references
#       And: 'SubnetIds' in 'VpcConfig' has not been been provided or has been provided
 as an empty list
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function resource
#       And: 'VpcConfig' has been provided
#       And: 'SecurityGroupIds' in 'VpcConfig' has been provided as a non-empty list
 that contains non-empty strings or
#            valid local references
#       And: 'SubnetIds' in 'VpcConfig' has been provided as a non-empty list that
 contains non-empty strings or valid
#            local references
#      Then: PASS

#
# Constants
#
let LAMBDA_FUNCTION_TYPE = "AWS::Lambda::Function"
let INPUT_DOCUMENT = this

#
# Assignments
#
let lambda_functions = Resources.*[ Type == %LAMBDA_FUNCTION_TYPE ]

#
```

```
# Primary Rules
#
rule lambda_inside_vpc_check when is_cfn_template(%INPUT_DOCUMENT)
                                 %lambda_functions not empty {
    check(%lambda_functions.Properties)
        <<
        [CT.LAMBDA.PR.3]: Require an AWS Lambda function to be in a customer-managed
 Amazon Virtual Private Cloud (VPC)
        [FIX]: In 'VpcConfig', provide the 'SubnetIds' property with one or more Subnet
 IDs, and provide the 'SecurityGroupIds' property with one or more Security Group IDs.
        >>
}

rule lambda_inside_vpc_check when is_cfn_hook(%INPUT_DOCUMENT, %LAMBDA_FUNCTION_TYPE) {
    check(%INPUT_DOCUMENT.%LAMBDA_FUNCTION_TYPE.resourceProperties)
        <<
        [CT.LAMBDA.PR.3]: Require an AWS Lambda function to be in a customer-managed
 Amazon Virtual Private Cloud (VPC)
        [FIX]: In 'VpcConfig', provide the 'SubnetIds' property with one or more Subnet
 IDs, and provide the 'SecurityGroupIds' property with one or more Security Group IDs.
        >>
}

#
# Parameterized Rules
#
rule check(lambda_function) {
    %lambda_function {
        # Scenario 2
        VpcConfig exists
        VpcConfig is_struct

        VpcConfig {
            # Scenario 3 and 5
            SubnetIds exists
            SubnetIds is_list
            SubnetIds not empty
            SubnetIds[*] {
                check_is_string_and_not_empty(this) or
                check_local_references(%INPUT_DOCUMENT, this, "AWS::EC2::Subnet")
            }
            # Scenario 4 and 5
            SecurityGroupIds exists
            SecurityGroupIds is_list
```

```
            SecurityGroupIds not empty
            SecurityGroupIds[*] {
                check_is_string_and_not_empty(this) or
                check_local_references(%INPUT_DOCUMENT, this,
 "AWS::EC2::SecurityGroup")
            }
        }
     }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
```

```
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.LAMBDA.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
```

```
  SecurityGroup1:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VPC
      GroupDescription:
        Fn::Sub: ${AWS::StackName}-example
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            - ec2:CreateNetworkInterface
            - ec2:DescribeNetworkInterfaces
            - ec2:DeleteNetworkInterface
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: |
          def handler(event, context):
              print("example")
```

```
      Runtime: python3.9
      VpcConfig:
        SubnetIds:
        - Fn::GetAtt:
          - SubnetOne
          - SubnetId
        - Fn::GetAtt:
          - SubnetTwo
          - SubnetId
        SecurityGroupIds:
        - Fn::GetAtt:
          - SecurityGroup1
          - GroupId
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
```

```
            - logs:PutLogEvents
            - ec2:CreateNetworkInterface
            - ec2:DescribeNetworkInterfaces
            - ec2:DeleteNetworkInterface
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: |
          def handler(event, context):
              print("example")
      Runtime: python3.9
```

[CT.LAMBDA.PR.4] Require an AWS Lambda layer permission to grant access to an AWS organization or specific AWS account

This control checks whether an AWS Lambda layer permission has been configured to grant access to an AWS organization or to a specific AWS account only, by ensuring that public access from all AWS accounts has not been granted to a layer.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Lambda::LayerVersionPermission`
- **CloudFormation guard rule:** CT.LAMBDA.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.LAMBDA.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.LAMBDA.PR.4 example templates

Explanation

By default, a layer that you create is **private** to your AWS account. However, you can share the layer with other accounts or make it public, optionally.

A **public** layer may allow unintended access to your source code and applications. A public Lambda layer can expose valuable information about your account, resources, and internal processes.

Remediation for rule failure

Set the `OrganizationId` parameter to the ID of an AWS organization, or set the `Principal` parameter to an AWS account ID.

The examples that follow show how to implement this remediation.

AWS Lambda layer permission - Example one

An AWS Lambda version permission URL configured to grant layer usage permission to all accounts in an organization. The example is shown in JSON and in YAML.

JSON example

```
{
    "LayerVersionPermission": {
        "Type": "AWS::Lambda::LayerVersionPermission",
        "Properties": {
            "Action": "lambda:GetLayerVersion",
            "LayerVersionArn": {
                "Ref": "LayerVersion"
            },
            "OrganizationId": "o-abc123defg"
        }
    }
}
```

YAML example

```
LayerVersionPermission:
  Type: AWS::Lambda::LayerVersionPermission
```

```
  Properties:
    Action: lambda:GetLayerVersion
    LayerVersionArn: !Ref 'LayerVersion'
    OrganizationId: o-abc123defg
```

The examples that follow show how to implement this remediation.

AWS Lambda layer permission - Example two

An AWS Lambda version permission URL configured to grant layer usage permission for an AWS account. The example is shown in JSON and in YAML.

JSON example

```
{
    "LayerVersionPermission": {
        "Type": "AWS::Lambda::LayerVersionPermission",
        "Properties": {
            "Action": "lambda:GetLayerVersion",
            "LayerVersionArn": {
                "Ref": "LayerVersion"
            },
            "Principal": "123456789012"
        }
    }
}
```

YAML example

```
LayerVersionPermission:
  Type: AWS::Lambda::LayerVersionPermission
  Properties:
    Action: lambda:GetLayerVersion
    LayerVersionArn: !Ref 'LayerVersion'
    Principal: '123456789012'
```

CT.LAMBDA.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   lambda_layer_public_access_prohibited_check
#
# Description:
#   This control checks whether an AWS Lambda layer permission has been configured to
 grant access to an AWS organization or to a specific AWS account only, by ensuring
 that public access from all AWS accounts has not been granted to a layer.
#
# Reports on:
#   AWS::Lambda::LayerVersionPermission
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Lambda layer version permission
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda layer version permission resource
#       And: 'OrganizationId' has not been provided
#       And: 'Principal' has been provided and set to '*'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda layer version permission resource
#       And: 'OrganizationId' has not been provided
#       And: 'Principal' has been provided and set to a non-empty string value other
 than '*'
#      Then: PASS
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda layer version permission resource
#       And: 'OrganizationId' has been provided as a non-empty string
#      Then: PASS

#
# Constants
#
let LAMBDA_LAYER_PERMISSION_TYPE = "AWS::Lambda::LayerVersionPermission"
let INPUT_DOCUMENT = this

#
# Assignments
#
let lambda_layer_permissions = Resources.*[ Type == %LAMBDA_LAYER_PERMISSION_TYPE ]

#
# Primary Rules
#
rule lambda_layer_public_access_prohibited_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %lambda_layer_permissions not
 empty {
    check(%lambda_layer_permissions.Properties)
        <<
        [CT.LAMBDA.PR.4]: Require an AWS Lambda layer permission to grant access to an
 AWS organization or specific AWS account
        [FIX]: Set the 'OrganizationId' parameter to the ID of an AWS organization, or
 set the 'Principal' parameter to an AWS account ID.
        >>
}

rule lambda_layer_public_access_prohibited_check when is_cfn_hook(%INPUT_DOCUMENT,
 %LAMBDA_LAYER_PERMISSION_TYPE) {
    check(%INPUT_DOCUMENT.%LAMBDA_LAYER_PERMISSION_TYPE.resourceProperties)
        <<
        [CT.LAMBDA.PR.4]: Require an AWS Lambda layer permission to grant access to an
 AWS organization or specific AWS account
        [FIX]: Set the 'OrganizationId' parameter to the ID of an AWS organization, or
 set the 'Principal' parameter to an AWS account ID.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(lambda_layer_permission) {
    %lambda_layer_permission [
        OrganizationId not exists
    ] {
        # Scenarios 2 and 3
        Principal exists
        check_is_string_and_not_empty(Principal)
        Principal != "*"
    }
    %lambda_layer_permission [
        OrganizationId exists
    ] {
        # Scenario 4
        check_is_string_and_not_empty(OrganizationId)
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.LAMBDA.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LayerVersion:
    Type: AWS::Lambda::LayerVersion
    Properties:
      CompatibleRuntimes:
      - python3.9
      Content:
        S3Bucket: example-layer-bucket
        S3Key: layer.zip
      Description: Example layer
      LayerName: example-layer
      LicenseInfo: MIT
  LayerVersionPermission:
    Type: AWS::Lambda::LayerVersionPermission
    Properties:
      Action: lambda:GetLayerVersion
      LayerVersionArn:
        Ref: LayerVersion
      OrganizationId: o-abc123defg
      Principal: "*"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LayerVersion:
    Type: AWS::Lambda::LayerVersion
    Properties:
      CompatibleRuntimes:
      - python3.9
      Content:
        S3Bucket: example-layer-bucket
```

```
        S3Key: layer.zip
      Description: Example layer
      LayerName: example-layer
      LicenseInfo: MIT
  LayerVersionPermission:
    Type: AWS::Lambda::LayerVersionPermission
    Properties:
      Action: lambda:GetLayerVersion
      LayerVersionArn:
        Ref: LayerVersion
      Principal: "*"
```

[CT.LAMBDA.PR.5] Require an AWS Lambda function URL to use AWS IAM-based authentication

This control checks whether an AWS Lambda function URL is configured to use authentication that's based on IAM.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Lambda::Url`
- **CloudFormation guard rule:** CT.LAMBDA.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.LAMBDA.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.LAMBDA.PR.5 example templates

Explanation

You can control access to a Lambda function URL using the `AuthType` parameter, combined with resource-based policies that are attached to your specific function. The configuration of these two

components determines who can invoke or perform other administrative actions on your function URL.

The `AuthType` parameter determines how Lambda authenticates or authorizes requests to your Lambda function URL (endpoint). Setting `AuthType` to `NONE` means that Lambda does not perform any authentication before it invokes your function. However, your function's resource-based policy is always in effect, and the policy must grant public access before your Lambda function URL (endpoint) can receive requests.

Remediation for rule failure

Set the `AuthType` parameter to `AWS_IAM`

The examples that follow show how to implement this remediation.

AWS Lambda function URL - Example

An AWS Lambda function URL (endpoint) configured with AWS IAM-based authentication. The example is shown in JSON and in YAML.

JSON example

```
{
    "FunctionUrl": {
        "Type": "AWS::Lambda::Url",
        "Properties": {
            "TargetFunctionArn": {
                "Fn::GetAtt": [
                    "LambdaFunction",
                    "Arn"
                ]
            },
            "AuthType": "AWS_IAM"
        }
    }
}
```

YAML example

```
FunctionUrl:
  Type: AWS::Lambda::Url
  Properties:
    TargetFunctionArn: !GetAtt 'LambdaFunction.Arn'
    AuthType: AWS_IAM
```

CT.LAMBDA.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   lambda_function_url_auth_check
#
# Description:
#   This control checks whether an AWS Lambda function URL is configured to use
 authentication that's
#   based on AWS IAM.
#
# Reports on:
#   AWS::Lambda::Url
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Lambda function URL resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
#       And: 'AuthType' has not been provided
#      Then: FAIL
#   Scenario: 3
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
#       And: 'AuthType' been provided and set to a value other than 'AWS_IAM'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
#       And: 'AuthType' been provided and set to 'AWS_IAM'
#      Then: PASS

#
# Constants
#
let LAMBDA_FUNCTION_URL_TYPE = "AWS::Lambda::Url"
let AUTHORIZED_AUTHENTICATION_TYPES = ["AWS_IAM"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let lambda_function_urls = Resources.*[ Type == %LAMBDA_FUNCTION_URL_TYPE ]

#
# Primary Rules
#
rule lambda_function_url_auth_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %lambda_function_urls not empty {
    check(%lambda_function_urls.Properties)
        <<
        [CT.LAMBDA.PR.5]: Require an AWS Lambda function URL to use AWS IAM-based
 authentication
        [FIX]: Set the 'AuthType' parameter to 'AWS_IAM'
        >>
}

rule lambda_function_url_auth_check when is_cfn_hook(%INPUT_DOCUMENT,
 %LAMBDA_FUNCTION_URL_TYPE) {
    check(%INPUT_DOCUMENT.%LAMBDA_FUNCTION_URL_TYPE.resourceProperties)
        <<
        [CT.LAMBDA.PR.5]: Require an AWS Lambda function URL to use AWS IAM-based
 authentication
        [FIX]: Set the 'AuthType' parameter to 'AWS_IAM'
        >>
}
```

```
#
# Parameterized Rules
#
rule check(lambda_function_url) {
    %lambda_function_url {
        # Scenario 2
        AuthType exists
        # Scenarios 3 and 4
        AuthType in %AUTHORIZED_AUTHENTICATION_TYPES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.LAMBDA.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
```

```
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: "def handler(event, context):\n    print(\"example\")\n"
      Runtime: python3.9
  FunctionUrl:
    Type: AWS::Lambda::Url
    Properties:
      TargetFunctionArn:
        Fn::GetAtt:
        - LambdaFunction
        - Arn
      AuthType: AWS_IAM
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: "def handler(event, context):\n    print(\"example\")\n"
      Runtime: python3.9
  FunctionUrl:
    Type: AWS::Lambda::Url
    Properties:
      TargetFunctionArn:
        Fn::GetAtt:
        - LambdaFunction
        - Arn
      AuthType: NONE
```

[CT.LAMBDA.PR.6] Require an AWS Lambda function URL CORS policy to restrict access to specific origins

This control checks whether an AWS Lambda function URL is configured with a cross-origin resource sharing (CORS) policy that does not grant access to all origins.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Lambda::Url`
- **CloudFormation guard rule:** CT.LAMBDA.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.LAMBDA.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.LAMBDA.PR.6 example templates

Explanation

Cross-Origin Resource Sharing (CORS) is a mechanism based on an HTTP-header, which allows a server to indicate any origins (domain, scheme, or port) other than its own, from which a browser should permit loading resources.

If you set a wildcard origin (*) in a CORS policy, you allow code running in browsers from any origin to gain access to your function URL.

Remediation for rule failure

In the `Cors` parameter, ensure that the value of `AllowOrigins` does not contain wildcard origins (*, `http://*` and `https://*`)

The examples that follow show how to implement this remediation.

AWS Lambda Function URL - Example

AWS Lambda function URL configured with a cross-origin resource sharing (CORS) policy that restricts access to a specific origin. The example is shown in JSON and in YAML.

JSON example

```
{
    "FunctionUrl": {
        "Type": "AWS::Lambda::Url",
        "Properties": {
            "TargetFunctionArn": {
                "Fn::GetAtt": [
                    "LambdaFunction",
                    "Arn"
                ]
            },
            "AuthType": "AWS_IAM",
            "Cors": {
                "AllowOrigins": [
                    "https://example.com"
                ]
            }
        }
    }
}
```

YAML example

```
FunctionUrl:
  Type: AWS::Lambda::Url
  Properties:
    TargetFunctionArn: !GetAtt 'LambdaFunction.Arn'
    AuthType: AWS_IAM
    Cors:
      AllowOrigins:
        - https://example.com
```

CT.LAMBDA.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   lambda_function_url_cors_check
#
# Description:
#   This control checks whether an AWS Lambda function URL is configured with a cross-
origin resource sharing (CORS) policy that does not grant access to all origins.
#
# Reports on:
#   AWS::Lambda::Url
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Lambda function URL resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
#       And: 'AllowOrigins' in 'Cors' has not been provided or has been provided as an
 empty list
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
#       And: 'Cors' has been provided
#       And: 'AllowOrigins' in 'Cors' has been provided as a non-empty list
#       And: 'AllowOrigins' has an entry that contains a wildcard value '*'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Lambda function URL resource
```

```
#       And: 'Cors' has been provided
#       And: 'AllowOrigins' in 'Cors' has been provided as a non-empty list
#       And: No entries in 'AllowOrigins' contain a wildcard value '*'
#      Then: PASS

#
# Constants
#
let LAMBDA_FUNCTION_URL_TYPE = "AWS::Lambda::Url"
let INPUT_DOCUMENT = this

#
# Assignments
#
let lambda_function_urls = Resources.*[ Type == %LAMBDA_FUNCTION_URL_TYPE ]

#
# Primary Rules
#
rule lambda_function_url_cors_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %lambda_function_urls not empty {
    check(%lambda_function_urls.Properties)
        <<
        [CT.LAMBDA.PR.6]: Require an AWS Lambda function URL CORS policy to restrict
 access to specific origins
        [FIX]: In the 'Cors' parameter, ensure that the value of 'AllowOrigins' does
 not contain wildcard origins ('*', 'http://*' and 'https://*')
        >>
}

rule lambda_function_url_cors_check when is_cfn_hook(%INPUT_DOCUMENT,
 %LAMBDA_FUNCTION_URL_TYPE) {
    check(%INPUT_DOCUMENT.%LAMBDA_FUNCTION_URL_TYPE.resourceProperties)
        <<
        [CT.LAMBDA.PR.6]: Require an AWS Lambda function URL CORS policy to restrict
 access to specific origins
        [FIX]: In the 'Cors' parameter, ensure that the value of 'AllowOrigins' does
 not contain wildcard origins ('*', 'http://*' and 'https://*')
        >>
}

#
# Parameterized Rules
#
```

```
rule check(lambda_function_url) {
    %lambda_function_url[
        # Scenario 2
        filter_cors_origins(this)
    ] {
        Cors {
            # Scenarios 3 and 4
            AllowOrigins[*] != /\*/
        }
    }
}

rule filter_cors_origins(lambda_function_url) {
    %lambda_function_url {
        Cors exists
        Cors is_struct
        Cors {
            AllowOrigins exists
            AllowOrigins is_list
            AllowOrigins not empty
        }
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.LAMBDA.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: "def handler(event, context):\n    print(\"example\")\n"
      Runtime: python3.9
  FunctionUrl:
    Type: AWS::Lambda::Url
```

```
    Properties:
      TargetFunctionArn:
        Fn::GetAtt:
        - LambdaFunction
        - Arn
      AuthType: AWS_IAM
      Cors:
        AllowOrigins:
        - https://example.com
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  LambdaFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: LambdaFunctionPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
  LambdaFunction:
    Type: AWS::Lambda::Function
```

```
    Properties:
      Role:
        Fn::GetAtt: LambdaFunctionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: "def handler(event, context):\n    print(\"example\")\n"
      Runtime: python3.9
  FunctionUrl:
    Type: AWS::Lambda::Url
    Properties:
      TargetFunctionArn:
        Fn::GetAtt:
        - LambdaFunction
        - Arn
      AuthType: AWS_IAM
      Cors:
        AllowOrigins:
        - '*'
```

Amazon MQ controls

Topics

- [CT.MQ.PR.1] Require an Amazon MQ ActiveMQ broker to use use active/standby deployment mode for high availability
- [CT.MQ.PR.2] Require an Amazon MQ Rabbit MQ broker to use Multi-AZ cluster mode for high availability

[CT.MQ.PR.1] Require an Amazon MQ ActiveMQ broker to use use active/standby deployment mode for high availability

This control checks whether an Amazon MQ ActiveMQ broker is configured in an active/standby deployment mode.

- **Control objective:** Improve resiliency, Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AmazonMQ::Broker`

- **CloudFormation guard rule:** CT.MQ.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.MQ.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.MQ.PR.1 example templates

Explanation

Amazon MQ ActiveMQ active/standby deployment mode helps you achieve high availability for your Amazon MQ brokers across a single region. The Amazon MQ active/standby deployment mode includes two broker instances, which are configured in a redundant pair across different availability zones.

Usage considerations

- This control applies only to Amazon MQ brokers with an engine type of ACTIVEMQ.

Remediation for rule failure

For Amazon MQ brokers with an engine type of ACTIVEMQ, set the DeploymentMode property to ACTIVE_STANDBY_MULTI_AZ.

The examples that follow show how to implement this remediation.

Amazon MQ ActiveMQ Broker - Example

An Amazon MQ ActiveMQ broker configured in active/standby deployment mode. The example is shown in JSON and in YAML.

JSON example

```
{
    "MQBroker": {
        "Type": "AWS::AmazonMQ::Broker",
```

```
        "Properties": {
            "AutoMinorVersionUpgrade": true,
            "BrokerName": "sample-broker",
            "EngineVersion": "5.17.2",
            "HostInstanceType": "mq.m5.large",
            "PubliclyAccessible": false,
            "Users": [
                {
                    "ConsoleAccess": true,
                    "Username": {
                        "Fn::Sub": "{{resolve:secretsmanager:
${MQBrokerSecret}::username}}"
                    },
                    "Password": {
                        "Fn::Sub": "{{resolve:secretsmanager:
${MQBrokerSecret}::password}}"
                    }
                }
            ],
            "EngineType": "ACTIVEMQ",
            "DeploymentMode": "ACTIVE_STANDBY_MULTI_AZ"
        }
    }
}
```

YAML example

```
MQBroker:
  Type: AWS::AmazonMQ::Broker
  Properties:
    AutoMinorVersionUpgrade: true
    BrokerName: sample-broker
    EngineVersion: 5.17.2
    HostInstanceType: mq.m5.large
    PubliclyAccessible: false
    Users:
      - ConsoleAccess: true
        Username: !Sub '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password: !Sub '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
    EngineType: ACTIVEMQ
    DeploymentMode: ACTIVE_STANDBY_MULTI_AZ
```

CT.MQ.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   mq_active_deployment_mode_check
#
# Description:
#   This control checks whether an Amazon MQ ActiveMQ broker is configured in an
 active/standby deployment mode.
#
# Reports on:
#   AWS::AmazonMQ::Broker
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon MQ broker resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
#       And: 'EngineType' has been provided and is equal to a value other than
 'ACTIVEMQ'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
#       And: 'EngineType' has been provided and set to 'ACTIVEMQ'
#       And: 'DeploymentMode' has not been provided or has been provided and set to a
 value other
```

```
#            than 'ACTIVE_STANDBY_MULTI_AZ'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
#       And: 'EngineType' has been provided and set to 'ACTIVEMQ'
#       And: 'DeploymentMode' has been provided and set to 'ACTIVE_STANDBY_MULTI_AZ'
#      Then: PASS

#
# Constants
#
let MQ_BROKER_TYPE = "AWS::AmazonMQ::Broker"
let ENGINES_WITH_CLUSTER_DEPLOYMENT_SUPPORT = ["ACTIVEMQ"]
let ALLOWED_DEPLOYMENT_MODES = ["ACTIVE_STANDBY_MULTI_AZ"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let mq_brokers = Resources.*[ Type == %MQ_BROKER_TYPE ]

#
# Primary Rules
#
rule mq_active_deployment_mode_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %mq_brokers not empty {
    check(%mq_brokers.Properties)
        <<
        [CT.MQ.PR.1]: Require an Amazon MQ ActiveMQ broker to use use active/standby
 deployment mode for high availability
        [FIX]: For Amazon MQ brokers with an engine type of ACTIVEMQ, set the
 DeploymentMode property to ACTIVE_STANDBY_MULTI_AZ.
        >>
}

rule mq_active_deployment_mode_check when is_cfn_hook(%INPUT_DOCUMENT, %MQ_BROKER_TYPE)
 {
    check(%INPUT_DOCUMENT.%MQ_BROKER_TYPE.resourceProperties)
        <<
        [CT.MQ.PR.1]: Require an Amazon MQ ActiveMQ broker to use use active/standby
 deployment mode for high availability
        [FIX]: For Amazon MQ brokers with an engine type of ACTIVEMQ, set the
 DeploymentMode property to ACTIVE_STANDBY_MULTI_AZ.
```

```
        >>
}

#
# Parameterized Rules
#
rule check(mq_broker) {
    %mq_broker [
        # Scenario 2
        filter_engine(this)
    ] {
        # Scenarios 3 and 4
        DeploymentMode exists
        DeploymentMode in %ALLOWED_DEPLOYMENT_MODES
    }
}

rule filter_engine(mq_broker) {
    %mq_broker {
        EngineType exists
        EngineType in %ENGINES_WITH_CLUSTER_DEPLOYMENT_SUPPORT
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.MQ.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  MQBrokerSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: MQ broker secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemqusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: ',:='
  MQBroker:
    Type: AWS::AmazonMQ::Broker
    Properties:
      AutoMinorVersionUpgrade: true
      BrokerName:
        Ref: AWS::StackName
      EngineVersion: 5.17.2
      HostInstanceType: mq.m5.large
      PubliclyAccessible: false
      Users:
      - ConsoleAccess: true
        Username:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
      EngineType: ACTIVEMQ
      DeploymentMode: ACTIVE_STANDBY_MULTI_AZ
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
```

```
  MQBrokerSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: MQ broker secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemqusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: ',:='
  MQBroker:
    Type: AWS::AmazonMQ::Broker
    Properties:
      AutoMinorVersionUpgrade: true
      BrokerName:
        Ref: AWS::StackName
      EngineVersion: 5.17.2
      HostInstanceType: mq.m5.large
      PubliclyAccessible: false
      Users:
      - ConsoleAccess: true
        Username:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
      EngineType: ACTIVEMQ
      DeploymentMode: SINGLE_INSTANCE
```

[CT.MQ.PR.2] Require an Amazon MQ Rabbit MQ broker to use Multi-AZ cluster mode for high availability

This control checks whether an Amazon MQ RabbitMQ broker is configured in a cluster deployment mode, to allow for high availability.

- **Control objective:** Improve resiliency, Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::AmazonMQ::Broker`
- **CloudFormation guard rule:** CT.MQ.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.MQ.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.MQ.PR.2 example templates

Explanation

Amazon MQ cluster deployments for RabbitMQ help you achieve high availability for your Amazon MQ brokers across a single region. RabbitMQ clusters include three broker instances, which are configured in a cluster across different availability zones.

Usage considerations

- This control applies only to Amazon MQ brokers with an engine type of RABBITMQ.

Remediation for rule failure

For Amazon MQ brokers with an engine type of RABBITMQ, set the DeploymentMode property to CLUSTER_MULTI_AZ.

The examples that follow show how to implement this remediation.

Amazon MQ RabbitMQ Broker - Example

An Amazon MQ RabbitMQ broker configured in a cluster deployment mode. The example is shown in JSON and in YAML.

JSON example

```
{
    "MQBroker": {
        "Type": "AWS::AmazonMQ::Broker",
        "Properties": {
            "AutoMinorVersionUpgrade": true,
            "BrokerName": "sample-mq-broker",
```

```
            "EngineVersion": "3.10.10",
            "HostInstanceType": "mq.m5.large",
            "PubliclyAccessible": false,
            "Users": [
                {
                    "ConsoleAccess": true,
                    "Username": {
                        "Fn::Sub": "{{resolve:secretsmanager:
${MQBrokerSecret}::username}}"
                    },
                    "Password": {
                        "Fn::Sub": "{{resolve:secretsmanager:
${MQBrokerSecret}::password}}"
                    }
                }
            ],
            "EngineType": "RABBITMQ",
            "DeploymentMode": "CLUSTER_MULTI_AZ"
        }
    }
}
```

YAML example

```
MQBroker:
  Type: AWS::AmazonMQ::Broker
  Properties:
    AutoMinorVersionUpgrade: true
    BrokerName: sample-mq-broker
    EngineVersion: 3.10.10
    HostInstanceType: mq.m5.large
    PubliclyAccessible: false
    Users:
      - ConsoleAccess: true
        Username: !Sub '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password: !Sub '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
    EngineType: RABBITMQ
    DeploymentMode: CLUSTER_MULTI_AZ
```

CT.MQ.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   mq_rabbit_deployment_mode_check
#
# Description:
#   This control checks whether an Amazon MQ RabbitMQ broker is configured in a cluster
 deployment mode, to allow for high availability.
#
# Reports on:
#   AWS::AmazonMQ::Broker
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon MQ broker resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
#       And: 'EngineType' been provided and is equal to a value other than 'RABBITMQ'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
#       And: 'EngineType' been provided and is equal to 'RABBITMQ'
#       And: 'DeploymentMode' has not been provided or has been provided and set to a
 value other
#            than 'CLUSTER_MULTI_AZ'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MQ broker resource
```

```
#       And: 'EngineType' been provided and is equal to 'RABBITMQ'
#       And: 'DeploymentMode' has been provided and set to 'CLUSTER_MULTI_AZ'
#      Then: PASS

#
# Constants
#
let MQ_BROKER_TYPE = "AWS::AmazonMQ::Broker"
let ENGINES_WITH_CLUSTER_DEPLOYMENT_SUPPORT = ["RABBITMQ"]
let ALLOWED_DEPLOYMENT_MODES = ["CLUSTER_MULTI_AZ"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let mq_brokers = Resources.*[ Type == %MQ_BROKER_TYPE ]

#
# Primary Rules
#
rule mq_rabbit_deployment_mode_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %mq_brokers not empty {
    check(%mq_brokers.Properties)
        <<
        [CT.MQ.PR.2]: Require an Amazon MQ Rabbit MQ broker to use Multi-AZ cluster
 mode for high availability
        [FIX]: For Amazon MQ brokers with an engine type of RABBITMQ, set the
 DeploymentMode property to CLUSTER_MULTI_AZ.
        >>
}

rule mq_rabbit_deployment_mode_check when is_cfn_hook(%INPUT_DOCUMENT, %MQ_BROKER_TYPE)
 {
    check(%INPUT_DOCUMENT.%MQ_BROKER_TYPE.resourceProperties)
        <<
        [CT.MQ.PR.2]: Require an Amazon MQ Rabbit MQ broker to use Multi-AZ cluster
 mode for high availability
        [FIX]: For Amazon MQ brokers with an engine type of RABBITMQ, set the
 DeploymentMode property to CLUSTER_MULTI_AZ.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(mq_broker) {
    %mq_broker [
        # Scenario 2
        filter_engine(this)
    ] {
        # Scenarios 3 and 4
        DeploymentMode exists
        DeploymentMode in %ALLOWED_DEPLOYMENT_MODES
    }
}

rule filter_engine(mq_broker) {
    %mq_broker {
        EngineType exists
        EngineType in %ENGINES_WITH_CLUSTER_DEPLOYMENT_SUPPORT
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.MQ.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  MQBrokerSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: MQ broker secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemqusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: ',:='
  MQBroker:
    Type: AWS::AmazonMQ::Broker
    Properties:
      AutoMinorVersionUpgrade: true
      BrokerName:
        Ref: AWS::StackName
      EngineVersion: 3.10.10
      HostInstanceType: mq.m5.large
      PubliclyAccessible: false
      Users:
      - ConsoleAccess: true
        Username:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
      EngineType: RABBITMQ
      DeploymentMode: CLUSTER_MULTI_AZ
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  MQBrokerSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: MQ broker secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemqusername"}'
        GenerateStringKey: password
        PasswordLength: 16
```

```
      ExcludeCharacters: ',:='
  MQBroker:
    Type: AWS::AmazonMQ::Broker
    Properties:
      AutoMinorVersionUpgrade: true
      BrokerName:
        Ref: AWS::StackName
      EngineVersion: 3.10.10
      HostInstanceType: mq.m5.large
      PubliclyAccessible: false
      Users:
      - ConsoleAccess: true
        Username:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::username}}'
        Password:
          Fn::Sub: '{{resolve:secretsmanager:${MQBrokerSecret}::password}}'
      EngineType: RABBITMQ
      DeploymentMode: SINGLE_INSTANCE
```

Amazon Managed Streaming for Apache Kafka (Amazon MSK) controls

Topics

- [CT.MSK.PR.1] Require an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to enforce encryption in transit between cluster broker nodes
- [CT.MSK.PR.2] Require an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to be configured with PublicAccess disabled

[CT.MSK.PR.1] Require an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to enforce encryption in transit between cluster broker nodes

This control checks whether an Amazon MSK cluster is configured to encrypt data in transit between broker nodes of the cluster.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::MSK::Cluster`
- **CloudFormation guard rule:** CT.MSK.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.MSK.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.MSK.PR.1 example templates

Explanation

Amazon MSK uses TLSv1.2. By default, it encrypts data in transit between the brokers of your Amazon MSK cluster. However, you can override this default at the time you create the cluster.

Usage considerations

- Although we highly recommend enabling in-transit encryption, it can add additional CPU overhead and a few milliseconds of latency. Most use cases aren't sensitive to these differences, and the magnitude of impact depends on the configuration of your cluster, clients, and usage profile.

Remediation for rule failure

In the EncryptionInfo property, provide an `EncryptionInTransit` configuration and set the value of `InCluster` to true. Otherwise, omit the `InCluster` property to adopt the default value of true.

The examples that follow show how to implement this remediation.

Amazon MSK Cluster - Example

An Amazon MSK cluster configured to encrypt data in transit between the broker nodes of the cluster. The example is shown in JSON and in YAML.

JSON example

```
{
    "MSKCluster": {
        "Type": "AWS::MSK::Cluster",
        "Properties": {
            "BrokerNodeGroupInfo": {
                "ClientSubnets": [
                    {
                        "Ref": "SubnetOne"
                    },
                    {
                        "Ref": "SubnetTwo"
                    }
                ],
                "InstanceType": "kafka.t3.small",
                "SecurityGroups": [
                    {
                        "Fn::GetAtt": [
                            "SecurityGroup",
                            "GroupId"
                        ]
                    }
                ],
                "StorageInfo": {
                    "EBSStorageInfo": {
                        "VolumeSize": 1000
                    }
                }
            },
            "ClusterName": {
                "Fn::Sub": "MSKCluster-${AWS::StackName}"
            },
            "KafkaVersion": "3.4.0",
            "NumberOfBrokerNodes": 2,
            "EnhancedMonitoring": "DEFAULT",
            "EncryptionInfo": {
                "EncryptionInTransit": {
                    "InCluster": true
                }
            }
        }
    }
}
```

YAML example

```
MSKCluster:
  Type: AWS::MSK::Cluster
  Properties:
    BrokerNodeGroupInfo:
      ClientSubnets:
        - !Ref 'SubnetOne'
        - !Ref 'SubnetTwo'
      InstanceType: kafka.t3.small
      SecurityGroups:
        - !GetAtt 'SecurityGroup.GroupId'
      StorageInfo:
        EBSStorageInfo:
          VolumeSize: 1000
    ClusterName: !Sub 'MSKCluster-${AWS::StackName}'
    KafkaVersion: 3.4.0
    NumberOfBrokerNodes: 2
    EnhancedMonitoring: DEFAULT
    EncryptionInfo:
      EncryptionInTransit:
        InCluster: true
```

CT.MSK.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   msk_broker_node_tls_check
#
# Description:
#   This control checks whether an Amazon MSK cluster is configured to encrypt data in
 transit between broker nodes of the cluster.
#
# Reports on:
#   AWS::MSK::Cluster
#
```

```
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon MSK cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'InCluster' in 'EncryptionInfo.EncryptionInTransit' has been provided and
#            set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'InCluster' in 'EncryptionInfo.EncryptionInTransit' has not been provided
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'InCluster' in 'EncryptionInfo.EncryptionInTransit' has been provided and
#            set to bool(true)
#      Then: PASS

#
# Constants
#
let MSK_CLUSTER_TYPE = "AWS::MSK::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let msk_clusters = Resources.*[ Type == %MSK_CLUSTER_TYPE ]

#
# Primary Rules
#
rule msk_broker_node_tls_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                    %msk_clusters not empty {
    check(%msk_clusters.Properties)
        <<
        [CT.MSK.PR.1]: Require an Amazon Managed Streaming for Apache Kafka (Amazon
 MSK) cluster to enforce encryption in transit between cluster broker nodes
        [FIX]: In the EncryptionInfo property, provide an 'EncryptionInTransit'
 configuration and set the value of 'InCluster' to true. Otherwise, omit the
 'InCluster' property to adopt the default value of true.
        >>
}

rule msk_broker_node_tls_check when is_cfn_hook(%INPUT_DOCUMENT, %MSK_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%MSK_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.MSK.PR.1]: Require an Amazon Managed Streaming for Apache Kafka (Amazon
 MSK) cluster to enforce encryption in transit between cluster broker nodes
        [FIX]: In the EncryptionInfo property, provide an 'EncryptionInTransit'
 configuration and set the value of 'InCluster' to true. Otherwise, omit the
 'InCluster' property to adopt the default value of true.
        >>
}

#
# Parameterized Rules
#
rule check(msk_cluster) {
    %msk_cluster {
        # Scenario 3
        EncryptionInfo not exists or
        # Scenarios 2 and 4
        check_encryption_info_config(this)
    }
}

 rule check_encryption_info_config(msk_cluster) {
     %msk_cluster {
        EncryptionInfo exists
        EncryptionInfo is_struct

        EncryptionInfo {
            EncryptionInTransit not exists or
            check_encryption_in_transit_config(this)
        }
     }
```

```
 }

 rule check_encryption_in_transit_config(encryption_info_config) {
     %encryption_info_config {
        EncryptionInTransit exists
        EncryptionInTransit is_struct

        EncryptionInTransit {
            InCluster not exists or
            InCluster == true
        }
     }
 }


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.MSK.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
```

```
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: MSK Security Group
      SecurityGroupIngress:
      - Description: ZooKeeper plaintext
        FromPort: 2181
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 2181
      - Description: Bootstrap servers plaintext
        FromPort: 9092
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9092
      - Description: Bootstrap servers TLS
```

```
        FromPort: 9094
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9094
      VpcId:
        Ref: VPC
  MSKCluster:
    Type: AWS::MSK::Cluster
    Properties:
      BrokerNodeGroupInfo:
        ClientSubnets:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        InstanceType: kafka.t3.small
        SecurityGroups:
        - Fn::GetAtt:
          - SecurityGroup
          - GroupId
        StorageInfo:
          EBSStorageInfo:
            VolumeSize: 1000
      ClusterName:
        Fn::Sub: MSKCluster-${AWS::StackName}
      KafkaVersion: 3.4.0
      NumberOfBrokerNodes: 2
      EnhancedMonitoring: DEFAULT
      EncryptionInfo:
        EncryptionInTransit:
          InCluster: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
```

```
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: MSK Security Group
      SecurityGroupIngress:
      - Description: ZooKeeper plaintext
        FromPort: 2181
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 2181
      - Description: Bootstrap servers plaintext
        FromPort: 9092
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9092
```

```
      - Description: Bootstrap servers TLS
        FromPort: 9094
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9094
      VpcId:
        Ref: VPC
  MSKCluster:
    Type: AWS::MSK::Cluster
    Properties:
      BrokerNodeGroupInfo:
        ClientSubnets:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        InstanceType: kafka.t3.small
        SecurityGroups:
        - Fn::GetAtt:
          - SecurityGroup
          - GroupId
        StorageInfo:
          EBSStorageInfo:
            VolumeSize: 1000
      ClusterName:
        Fn::Sub: MSKCluster-${AWS::StackName}
      KafkaVersion: 3.4.0
      NumberOfBrokerNodes: 2
      EnhancedMonitoring: DEFAULT
      EncryptionInfo:
        EncryptionInTransit:
          InCluster: false
```

[CT.MSK.PR.2] Require an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to be configured with PublicAccess disabled

This control checks whether an Amazon MSK cluster is configured to disallow public access to cluster brokers by means of the PublicAccess property.

- **Control objective:** Limit network access

- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::MSK::Cluster`
- **CloudFormation guard rule:** CT.MSK.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.MSK.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.MSK.PR.2 example templates

Explanation

Amazon MSK gives you the option to turn on public access to the brokers of Amazon MSK clusters that run Apache Kafka version 2.6.0 or later. For security reasons, you can't turn on public access while creating an Amazon MSK cluster. However, you can update an existing cluster to make it publicly accessible.

Usage considerations

- In addition to configuring the PublicAccess property, other prerequisite conditions are required when you enable public access to Amazon MSK clusters. For more information on configuring Amazon MSK clusters for public access, see Public Access in the *Amazon MSK Developer Guide*.

Remediation for rule failure

In the parameter BrokerNodeGroupInfo.ConnectivityInfo.PublicAccess, set the value of Type to DISABLED, or to adopt the default value of DISABLED, do not provide a PublicAccess configuration.

The examples that follow show how to implement this remediation.

Amazon MSK Cluster - Example

An Amazon MSK cluster configured to disallow public access to cluster brokers through the PublicAccess property. The example is shown in JSON and in YAML.

JSON example

```
{
    "MSKCluster": {
        "Type": "AWS::MSK::Cluster",
        "Properties": {
            "ClusterName": {
                "Fn::Sub": "MSKCluster-${AWS::StackName}"
            },
            "KafkaVersion": "3.4.0",
            "NumberOfBrokerNodes": 2,
            "EnhancedMonitoring": "DEFAULT",
            "EncryptionInfo": {
                "EncryptionInTransit": {
                    "InCluster": true
                }
            },
            "ClientAuthentication": {
                "Sasl": {
                    "Iam": {
                        "Enabled": true
                    }
                }
            },
            "BrokerNodeGroupInfo": {
                "ClientSubnets": [
                    {
                        "Ref": "SubnetOne"
                    },
                    {
                        "Ref": "SubnetTwo"
                    }
                ],
                "InstanceType": "kafka.t3.small",
                "SecurityGroups": [
                    {
                        "Fn::GetAtt": [
                            "SecurityGroup",
                            "GroupId"
                        ]
                    }
                ],
                "StorageInfo": {
```

```
                    "EBSStorageInfo": {
                        "VolumeSize": 1000
                    }
                },
                "ConnectivityInfo": {
                    "PublicAccess": {
                        "Type": "DISABLED"
                    }
                }
            }
        }
    }
}
```

YAML example

```
MSKCluster:
  Type: AWS::MSK::Cluster
  Properties:
    ClusterName: !Sub 'MSKCluster-${AWS::StackName}'
    KafkaVersion: 3.4.0
    NumberOfBrokerNodes: 2
    EnhancedMonitoring: DEFAULT
    EncryptionInfo:
      EncryptionInTransit:
        InCluster: true
    ClientAuthentication:
      Sasl:
        Iam:
          Enabled: true
    BrokerNodeGroupInfo:
      ClientSubnets:
        - !Ref 'SubnetOne'
        - !Ref 'SubnetTwo'
      InstanceType: kafka.t3.small
      SecurityGroups:
        - !GetAtt 'SecurityGroup.GroupId'
      StorageInfo:
        EBSStorageInfo:
          VolumeSize: 1000
      ConnectivityInfo:
```

```
        PublicAccess:
          Type: DISABLED
```

CT.MSK.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   msk_public_access_check
#
# Description:
#   This control checks whether an Amazon MSK cluster is configured to disallow public
 access to cluster brokers by means of the PublicAccess property.
#
# Reports on:
#   AWS::MSK::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon MSK cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'BrokerNodeGroupInfo' has not been provided
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an MSK cluster resource
#       And: 'BrokerNodeGroupInfo' has been provided
#       And: 'Type' in 'ConnectivityInfo.PublicAccess' has been provided and
```

```
#            set to a value other than 'DISABLED'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'BrokerNodeGroupInfo' has been provided
#       And: 'Type' in 'ConnectivityInfo.PublicAccess' has not been provided
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon MSK cluster resource
#       And: 'BrokerNodeGroupInfo' has been provided
#       And: 'Type' in 'ConnectivityInfo.PublicAccess' has been provided and
#            set to 'DISABLED'
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let MSK_CLUSTER_TYPE = "AWS::MSK::Cluster"
let DISABLED_PUBLIC_ACCESS_TYPE = "DISABLED"

#
# Assignments
#
let msk_clusters = Resources.*[ Type == %MSK_CLUSTER_TYPE ]

#
# Primary Rules
#
rule msk_public_access_check when is_cfn_template(%INPUT_DOCUMENT)
                                  %msk_clusters not empty {
    check(%msk_clusters.Properties)
        <<
        [CT.MSK.PR.2]: Require an Amazon Managed Streaming for Apache Kafka (Amazon
 MSK) cluster to be configured with PublicAccess disabled
        [FIX]: In the parameter BrokerNodeGroupInfo.ConnectivityInfo.PublicAccess,
 set the value of Type to DISABLED, or to adopt the default value of DISABLED, do not
 provide a PublicAccess configuration.
        >>
}

rule msk_public_access_check when is_cfn_hook(%INPUT_DOCUMENT, %MSK_CLUSTER_TYPE) {
```

```
    check(%INPUT_DOCUMENT.%MSK_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.MSK.PR.2]: Require an Amazon Managed Streaming for Apache Kafka (Amazon
 MSK) cluster to be configured with PublicAccess disabled
        [FIX]: In the parameter BrokerNodeGroupInfo.ConnectivityInfo.PublicAccess,
 set the value of Type to DISABLED, or to adopt the default value of DISABLED, do not
 provide a PublicAccess configuration.
        >>
}

#
# Parameterized Rules
#
rule check(msk_cluster) {
    %msk_cluster [
        # Scenario 2
        BrokerNodeGroupInfo exists
        BrokerNodeGroupInfo is_struct
    ] {
        BrokerNodeGroupInfo {
            # Scenarios 3, 4 and 5
            ConnectivityInfo not exists or
            check_connectivity_info_config(this)
        }
    }
}

 rule check_connectivity_info_config(broker_node_group_info) {
     %broker_node_group_info {
        ConnectivityInfo exists
        ConnectivityInfo is_struct

        ConnectivityInfo {
            PublicAccess not exists or
            check_public_access_config(this)
        }
     }
 }

 rule check_public_access_config(connectivity_info_config) {
     %connectivity_info_config {
        PublicAccess exists
        PublicAccess is_struct
```

```
        PublicAccess {
            Type not exists or
            Type == %DISABLED_PUBLIC_ACCESS_TYPE
        }
     }
 }

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.MSK.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: MSK Security Group
      SecurityGroupIngress:
      - Description: ZooKeeper plaintext
        FromPort: 2181
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 2181
      - Description: Bootstrap servers plaintext
        FromPort: 9092
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9092
      - Description: Bootstrap servers TLS
        FromPort: 9094
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9094
      VpcId:
```

```
        Ref: VPC
  MSKCluster:
    Type: AWS::MSK::Cluster
    Properties:
      ClusterName:
        Fn::Sub: MSKCluster-${AWS::StackName}
      KafkaVersion: 3.4.0
      NumberOfBrokerNodes: 2
      EnhancedMonitoring: DEFAULT
      EncryptionInfo:
        EncryptionInTransit:
          InCluster: true
      ClientAuthentication:
        Sasl:
          Iam:
            Enabled: true
      BrokerNodeGroupInfo:
        ClientSubnets:
        - Ref: SubnetOne
        - Ref: SubnetTwo
        InstanceType: kafka.t3.small
        SecurityGroups:
        - Fn::GetAtt:
          - SecurityGroup
          - GroupId
        StorageInfo:
          EBSStorageInfo:
            VolumeSize: 1000
        ConnectivityInfo:
          PublicAccess:
            Type: DISABLED
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
```

```
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: MSK Security Group
      SecurityGroupIngress:
      - Description: ZooKeeper plaintext
        FromPort: 2181
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 2181
      - Description: Bootstrap servers plaintext
        FromPort: 9092
        IpProtocol: tcp
        CidrIp:
          Fn::GetAtt:
          - VPC
          - CidrBlock
        ToPort: 9092
      - Description: Bootstrap servers TLS
```

```
      FromPort: 9094
      IpProtocol: tcp
      CidrIp:
        Fn::GetAtt:
        - VPC
        - CidrBlock
      ToPort: 9094
    VpcId:
      Ref: VPC
MSKCluster:
  Type: AWS::MSK::Cluster
  Properties:
    ClusterName:
      Fn::Sub: MSKCluster-${AWS::StackName}
    KafkaVersion: 3.4.0
    NumberOfBrokerNodes: 2
    EnhancedMonitoring: DEFAULT
    EncryptionInfo:
      EncryptionInTransit:
        InCluster: true
    ClientAuthentication:
      Sasl:
        Iam:
          Enabled: true
    BrokerNodeGroupInfo:
      ClientSubnets:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      InstanceType: kafka.t3.small
      SecurityGroups:
      - Fn::GetAtt:
        - SecurityGroup
        - GroupId
      StorageInfo:
        EBSStorageInfo:
          VolumeSize: 1000
      ConnectivityInfo:
        PublicAccess:
          Type: SERVICE_PROVIDED_EIPS
```

Amazon Neptune controls

Topics

- [CT.NEPTUNE.PR.1] Require an Amazon Neptune DB cluster to have AWS Identity and Access Management (IAM) database authentication enabled
- [CT.NEPTUNE.PR.2] Require an Amazon Neptune DB cluster to have deletion protection enabled
- [CT.NEPTUNE.PR.3] Require an Amazon Neptune DB cluster to have storage encryption enabled
- [CT.NEPTUNE.PR.4] Require an Amazon Neptune DB cluster to enable Amazon CloudWatch Logs export for audit logs
- [CT.NEPTUNE.PR.5] Require an Amazon Neptune DB cluster to set a backup retention period greater than or equal to seven days

[CT.NEPTUNE.PR.1] Require an Amazon Neptune DB cluster to have AWS Identity and Access Management (IAM) database authentication enabled

This control checks whether an Amazon Neptune cluster has AWS Identity and Access Management (IAM) database authentication enabled.

- **Control objective:** Enforce least privilege, Use strong authentication
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Neptune::DBCluster`
- **CloudFormation guard rule:** CT.NEPTUNE.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NEPTUNE.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.NEPTUNE.PR.1 example templates

Explanation

You can use AWS Identity and Access Management (IAM) to authenticate to your Neptune DB instance or DB cluster. IAM allows you to manage access to your database resources centrally, instead of managing access individually on each DB instance or cluster.

Remediation for rule failure

Set the value of the `IamAuthEnabled` parameter to true.

The examples that follow show how to implement this remediation.

Amazon Neptune Cluster - Example

Neptune Cluster configured with AWS IAM database authentication enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "NeptuneDBCluster": {
        "Type": "AWS::Neptune::DBCluster",
        "Properties": {
            "IamAuthEnabled": true
        }
    }
}
```

YAML example

```
NeptuneDBCluster:
  Type: AWS::Neptune::DBCluster
  Properties:
    IamAuthEnabled: true
```

CT.NEPTUNE.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   neptune_cluster_iam_database_authentication_check
#
# Description:
#   This control checks whether an Amazon Neptune cluster has AWS Identity and Access
 Management (IAM) database authentication enabled.
#
# Reports on:
#   AWS::Neptune::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Neptune DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'IamAuthEnabled' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'IamAuthEnabled' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'IamAuthEnabled' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let NEPTUNE_CLUSTER_TYPE = "AWS::Neptune::DBCluster"
```

```
let INPUT_DOCUMENT = this

#
# Assignments
#
let neptune_db_clusters = Resources.*[ Type == %NEPTUNE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule neptune_cluster_iam_database_authentication_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %neptune_db_clusters not
 empty {
    check(%neptune_db_clusters.Properties)
        <<
        [CT.NEPTUNE.PR.1]: Require an Amazon Neptune DB cluster to have AWS Identity
 and Access Management (IAM) database authentication enabled
        [FIX]: Set the value of the 'IamAuthEnabled' parameter to true.
        >>
}

rule neptune_cluster_iam_database_authentication_check when
 is_cfn_hook(%INPUT_DOCUMENT, %NEPTUNE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%NEPTUNE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.NEPTUNE.PR.1]: Require an Amazon Neptune DB cluster to have AWS Identity
 and Access Management (IAM) database authentication enabled
        [FIX]: Set the value of the 'IamAuthEnabled' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(neptune_cluster) {
    %neptune_cluster {
        # Scenario 2
        IamAuthEnabled exists
        # Scenarios 3 and 4
        IamAuthEnabled == true
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NEPTUNE.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      IamAuthEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      IamAuthEnabled: false
```

[CT.NEPTUNE.PR.2] Require an Amazon Neptune DB cluster to have deletion protection enabled

This control checks whether an Amazon Neptune cluster has deletion protection enabled.

- **Control objective:** Improve availability, Protect configurations
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Neptune::DBCluster`
- **CloudFormation guard rule:** CT.NEPTUNE.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NEPTUNE.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.NEPTUNE.PR.2 example templates

Explanation

Cluster deletion protection adds an additional layer of protection against accidental database deletion, or deletion by an unauthorized entity. A Neptune cluster cannot be deleted while deletion protection is enabled. Deletion protection must be disabled first, before a delete request can succeed.

Remediation for rule failure

Set the value of the `DeletionProtection` parameter to true.

The examples that follow show how to implement this remediation.

Amazon Neptune Cluster - Example

An Amazon Neptune Cluster configured with deletion protection enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "NeptuneDBCluster": {
        "Type": "AWS::Neptune::DBCluster",
        "Properties": {
            "DeletionProtection": true
        }
    }
}
```

YAML example

```
NeptuneDBCluster:
  Type: AWS::Neptune::DBCluster
  Properties:
    DeletionProtection: true
```

CT.NEPTUNE.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   neptune_cluster_deletion_protection_enabled_check
#
# Description:
#   This control checks whether an Amazon Neptune cluster has deletion protection
 enabled.
#
# Reports on:
#   AWS::Neptune::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Neptune DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'DeletionProtection' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'DeletionProtection' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'DeletionProtection' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let NEPTUNE_CLUSTER_TYPE = "AWS::Neptune::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let neptune_db_clusters = Resources.*[ Type == %NEPTUNE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule neptune_cluster_deletion_protection_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %neptune_db_clusters not
 empty {
    check(%neptune_db_clusters.Properties)
        <<
```

```
        [CT.NEPTUNE.PR.2]: Require an Amazon Neptune DB cluster to have deletion
 protection enabled
        [FIX]: Set the value of the 'DeletionProtection' parameter to true.
        >>
}

rule neptune_cluster_deletion_protection_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %NEPTUNE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%NEPTUNE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.NEPTUNE.PR.2]: Require an Amazon Neptune DB cluster to have deletion
 protection enabled
        [FIX]: Set the value of the 'DeletionProtection' parameter to true.
        >>
}

#
# Parameterized Rules
#
rule check(neptune_cluster) {
    %neptune_cluster {
        # Scenario 2
        DeletionProtection exists
        # Scenarios 3 and 4
        DeletionProtection == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NEPTUNE.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      DeletionProtection: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      DeletionProtection: false
```

[CT.NEPTUNE.PR.3] Require an Amazon Neptune DB cluster to have storage encryption enabled

This control checks whether an Amazon Neptune cluster has storage encryption enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Neptune::DBCluster`
- **CloudFormation guard rule:** CT.NEPTUNE.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NEPTUNE.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.NEPTUNE.PR.3 example templates

Explanation

Neptune encrypted instances provide an additional layer of data protection, because they help to secure your data from unauthorized access to the underlying storage. Neptune encryption helps increase data protection of your applications that are deployed in the cloud. You also can use it to fulfill compliance requirements for data-at-rest encryption.

Usage considerations

- This control checks only whether the `StorageEncrypted` property is provided and set to `true`. When you create an encrypted Neptune DB instance, you also can supply the AWS KMS key identifier for your encryption key by means of the `KmsKeyId` property. If you don't specify an AWS KMS key identifier, Neptune uses your default Amazon RDS encryption key (aws/rds) for your new Neptune DB instance.

Remediation for rule failure

Set `StorageEncrypted` to `true`.

The examples that follow show how to implement this remediation.

Amazon Neptune Cluster - Example

An Amazon Neptune Cluster configured with storage encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "NeptuneDBCluster": {
        "Type": "AWS::Neptune::DBCluster",
        "Properties": {
            "StorageEncrypted": true
```

```
        }
    }
}
```

YAML example

```
NeptuneDBCluster:
  Type: AWS::Neptune::DBCluster
  Properties:
    StorageEncrypted: true
```

CT.NEPTUNE.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   neptune_cluster_encrypted_check
#
# Description:
#   This control checks whether an Amazon Neptune cluster has storage encryption
 enabled.
#
# Reports on:
#   AWS::Neptune::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Neptune DB cluster resources
#      Then: SKIP
```

```
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'StorageEncrypted' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'StorageEncrypted' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'StorageEncrypted' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let NEPTUNE_CLUSTER_TYPE = "AWS::Neptune::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let neptune_db_clusters = Resources.*[ Type == %NEPTUNE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule neptune_cluster_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %neptune_db_clusters not empty {
    check(%neptune_db_clusters.Properties)
        <<
        [CT.NEPTUNE.PR.3]: Require an Amazon Neptune DB cluster to have storage
 encryption enabled
        [FIX]: Set 'StorageEncrypted' to 'true'.
        >>
}

rule neptune_cluster_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NEPTUNE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%NEPTUNE_CLUSTER_TYPE.resourceProperties)
```

```
        <<
        [CT.NEPTUNE.PR.3]: Require an Amazon Neptune DB cluster to have storage
 encryption enabled
        [FIX]: Set 'StorageEncrypted' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(neptune_cluster) {
    %neptune_cluster {
        # Scenario 2
        StorageEncrypted exists
        # Scenarios 3 and 4
        StorageEncrypted == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NEPTUNE.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      StorageEncrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      StorageEncrypted: false
```

[CT.NEPTUNE.PR.4] Require an Amazon Neptune DB cluster to enable Amazon CloudWatch Logs export for audit logs

This control checks whether an Amazon Neptune cluster is configured to send audit logs to Amazon CloudWatch Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Neptune::DBCluster`
- **CloudFormation guard rule:** CT.NEPTUNE.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NEPTUNE.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NEPTUNE.PR.4 example templates

Explanation

You can configure an Amazon Neptune DB cluster to publish audit log data to a log group in Amazon CloudWatch Logs. Storing your Neptune DB cluster audit log data in Amazon CloudWatch Logs allows you to perform real-time analysis of the log data, and also to use Amazon CloudWatch to create alarms and view metrics.

Remediation for rule failure

In the `EnableCloudwatchLogsExports` parameter, set an entry to the value audit.

The examples that follow show how to implement this remediation.

Amazon Neptune cluster - Example

An Amazon Neptune Cluster configured to export audit logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "NeptuneDBCluster": {
        "Type": "AWS::Neptune::DBCluster",
        "Properties": {
            "EnableCloudwatchLogsExports": [
                "audit"
            ]
        }
    }
}
```

YAML example

```
NeptuneDBCluster:
  Type: AWS::Neptune::DBCluster
  Properties:
    EnableCloudwatchLogsExports:
      - audit
```

CT.NEPTUNE.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   neptune_cluster_cloudwatch_audit_log_export_enabled
#
# Description:
#   This control checks whether an Amazon Neptune cluster is configured to send audit
 logs to Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::Neptune::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Neptune DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'EnableCloudwatchLogsExports' has not been provided or has been provided
 as an
#            empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'EnableCloudwatchLogsExports' has been provided as a non-empty list
#       And: 'EnableCloudwatchLogsExports' does not contain an entry with the value
 'audit'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains a Neptune DB cluster resource
#       And: 'EnableCloudwatchLogsExports' has been provided as a non-empty list
#       And: 'EnableCloudwatchLogsExports' contains an entry with the value 'audit'
#      Then: PASS

#
# Constants
#
let NEPTUNE_CLUSTER_TYPE = "AWS::Neptune::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let neptune_db_clusters = Resources.*[ Type == %NEPTUNE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule neptune_cluster_cloudwatch_audit_log_export_enabled when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %neptune_db_clusters not
 empty {
    check(%neptune_db_clusters.Properties)
        <<
        [CT.NEPTUNE.PR.4]: Require an Amazon Neptune DB cluster to enable Amazon
 CloudWatch log export for audit logs
        [FIX]: In the 'EnableCloudwatchLogsExports' parameter, set an entry to the
 value audit.
        >>
}

rule neptune_cluster_cloudwatch_audit_log_export_enabled when
 is_cfn_hook(%INPUT_DOCUMENT, %NEPTUNE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%NEPTUNE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.NEPTUNE.PR.4]: Require an Amazon Neptune DB cluster to enable Amazon
 CloudWatch log export for audit logs
        [FIX]: In the 'EnableCloudwatchLogsExports' parameter, set an entry to the
 value audit.
        >>
}

#
```

```
# Parameterized Rules
#
rule check(neptune_cluster) {
    %neptune_cluster {
        # Scenario 2
        EnableCloudwatchLogsExports exists
        EnableCloudwatchLogsExports is_list
        EnableCloudwatchLogsExports not empty

        # Scenarios 3 and 4
        some EnableCloudwatchLogsExports[*] == "audit"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NEPTUNE.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      EnableCloudwatchLogsExports:
```

```
      - audit
      - slowquery
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      EnableCloudwatchLogsExports:
      - slowquery
```

[CT.NEPTUNE.PR.5] Require an Amazon Neptune DB cluster to set a backup retention period greater than or equal to seven days

This control checks whether Amazon Neptune DB clusters have configured automatic backups with a retention period set to 7 or more days (>=7). The default retention period is one day.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Neptune::DBCluster`
- **CloudFormation guard rule:** CT.NEPTUNE.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NEPTUNE.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NEPTUNE.PR.5 example templates

Explanation

Amazon Neptune backs up your cluster volume automatically, and it retains restore data for the length of the backup retention period. Backups are continuous and incremental. You can restore to any point within the backup retention period, quickly. No performance impact or interruption of database service occurs as backup data is being written.

Remediation for rule failure

Set the `BackupRetentionPeriod` parameter to an integer value between 7 and 35 days (inclusive).

The examples that follow show how to implement this remediation.

Amazon Neptune cluster - Example

An Amazon Neptune Cluster configured with a backup retention period of seven (7) days. The example is shown in JSON and in YAML.

JSON example

```
{
    "NeptuneDBCluster": {
        "Type": "AWS::Neptune::DBCluster",
        "Properties": {
            "BackupRetentionPeriod": 7
        }
    }
}
```

YAML example

```
NeptuneDBCluster:
  Type: AWS::Neptune::DBCluster
  Properties:
    BackupRetentionPeriod: 7
```

CT.NEPTUNE.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   neptune_cluster_backup_retention_check
#
# Description:
#   This control checks whether Amazon Neptune DB clusters have configured automatic
 backups with a retention period set to 7 or more days (>=7). The default retention
 period is one day.
#
# Reports on:
#   AWS::Neptune::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Neptune DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'BackupRetentionPeriod' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'BackupRetentionPeriod' has been provided and set to an integer value
#            less than seven (< 7)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Neptune DB cluster resource
#       And: 'BackupRetentionPeriod' has been provided and set to an integer value
#            greater than or equal to seven (>=7)
#      Then: PASS
```

```
#
# Constants
#
let NEPTUNE_CLUSTER_TYPE = "AWS::Neptune::DBCluster"
let MINIMUM_BACKUP_RETENTION_PERIOD = 7
let INPUT_DOCUMENT = this

#
# Assignments
#
let neptune_db_clusters = Resources.*[ Type == %NEPTUNE_CLUSTER_TYPE ]

#
# Primary Rules
#
rule neptune_cluster_backup_retention_check when is_cfn_template(%INPUT_DOCUMENT)
                                                 %neptune_db_clusters not empty {
    check(%neptune_db_clusters.Properties)
        <<
        [CT.NEPTUNE.PR.5]: Require an Amazon Neptune DB cluster to set a backup
 retention period greater than or equal to seven days
        [FIX]: Set the 'BackupRetentionPeriod' parameter to an integer value between 7
 and 35 days (inclusive).
        >>
}

rule neptune_cluster_backup_retention_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NEPTUNE_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%NEPTUNE_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.NEPTUNE.PR.5]: Require an Amazon Neptune DB cluster to set a backup
 retention period greater than or equal to seven days
        [FIX]: Set the 'BackupRetentionPeriod' parameter to an integer value between 7
 and 35 days (inclusive).
        >>
}

#
# Parameterized Rules
#
rule check(neptune_cluster) {
    %neptune_cluster {
        # Scenario 2
        BackupRetentionPeriod exists
```

```
        # Scenarios 3 and 4
        BackupRetentionPeriod >= %MINIMUM_BACKUP_RETENTION_PERIOD
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NEPTUNE.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NeptuneDBCluster:
    Type: AWS::Neptune::DBCluster
    Properties:
      BackupRetentionPeriod: 7
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NeptuneDBCluster:
```

```
    Type: AWS::Neptune::DBCluster
    Properties:
      BackupRetentionPeriod: 1
```

AWS Network Firewall controls

Topics

- [CT.NETWORK-FIREWALL.PR.1] Require any AWS Network Firewall firewall policy to have an associated rule group
- [CT.NETWORK-FIREWALL.PR.2] Require any AWS Network Firewall firewall policy to drop or forward stateless full packets by default when they do not match a rule
- [CT.NETWORK-FIREWALL.PR.3] Require any AWS Network Firewall firewall policy to drop or forward fragmented packets by default when they do not match a stateless rule
- [CT.NETWORK-FIREWALL.PR.4] Require any AWS Network Firewall rule group to contain at least one rule
- [CT.NETWORK-FIREWALL.PR.5] Require an AWS Network Firewall firewall to be deployed across multiple Availability Zones

[CT.NETWORK-FIREWALL.PR.1] Require any AWS Network Firewall firewall policy to have an associated rule group

This control checks whether there is at least one stateful or stateless rule group associated with an AWS Network Firewall firewall policy.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::NetworkFirewall::FirewallPolicy`
- **CloudFormation guard rule:** CT.NETWORK-FIREWALL.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NETWORK-FIREWALL.PR.1 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NETWORK-FIREWALL.PR.1 example templates

Explanation

A firewall policy defines how your firewall monitors and handles traffic in Amazon Virtual Private Cloud (Amazon VPC). Configuration of stateless and stateful rule groups helps to filter packets and traffic flows, and to define the default traffic handling settings.

Remediation for rule failure

Within the `FirewallPolicy` definition, refer to one or more rule groups in `StatefulRuleGroupReferences` or `StatelessRuleGroupReferences`.

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall Policy - Example

AWS Network Firewall firewall policy configured with stateful and stateless rule group associations. The example is shown in JSON and in YAML.

JSON example

```
{
    "FirewallPolicy": {
        "Type": "AWS::NetworkFirewall::FirewallPolicy",
        "Properties": {
            "FirewallPolicyName": "sample-firewall-policy",
            "FirewallPolicy": {
                "StatelessDefaultActions": [
                    "aws:forward_to_sfe"
                ],
                "StatelessFragmentDefaultActions": [
                    "aws:drop"
                ],
                "StatefulRuleGroupReferences": [
                    {
                        "ResourceArn": {
                            "Ref": "StatefulRuleGroup"
                        }
                    }
                ],
```

```
                "StatelessRuleGroupReferences": [
                    {
                        "ResourceArn": {
                            "Ref": "StatelessRuleGroup"
                        },
                        "Priority": 100
                    }
                ]
            }
        }
    }
}
```

YAML example

```
FirewallPolicy:
  Type: AWS::NetworkFirewall::FirewallPolicy
  Properties:
    FirewallPolicyName: sample-firewall-policy
    FirewallPolicy:
      StatelessDefaultActions:
        - aws:forward_to_sfe
      StatelessFragmentDefaultActions:
        - aws:drop
      StatefulRuleGroupReferences:
        - ResourceArn: !Ref 'StatefulRuleGroup'
      StatelessRuleGroupReferences:
        - ResourceArn: !Ref 'StatelessRuleGroup'
          Priority: 100
```

CT.NETWORK-FIREWALL.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   netfw_policy_rule_group_associated_check
```

```
#
# Description:
#   This control checks whether there is at least one stateful or stateless rule group
 associated with an AWS Network Firewall firewall policy.
#
# Reports on:
#   AWS::NetworkFirewall::FirewallPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any NetworkFirewall firewall policy
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a NetworkFirewall firewall policy resource
#       And: 'StatefulRuleGroupReferences' has not been provided in 'FirewallPolicy'
#       And: 'StatelessRuleGroupReferences' has not been provided in 'FirewallPolicy'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a NetworkFirewall firewall policy resource
#       And: 'StatefulRuleGroupReferences' has not been provided in 'FirewallPolicy'
#       And: 'StatelessRuleGroupReferences' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a NetworkFirewall firewall policy resource
#       And: 'StatelessRuleGroupReferences' has not been provided in 'FirewallPolicy'
#       And: 'StatefulRuleGroupReferences' has been provided as an empty list
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a NetworkFirewall firewall policy resource
#       And: 'StatelessRuleGroupReferences' has been provided as an empty list
#       And: 'StatefulRuleGroupReferences' has been provided as an empty list
#      Then: FAIL
```

```
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a NetworkFirewall firewall policy resource
#       And: One or both of 'StatelessRuleGroupReferences' and
 'StatefulRuleGroupReferences' have been provided as a
#            non-empty list
#      Then: PASS

#
# Constants
#
let NETFW_FIREWALL_POLICY_TYPE = "AWS::NetworkFirewall::FirewallPolicy"
let INPUT_DOCUMENT = this

#
# Assignments
#
let netfw_firewall_policies = Resources.*[ Type == %NETFW_FIREWALL_POLICY_TYPE ]

#
# Primary Rules
#
rule netfw_policy_rule_group_associated_check when is_cfn_template(%INPUT_DOCUMENT)
                                                   %netfw_firewall_policies not empty {
    check(%netfw_firewall_policies.Properties)
        <<
        [CT.NETWORK-FIREWALL.PR.1]: Require any AWS Network Firewall firewall policy to
 have an associated rule group
            [FIX]: Within the 'FirewallPolicy' definition, refer to one or more rule
 groups in 'StatefulRuleGroupReferences' or 'StatelessRuleGroupReferences'.
        >>
}

rule netfw_policy_rule_group_associated_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NETFW_FIREWALL_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%NETFW_FIREWALL_POLICY_TYPE.resourceProperties)
        <<
        [CT.NETWORK-FIREWALL.PR.1]: Require any AWS Network Firewall firewall policy to
 have an associated rule group
            [FIX]: Within the 'FirewallPolicy' definition, refer to one or more rule
 groups in 'StatefulRuleGroupReferences' or 'StatelessRuleGroupReferences'.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(netfw_firewall_policy) {
    %netfw_firewall_policy {
        # Scenario 2
        FirewallPolicy exists
        FirewallPolicy is_struct

        FirewallPolicy {
            # Scenario 3, 4, 5 and 6
            StatefulRuleGroupReferences exists or
            StatelessRuleGroupReferences exists

            check_property_is_list_and_not_empty(StatefulRuleGroupReferences) or
            check_property_is_list_and_not_empty(StatelessRuleGroupReferences)
        }
    }
}

rule check_property_is_list_and_not_empty(property) {
    %property {
        this is_list
        this not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NETWORK-FIREWALL.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  StatefulRuleGroup:
    Type: AWS::NetworkFirewall::RuleGroup
    Properties:
      RuleGroupName:
        Fn::Sub: ${AWS::StackName}-stateful-example
      Type: STATEFUL
      RuleGroup:
        RulesSource:
          RulesString: pass tcp 10.20.20.0/24 45400:45500 < 10.10.10.0/24 5203
 (msg:"test";sid:1;rev:1;)
      Capacity: 100
  StatelessRuleGroup:
    Type: AWS::NetworkFirewall::RuleGroup
    Properties:
      RuleGroupName:
        Fn::Sub: ${AWS::StackName}-stateless-example
      Type: STATELESS
      RuleGroup:
        RulesSource:
          StatelessRulesAndCustomActions:
            StatelessRules: []
      Capacity: 100
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:forward_to_sfe
        StatelessFragmentDefaultActions:
        - aws:drop
        StatefulRuleGroupReferences:
        - ResourceArn:
            Ref: StatefulRuleGroup
```

```
        StatelessRuleGroupReferences:
        - ResourceArn:
            Ref: StatelessRuleGroup
          Priority: 100
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:forward_to_sfe
        StatelessFragmentDefaultActions:
        - aws:drop
```

[CT.NETWORK-FIREWALL.PR.2] Require any AWS Network Firewall firewall policy to drop or forward stateless full packets by default when they do not match a rule

This control checks whether an AWS Network Firewall firewall policy is configured with a user-defined stateless default action for full packets.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::NetworkFirewall::FirewallPolicy`
- **CloudFormation guard rule:** CT.NETWORK-FIREWALL.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NETWORK-FIREWALL.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NETWORK-FIREWALL.PR.2 example templates

Explanation

A firewall policy defines how your firewall monitors and handles traffic in Amazon VPC. You configure stateless and stateful rule groups to filter packets and traffic flows. Defaulting to `Pass` can allow unintended traffic.

Remediation for rule failure

Within `FirewallPolicy`, include one of `aws:drop` or `aws:forward_to_sfe` in `StatelessDefaultActions`.

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall Policy - Example One

AWS Network Firewall firewall policy configured with a stateless default action to drop full packets. The example is shown in JSON and in YAML.

JSON example

```
{
    "FirewallPolicy": {
        "Type": "AWS::NetworkFirewall::FirewallPolicy",
        "Properties": {
            "FirewallPolicyName": {
                "Fn::Sub": "${AWS::StackName}-sample"
            },
            "FirewallPolicy": {
                "StatelessFragmentDefaultActions": [
                    "aws:forward_to_sfe"
                ],
                "StatelessDefaultActions": [
                    "aws:drop"
                ]
            }
        }
```

```
    }
}
```

YAML example

```
FirewallPolicy:
  Type: AWS::NetworkFirewall::FirewallPolicy
  Properties:
    FirewallPolicyName: !Sub '${AWS::StackName}-sample'
    FirewallPolicy:
      StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
      StatelessDefaultActions:
        - aws:drop
```

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall Policy - Example Two

AWS Network Firewall firewall policy configured with a stateless default action to forward full packets to the stateful rule engine for further inspection. The example is shown in JSON and in YAML.

JSON example

```
{
    "FirewallPolicy": {
        "Type": "AWS::NetworkFirewall::FirewallPolicy",
        "Properties": {
            "FirewallPolicyName": {
                "Fn::Sub": "${AWS::StackName}-sample"
            },
            "FirewallPolicy": {
                "StatelessFragmentDefaultActions": [
                    "aws:forward_to_sfe"
                ],
                "StatelessDefaultActions": [
                    "aws:forward_to_sfe"
```

```
                ]
            }
        }
    }
}
```

YAML example

```
FirewallPolicy:
  Type: AWS::NetworkFirewall::FirewallPolicy
  Properties:
    FirewallPolicyName: !Sub '${AWS::StackName}-sample'
    FirewallPolicy:
      StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
      StatelessDefaultActions:
        - aws:forward_to_sfe
```

CT.NETWORK-FIREWALL.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   netfw_policy_default_action_full_packets_check
#
# Description:
#   This control checks whether an AWS Network Firewall firewall policy is configured
 with a user-defined stateless default action for full packets.
#
# Reports on:
#   AWS::NetworkFirewall::FirewallPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Network Firewall firewall policy
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessDefaultActions' has not been provided in 'FirewallPolicy'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessDefaultActions' has been provided in 'FirewallPolicy' as an
 empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessDefaultActions' has been provided in 'FirewallPolicy' as a list
 that does not contain
#             one of 'aws:drop' or 'aws:forward_to_sfe'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessDefaultActions' has been provided in 'FirewallPolicy' as a list
 that contains either
#            'aws:drop' or 'aws:forward_to_sfe'
#      Then: PASS



#
# Constants
#
let NETFW_FIREWALL_POLICY_TYPE = "AWS::NetworkFirewall::FirewallPolicy"
let INPUT_DOCUMENT = this
let ALLOWED_STATELESS_ACTIONS_LIST = [ "aws:drop", "aws:forward_to_sfe" ]

#
# Assignments
```

```
#
let netfw_firewall_policies = Resources.*[ Type == %NETFW_FIREWALL_POLICY_TYPE ]

#
# Primary Rules
#
rule netfw_policy_default_action_full_packets_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %netfw_firewall_policies
 not empty {
    check(%netfw_firewall_policies.Properties)
        <<
        [CT.NETWORK-FIREWALL.PR.2]: Require any AWS Network Firewall firewall policy to
 drop or forward stateless full packets by default when they do not match a rule
        [FIX]: Within 'FirewallPolicy', include one of 'aws:drop' or
 'aws:forward_to_sfe' in 'StatelessDefaultActions'.
        >>
}

rule netfw_policy_default_action_full_packets_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NETFW_FIREWALL_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%NETFW_FIREWALL_POLICY_TYPE.resourceProperties)
        <<
        [CT.NETWORK-FIREWALL.PR.2]: Require any AWS Network Firewall firewall policy to
 drop or forward stateless full packets by default when they do not match a rule
        [FIX]: Within 'FirewallPolicy', include one of 'aws:drop' or
 'aws:forward_to_sfe' in 'StatelessDefaultActions'.
        >>
}

#
# Parameterized Rules
#
rule check(netfw_firewall_policy) {
    %netfw_firewall_policy {
        # Scenario 2
        FirewallPolicy exists
        FirewallPolicy is_struct

        FirewallPolicy {
            StatelessDefaultActions exists
            # Scenario 3
            StatelessDefaultActions is_list
            StatelessDefaultActions not empty
```

```
            # Scenario 4 and 5
            some StatelessDefaultActions[*] {
                this in %ALLOWED_STATELESS_ACTIONS_LIST
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NETWORK-FIREWALL.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
        StatelessDefaultActions:
```

```
        - aws:drop
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessFragmentDefaultActions:
        - aws:pass
        StatelessDefaultActions:
        - aws:pass
```

[CT.NETWORK-FIREWALL.PR.3] Require any AWS Network Firewall firewall policy to drop or forward fragmented packets by default when they do not match a stateless rule

This control checks whether an AWS Network Firewall firewall policy is configured with a default action to drop or forward fragmented packets, when the packets do not match a stateless rule.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::NetworkFirewall::FirewallPolicy`
- **CloudFormation guard rule:** CT.NETWORK-FIREWALL.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NETWORK-FIREWALL.PR.3 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NETWORK-FIREWALL.PR.3 example templates

Explanation

A firewall policy defines how your firewall monitors and handles traffic in Amazon VPC. You configure stateless and stateful rule groups to filter packets and traffic flows. Defaulting to `Pass` can allow unintended traffic.

Remediation for rule failure

Within `FirewallPolicy`, include one of `aws:drop` or `aws:forward_to_sfe` in `StatelessFragmentDefaultActions`.

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall Policy - Example One

AWS Network Firewall firewall policy configured with a stateless default action to drop fragmented packets. The example is shown in JSON and in YAML.

JSON example

```
{
    "FirewallPolicy": {
        "Type": "AWS::NetworkFirewall::FirewallPolicy",
        "Properties": {
            "FirewallPolicyName": {
                "Fn::Sub": "${AWS::StackName}-sample"
            },
            "FirewallPolicy": {
                "StatelessDefaultActions": [
                    "aws:forward_to_sfe"
                ],
                "StatelessFragmentDefaultActions": [
                    "aws:drop"
                ]
            }
        }
    }
}
```

YAML example

```
FirewallPolicy:
  Type: AWS::NetworkFirewall::FirewallPolicy
  Properties:
    FirewallPolicyName: !Sub '${AWS::StackName}-sample'
    FirewallPolicy:
      StatelessDefaultActions:
        - aws:forward_to_sfe
      StatelessFragmentDefaultActions:
        - aws:drop
```

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall Policy - Example Two

AWS Network Firewall firewall policy configured with a stateless default action to forward fragmented packets to the stateful rule engine for further inspection. The example is shown in JSON and in YAML.

JSON example

```
{
    "FirewallPolicy": {
        "Type": "AWS::NetworkFirewall::FirewallPolicy",
        "Properties": {
            "FirewallPolicyName": {
                "Fn::Sub": "${AWS::StackName}-sample"
            },
            "FirewallPolicy": {
                "StatelessDefaultActions": [
                    "aws:forward_to_sfe"
                ],
                "StatelessFragmentDefaultActions": [
                    "aws:forward_to_sfe"
                ]
            }
```

```
        }
    }
}
```

YAML example

```
FirewallPolicy:
  Type: AWS::NetworkFirewall::FirewallPolicy
  Properties:
    FirewallPolicyName: !Sub '${AWS::StackName}-sample'
    FirewallPolicy:
      StatelessDefaultActions:
        - aws:forward_to_sfe
      StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
```

CT.NETWORK-FIREWALL.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   netfw_policy_default_action_fragment_packets_check
#
# Description:
#   This control checks whether an AWS Network Firewall firewall policy is configured
 with a default action to drop or forward fragmented packets, when the packets do not
 match a stateless rule.
#
# Reports on:
#   AWS::NetworkFirewall::FirewallPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Network Firewall firewall policy
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessFragmentDefaultActions' has not been provided in
 'FirewallPolicy'
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessFragmentDefaultActions' has been provided in 'FirewallPolicy' as
 an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessFragmentDefaultActions' has been provided in 'FirewallPolicy' as
 a list that does not contain
#             one of 'aws:drop' or 'aws:forward_to_sfe'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall policy resource
#       And: 'StatelessFragmentDefaultActions' has been provided in 'FirewallPolicy' as
 a list that contains either
#            'aws:drop' or 'aws:forward_to_sfe'
#      Then: PASS


#
# Constants
#
let NETFW_FIREWALL_POLICY_TYPE = "AWS::NetworkFirewall::FirewallPolicy"
let ALLOWED_STATELESS_FRAGMENT_ACTIONS_LIST = [ "aws:drop", "aws:forward_to_sfe" ]
let INPUT_DOCUMENT = this

#
# Assignments
```

```
#
let netfw_firewall_policies = Resources.*[ Type == %NETFW_FIREWALL_POLICY_TYPE ]

#
# Primary Rules
#
rule netfw_policy_default_action_fragment_packets_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                              %netfw_firewall_policies
 not empty {
    check(%netfw_firewall_policies.Properties)
        <<
        [CT.NETWORK-FIREWALL.PR.3]: Require any AWS Network Firewall firewall policy to
 drop or forward fragmented packets by default when they do not match a stateless rule
        [FIX]: Within 'FirewallPolicy', include one of 'aws:drop' or
 'aws:forward_to_sfe' in 'StatelessFragmentDefaultActions'.
        >>
}

rule netfw_policy_default_action_fragment_packets_check when
 is_cfn_hook(%INPUT_DOCUMENT, %NETFW_FIREWALL_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%NETFW_FIREWALL_POLICY_TYPE.resourceProperties)
        <<
        [CT.NETWORK-FIREWALL.PR.3]: Require any AWS Network Firewall firewall policy to
 drop or forward fragmented packets by default when they do not match a stateless rule
        [FIX]: Within 'FirewallPolicy', include one of 'aws:drop' or
 'aws:forward_to_sfe' in 'StatelessFragmentDefaultActions'.
        >>
}

#
# Parameterized Rules
#
rule check(netfw_firewall_policy) {
    %netfw_firewall_policy {
        # Scenario 2
        FirewallPolicy exists
        FirewallPolicy is_struct

        FirewallPolicy {
            # Scenario 2 and 3
            StatelessFragmentDefaultActions exists
            StatelessFragmentDefaultActions is_list
            StatelessFragmentDefaultActions not empty
```

```
            # Scenario 4 and 5
            some StatelessFragmentDefaultActions[*] {
                this in %ALLOWED_STATELESS_FRAGMENT_ACTIONS_LIST
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NETWORK-FIREWALL.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:forward_to_sfe
        StatelessFragmentDefaultActions:
```

```
        - aws:drop
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:pass
        StatelessFragmentDefaultActions:
        - aws:pass
```

[CT.NETWORK-FIREWALL.PR.4] Require any AWS Network Firewall rule group to contain at least one rule

This control checks whether an AWS Network Firewall stateless rule group contains rules.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::NetworkFirewall::RuleGroup`
- **CloudFormation guard rule:** CT.NETWORK-FIREWALL.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NETWORK-FIREWALL.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NETWORK-FIREWALL.PR.4 example templates

Explanation

A rule group contains rules that define how your firewall processes traffic in your VPC. An empty, stateless rule group, when present in a firewall policy, might give the impression that the rule group will process traffic. However, when the stateless rule group is empty, it does not process traffic.

Usage considerations

- This control applies only to AWS Network Firewall stateless rule groups.

Remediation for rule failure

Provide one or more AWS Network Firewall stateless rules within the `RuleGroup.RulesSource.StatelessRulesAndCustomActions.StatelessRules` property.

The examples that follow show how to implement this remediation.

AWS Network Firewall Rule Group - Example

AWS Network Firewall rule group configured with a stateless rule. The example is shown in JSON and in YAML.

JSON example

```
{
    "NetworkFirewallRuleGroup": {
        "Type": "AWS::NetworkFirewall::RuleGroup",
        "Properties": {
            "RuleGroupName": {
                "Fn::Sub": "${AWS::StackName}-sample"
            },
            "Capacity": 100,
            "Description": "Sample rule group",
            "Type": "STATELESS",
            "RuleGroup": {
                "RulesSource": {
                    "StatelessRulesAndCustomActions": {
                        "StatelessRules": [
                            {
```

```
                                "RuleDefinition": {
                                    "MatchAttributes": {
                                        "Sources": [
                                            {
                                                "AddressDefinition": "0.0.0.0/0"
                                            }
                                        ],
                                        "Destinations": [
                                            {
                                                "AddressDefinition": "10.0.0.0/8"
                                            }
                                        ],
                                        "SourcePorts": [
                                            {
                                                "FromPort": 15000,
                                                "ToPort": 30000
                                            }
                                        ],
                                        "DestinationPorts": [
                                            {
                                                "FromPort": 443,
                                                "ToPort": 443
                                            }
                                        ],
                                        "Protocols": [
                                            6
                                        ]
                                    },
                                    "Actions": [
                                        "aws:forward_to_sfe"
                                    ]
                                },
                                "Priority": 1
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

YAML example

```
NetworkFirewallRuleGroup:
  Type: AWS::NetworkFirewall::RuleGroup
  Properties:
    RuleGroupName: !Sub '${AWS::StackName}-sample'
    Capacity: 100
    Description: Sample rule group
    Type: STATELESS
    RuleGroup:
      RulesSource:
        StatelessRulesAndCustomActions:
          StatelessRules:
            - RuleDefinition:
                MatchAttributes:
                  Sources:
                    - AddressDefinition: '0.0.0.0/0'
                  Destinations:
                    - AddressDefinition: 10.0.0.0/8
                  SourcePorts:
                    - FromPort: 15000
                      ToPort: 30000
                  DestinationPorts:
                    - FromPort: 443
                      ToPort: 443
                  Protocols:
                    - 6
                Actions:
                  - aws:forward_to_sfe
              Priority: 1
```

CT.NETWORK-FIREWALL.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   netfw_stateless_rule_group_not_empty_check
```

```
#
# Description:
#   This control checks whether an AWS Network Firewall stateless rule group contains
 rules.
#
# Reports on:
#   AWS::NetworkFirewall::RuleGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Network Firewall rule group
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall rule group resource
#       And: 'Type' is not equal to 'STATELESS'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall rule group resource
#       And: 'Type' is 'STATELESS'
#       And: 'RuleGroup.RulesSource.StatelessRulesAndCustomActions' has not been
 provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall rule group resource
#       And: 'Type' is 'STATELESS'
#       And: 'RuleGroup.RulesSource.StatelessRulesAndCustomActions' has been provided
#       And: 'StatelessRules' has not been provided within
 'StatelessRulesAndCustomActions' or has been provided with
#            an empty list value
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall rule group resource
```

```
#       And: 'Type' is 'STATELESS'
#       And: 'RuleGroup.RulesSource.StatelessRulesAndCustomActions' has been provided
#       And: 'StatelessRules' has been provided within 'StatelessRulesAndCustomActions'
 as a non-empty list value
#      Then: PASS

#
# Constants
#
let NETFW_RULE_GROUP_TYPE = "AWS::NetworkFirewall::RuleGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let netfw_rule_group = Resources.*[ Type == %NETFW_RULE_GROUP_TYPE ]

#
# Primary Rules
#
rule netfw_stateless_rule_group_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %netfw_rule_group not empty {
    check(%netfw_rule_group.Properties)
        <<
        [CT.NETWORK-FIREWALL.PR.4]: Require any AWS Network Firewall rule group to
 contain at least one rule
        [FIX]: Provide one or more AWS Network Firewall stateless rules within the
 'RuleGroup.RulesSource.StatelessRulesAndCustomActions.StatelessRules' property.
        >>
}

rule netfw_stateless_rule_group_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NETFW_RULE_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%NETFW_RULE_GROUP_TYPE.resourceProperties)
        <<
        [CT.NETWORK-FIREWALL.PR.4]: Require any AWS Network Firewall rule group to
 contain at least one rule
        [FIX]: Provide one or more AWS Network Firewall stateless rules within the
 'RuleGroup.RulesSource.StatelessRulesAndCustomActions.StatelessRules' property.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(netfw_rule_group) {
    %netfw_rule_group[
        # Scenario 2
        Type exists
        Type == "STATELESS"
    ] {
        # Scenario 3
        RuleGroup exists
        RuleGroup is_struct
        RuleGroup {
            RulesSource exists
            RulesSource is_struct
            RulesSource {
                StatelessRulesAndCustomActions exists
                StatelessRulesAndCustomActions is_struct
                StatelessRulesAndCustomActions {
                    # Scenarios 4 and 5
                    StatelessRules exists
                    StatelessRules is_list
                    StatelessRules not empty
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NETWORK-FIREWALL.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NetworkFirewallRuleGroup:
    Type: AWS::NetworkFirewall::RuleGroup
    Properties:
      RuleGroupName:
        Fn::Sub: ${AWS::StackName}-example
      Capacity: 100
      Description: Example rule group
      Type: STATELESS
      RuleGroup:
        RulesSource:
          StatelessRulesAndCustomActions:
            StatelessRules:
            - RuleDefinition:
                MatchAttributes:
                  Sources:
                  - AddressDefinition: 0.0.0.0/0
                  Destinations:
                  - AddressDefinition: 10.0.0.0/8
                  SourcePorts:
                  - FromPort: 15000
                    ToPort: 30000
                  DestinationPorts:
                  - FromPort: 443
                    ToPort: 443
                  Protocols:
                  - 6
                Actions:
                - aws:forward_to_sfe
              Priority: 1
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NetworkFirewallRuleGroup:
    Type: AWS::NetworkFirewall::RuleGroup
    Properties:
      RuleGroupName:
        Fn::Sub: ${AWS::StackName}-example
      Capacity: 100
      Description: Example rule group
      Type: STATELESS
      RuleGroup:
        RulesSource:
          StatelessRulesAndCustomActions:
            StatelessRules: []
```

[CT.NETWORK-FIREWALL.PR.5] Require an AWS Network Firewall firewall to be deployed across multiple Availability Zones

This control checks whether an AWS Network Firewall firewall is deployed across multiple Availability Zones (AZs), to permit automatic failover between AZs.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::NetworkFirewall::Firewall`
- **CloudFormation guard rule:** CT.NETWORK-FIREWALL.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.NETWORK-FIREWALL.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.NETWORK-FIREWALL.PR.5 example templates

Explanation

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple Availability Zones (AZs), physically separated and isolated. These AZs are connected by low-latency, high-throughput, and highly redundant networking. You can design and operate applications and databases that fail over between Availability Zones without interruption, automatically. Availability Zones are more highly available, fault tolerant, and scalable than traditional single- or multiple-datacenter infrastructures.

Remediation for rule failure

In the SubnetMappings parameter, provide at least two entries that refer to subnets in different Availability Zones.

The examples that follow show how to implement this remediation.

AWS Network Firewall Firewall - Example

An AWS Network Firewall firewall configured to deploy across two subnets in different Availability Zones. The example is shown in JSON and in YAML.

JSON example

```
{
    "Firewall": {
        "Type": "AWS::NetworkFirewall::Firewall",
        "Properties": {
            "FirewallName": "SampleFirewall",
            "FirewallPolicyArn": {
                "Ref": "FirewallPolicy"
            },
            "VpcId": {
                "Ref": "VPC"
            },
            "Description": "Sample firewall",
            "SubnetMappings": [
                {
                    "SubnetId": {
                        "Ref": "SubnetOne"
                    }
                },
                {
                    "SubnetId": {
                        "Ref": "SubnetTwo"
```

```
                    }
                }
            ]
        }
    }
}
```

YAML example

```
Firewall:
  Type: AWS::NetworkFirewall::Firewall
  Properties:
    FirewallName: SampleFirewall
    FirewallPolicyArn: !Ref 'FirewallPolicy'
    VpcId: !Ref 'VPC'
    Description: Sample firewall
    SubnetMappings:
      - SubnetId: !Ref 'SubnetOne'
      - SubnetId: !Ref 'SubnetTwo'
```

CT.NETWORK-FIREWALL.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   netfw_multi_az_enabled_check
#
# Description:
#   This control checks whether an AWS Network Firewall firewall is deployed across
 multiple Availability Zones (AZs), to permit automatic failover between AZs.
#
# Reports on:
#   AWS::NetworkFirewall::Firewall
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Network Firewall firewall
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall resource
#       And: 'SubnetMappings' has not been specified or specified as an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall resource
#       And: 'SubnetMappings' has been specified
#       And: The number of entries in 'SubnetMappings' is less than two (< 2)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Network Firewall firewall resource
#       And: 'SubnetMappings' has been specified
#       And: The number of entries in 'SubnetMappings' is greater than or equal to two
 (>= 2)
#      Then: PASS

#
# Constants
#
let NETFW_FIREWALL_FIREWALL_TYPE = "AWS::NetworkFirewall::Firewall"
let INPUT_DOCUMENT = this

#
# Assignments
#
let netfw_firewalls = Resources.*[ Type == %NETFW_FIREWALL_FIREWALL_TYPE ]

#
# Primary Rules
#
rule netfw_multi_az_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
```

```
                                        %netfw_firewalls not empty {
    check(%netfw_firewalls.Properties)
        <<
        [CT.NETWORK-FIREWALL.PR.5]: Require an AWS Network Firewall firewall to be
 deployed across multiple Availability Zones
        [FIX]: In the SubnetMappings parameter, provide at least two entries that refer
 to subnets in different Availability Zones.
        >>
}

rule netfw_multi_az_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %NETFW_FIREWALL_FIREWALL_TYPE) {
    check(%INPUT_DOCUMENT.%NETFW_FIREWALL_FIREWALL_TYPE.resourceProperties)
        <<
        [CT.NETWORK-FIREWALL.PR.5]: Require an AWS Network Firewall firewall to be
 deployed across multiple Availability Zones
        [FIX]: In the SubnetMappings parameter, provide at least two entries that refer
 to subnets in different Availability Zones.
        >>
}

#
# Parameterized Rules
#
rule check(netfw_firewall) {
    %netfw_firewall {
        # Scenario 2
        SubnetMappings exists
        SubnetMappings is_list
        SubnetMappings not empty

        # Scenarios 3 and 4
        SubnetMappings[0] exists
        SubnetMappings[1] exists
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.NETWORK-FIREWALL.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ""
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ""
  FirewallPolicy:
```

```
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example-firewall-policy
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:forward_to_sfe
        StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
      Description: Example firewall policy
  Firewall:
    Type: AWS::NetworkFirewall::Firewall
    Properties:
      FirewallName:
        Fn::Sub: ${AWS::StackName}-example-firewall
      FirewallPolicyArn:
        Ref: FirewallPolicy
      VpcId:
        Ref: VPC
      Description: Example firewall
      SubnetMappings:
      - SubnetId:
          Ref: SubnetOne
      - SubnetId:
          Ref: SubnetTwo
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ""
  FirewallPolicy:
    Type: AWS::NetworkFirewall::FirewallPolicy
    Properties:
      FirewallPolicyName:
        Fn::Sub: ${AWS::StackName}-example-firewall-policy
      FirewallPolicy:
        StatelessDefaultActions:
        - aws:forward_to_sfe
        StatelessFragmentDefaultActions:
        - aws:forward_to_sfe
      Description: Example firewall policy
  Firewall:
    Type: AWS::NetworkFirewall::Firewall
    Properties:
      FirewallName:
        Fn::Sub: ${AWS::StackName}-example-firewall
      FirewallPolicyArn:
        Ref: FirewallPolicy
      VpcId:
        Ref: VPC
      Description: Example firewall
      SubnetMappings:
      - SubnetId:
          Ref: SubnetOne
```

Amazon OpenSearch controls

Topics

- [CT.OPENSEARCH.PR.1] Require an Elasticsearch domain to encrypt data at rest
- [CT.OPENSEARCH.PR.2] Require an Elasticsearch domain to be created in a user-specified Amazon VPC
- [CT.OPENSEARCH.PR.3] Require an Elasticsearch domain to encrypt data sent between nodes
- [CT.OPENSEARCH.PR.4] Require an Elasticsearch domain to send error logs to Amazon CloudWatch Logs

- [CT.OPENSEARCH.PR.5] Require an Elasticsearch domain to send audit logs to Amazon CloudWatch Logs
- [CT.OPENSEARCH.PR.6] Require an Elasticsearch domain to have zone awareness and at least three data nodes
- [CT.OPENSEARCH.PR.7] Require an Elasticsearch domain to have at least three dedicated master nodes
- [CT.OPENSEARCH.PR.8] Require an Elasticsearch Service domain to use TLSv1.2
- [CT.OPENSEARCH.PR.9] Require an Amazon OpenSearch Service domain to encrypt data at rest
- [CT.OPENSEARCH.PR.10] Require an Amazon OpenSearch Service domain to be created in a user-specified Amazon VPC
- [CT.OPENSEARCH.PR.11] Require an Amazon OpenSearch Service domain to encrypt data sent between nodes
- [CT.OPENSEARCH.PR.12] Require an Amazon OpenSearch Service domain to send error logs to Amazon CloudWatch Logs
- [CT.OPENSEARCH.PR.13] Require an Amazon OpenSearch Service domain to send audit logs to Amazon CloudWatch Logs
- [CT.OPENSEARCH.PR.14] Require an Amazon OpenSearch Service domain to have zone awareness and at least three data nodes
- [CT.OPENSEARCH.PR.15] Require an Amazon OpenSearch Service domain to use fine-grained access control
- [CT.OPENSEARCH.PR.16] Require an Amazon OpenSearch Service domain to use TLSv1.2

[CT.OPENSEARCH.PR.1] Require an Elasticsearch domain to encrypt data at rest

This control checks whether Elasticsearch domains have encryption-at-rest enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.1 example templates

Explanation

For an added layer of security for your sensitive data in OpenSearch, you should configure your OpenSearch to be encrypted at rest. Elasticsearch domains offer encryption of data at rest. The feature uses AWS KMS to store and manage your encryption keys. To perform the encryption, it uses the Advanced Encryption Standard algorithm with 256-bit keys (AES-256).

Remediation for rule failure

Within `EncryptionAtRestOptions`, set `Enabled` to `true`.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured with encryption-at-rest enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
```

```
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "EncryptionAtRestOptions": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
```

```
          Resource: '*'
    EncryptionAtRestOptions:
      Enabled: true
```

CT.OPENSEARCH.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticsearch_encrypted_at_rest_check
#
# Description:
#   This control checks whether Elasticsearch domains have encryption-at-rest enabled.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'EncryptionAtRestOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'EncryptionAtRestOptions' has been provided
#       And: 'Enabled' in 'EncryptionAtRestOptions' has not been provided or provided
```

```
#            and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'EncryptionAtRestOptions' has been provided
#       And: 'Enabled' in 'EncryptionAtRestOptions' has been provided and set to
 bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_encrypted_at_rest_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %elasticsearch_domains not empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.1]: Require an Elasticsearch domain to encrypt data at rest
            [FIX]: Within 'EncryptionAtRestOptions', set 'Enabled' to 'true'.
        >>
}

rule elasticsearch_encrypted_at_rest_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.1]: Require an Elasticsearch domain to encrypt data at rest
            [FIX]: Within 'EncryptionAtRestOptions', set 'Enabled' to 'true'.
        >>
}

#
# Parameterized Rules
```

```
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        EncryptionAtRestOptions exists
        EncryptionAtRestOptions is_struct

        EncryptionAtRestOptions {
            # Scenarios 3 and 4
            Enabled exists
            Enabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
```

```
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      EncryptionAtRestOptions:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
```

```
        Principal:
          AWS: '*'
        Action: es:*
        Resource: '*'
    EncryptionAtRestOptions:
      Enabled: false
```

[CT.OPENSEARCH.PR.2] Require an Elasticsearch domain to be created in a user-specified Amazon VPC

This control checks whether Elasticsearch domains are configured with VPC option settings that specify a target Amazon VPC.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.2 example templates

Explanation

Elasticsearch domains deployed within a VPC can communicate with VPC resources over the private AWS network, without the need to traverse the public internet. This configuration increases the security posture by limiting access to the data in transit. VPCs provide a number of network controls that help create secure access to Elasticsearch domains, including network ACLs and security groups. Security Hub CSPM recommends that you migrate public Elasticsearch domains to VPCs to take advantage of these controls.

Remediation for rule failure

Within `VPCOptions`, set `SubnetIds` to a list with one or more Amazon EC2 subnet IDs.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured to deploy within an Amazon VPC by means of VPC option settings. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "VPCOptions": {
                "SubnetIds": [
```

```
                    {
                        "Ref": "Subnet"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    VPCOptions:
      SubnetIds:
        - !Ref 'Subnet'
```

CT.OPENSEARCH.PR.2 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   elasticsearch_in_vpc_only_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured with VPC option
 settings that specify a target Amazon VPC.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'VPCOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'VPCOptions' has been provided
#       And: 'SubnetIds' in 'VPCOptions' has not been provided or has been provided
#            as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'VPCOptions' has been provided
#       And: 'SubnetIds' in 'VPCOptions' has been provided as a list with one or more
 values
#      Then: PASS
```

```
#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_in_vpc_only_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %elasticsearch_domains not empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.2]: Require an Elasticsearch domain to be created in a user-
specified Amazon VPC
            [FIX]: Within 'VPCOptions', set 'SubnetIds' to a list with one or more
 Amazon EC2 subnet IDs.
        >>
}

rule elasticsearch_in_vpc_only_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.2]: Require an Elasticsearch domain to be created in a user-
specified Amazon VPC
            [FIX]: Within 'VPCOptions', set 'SubnetIds' to a list with one or more
 Amazon EC2 subnet IDs.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        VPCOptions exists
        VPCOptions is_struct
```

```
        VPCOptions {
            # Scenarios 3 and 4
            SubnetIds exists
            SubnetIds is_list
            SubnetIds not empty
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
```

```
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      VPCOptions:
        SubnetIds:
        - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
```

```
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

[CT.OPENSEARCH.PR.3] Require an Elasticsearch domain to encrypt data sent between nodes

This control checks whether Elasticsearch domains have node-to-node encryption enabled.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.3 example templates

Explanation

HTTPS (TLS) can help prevent potential attackers from eavesdropping on or manipulating network traffic using person-in-the-middle, or similar, attacks. Only encrypted connections over HTTPS

(TLS) should be allowed. Enabling node-to-node encryption for Elasticsearch domains ensures that intra-cluster communications are encrypted in transit.

Usage considerations

- A performance penalty may be associated with this configuration. You should be aware of and test the performance trade-offs before enabling this option.

Remediation for rule failure

Within `NodeToNodeEncryptionOptions`, set `Enabled` to `true`.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured with node-to-node encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
```

```
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "NodeToNodeEncryptionOptions": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
```

CT.OPENSEARCH.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticsearch_node_to_node_encryption_check
#
# Description:
#   This control checks whether Elasticsearch domains have node-to-node encryption
 enabled.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'NodeToNodeEncryptionOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'NodeToNodeEncryptionOptions' has been provided
#       And: 'Enabled' in 'NodeToNodeEncryptionOptions' has not been provided or has
 been provided
#            and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
```

```
#       And: 'NodeToNodeEncryptionOptions' has been provided
#       And: 'Enabled' in 'NodeToNodeEncryptionOptions' has been provided and set to a
#            value of bool(true)
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_node_to_node_encryption_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %elasticsearch_domains not empty
 {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.3]: Require an Elasticsearch domain to encrypt data sent
 between nodes
            [FIX]: Within 'NodeToNodeEncryptionOptions', set 'Enabled' to 'true'.
        >>
}

rule elasticsearch_node_to_node_encryption_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.3]: Require an Elasticsearch domain to encrypt data sent
 between nodes
            [FIX]: Within 'NodeToNodeEncryptionOptions', set 'Enabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
```

```
    %elasticsearch_domain {
        # Scenario 2
        NodeToNodeEncryptionOptions exists
        NodeToNodeEncryptionOptions is_struct

        NodeToNodeEncryptionOptions {
            # Scenarios 3 and 4
            Enabled exists
            Enabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
```

```
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
```

```
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions: {}
```

[CT.OPENSEARCH.PR.4] Require an Elasticsearch domain to send error logs to Amazon CloudWatch Logs

This control checks whether Elasticsearch domains are configured to send error logs to an Amazon CloudWatch Logs log group.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.4 example templates

Explanation

Enable error logs (ES_APPLICATION_LOGS) for Elasticsearch domains and send those logs to CloudWatch Logs for retention and response. Domain error logs can assist with security and access audits, and can help to diagnose availability issues.

Remediation for rule failure

Within `LogPublishingOptions`, provide an `ES_APPLICATION_LOGS` configuration, set `Enabled` to `true`, and set `CloudWatchLogsLogGroupArn` to the ARN of a valid Amazon CloudWatch Logs log group.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured to send error logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "LogPublishingOptions": {
                "ES_APPLICATION_LOGS": {
                    "CloudWatchLogsLogGroupArn": {
                        "Fn::GetAtt": [
                            "LogGroup",
                            "Arn"
                        ]
                    },
```

```
                    "Enabled": true
                }
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    LogPublishingOptions:
      ES_APPLICATION_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt 'LogGroup.Arn'
        Enabled: true
```

CT.OPENSEARCH.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   elasticsearch_application_logging_enabled_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured to send error logs
 to an Amazon CloudWatch Logs log group.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been provided
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has not been provided or provided and
 set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has not been provided
 or provided
```

```
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has not been provided
 or provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has not been provided or provided and
 set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has been provided as
 a non-empty string
#            or valid local reference
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has been provided as
 a non-empty string
#            or valid local reference
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
```

```
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_application_logging_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %elasticsearch_domains not
 empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.4]: Require an Elasticsearch domain to send error logs to
 Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'ES_APPLICATION_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

rule elasticsearch_application_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.4]: Require an Elasticsearch domain to send error logs to
 Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'ES_APPLICATION_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        LogPublishingOptions exists
        LogPublishingOptions is_struct

        LogPublishingOptions {
            # Scenario 3
            ES_APPLICATION_LOGS exists
            ES_APPLICATION_LOGS is_struct
```

```
            ES_APPLICATION_LOGS {
                # Scenarios 4, 5, 6 and 7
                Enabled exists
                Enabled == true

                CloudWatchLogsLogGroupArn exists
                check_is_string_and_not_empty(CloudWatchLogsLogGroupArn) or
                check_local_references(%INPUT_DOCUMENT, CloudWatchLogsLogGroupArn,
 "AWS::Logs::LogGroup")
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
```

```
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.OPENSEARCH.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    DependsOn: LogGroupPolicy
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
```

```
          Resource: '*'
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          CloudWatchLogsLogGroupArn:
            Fn::GetAtt:
            - LogGroup
            - Arn
          Enabled: true
  LogGroup:
    Type: AWS::Logs::LogGroup
  LogGroupPolicy:
    Type: AWS::Logs::ResourcePolicy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-AllowES
      PolicyDocument:
        Fn::Sub:
        - '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal": {"Service": ["es.amazonaws.com"]},"Action":
["logs:PutLogEvents","logs:CreateLogStream"],"Resource":"${LogGroupArn}","Condition":
{"StringEquals":{ "aws:SourceAccount": "${AWS::AccountId}"}}}]}'
        - LogGroupArn:
            Fn::GetAtt: [ LogGroup, Arn ]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
```

```
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          Enabled: false
```

[CT.OPENSEARCH.PR.5] Require an Elasticsearch domain to send audit logs to Amazon CloudWatch Logs

This control checks whether Elasticsearch domains are configured to send audit logs to an Amazon CloudWatch Logs log group.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.5 example templates

Explanation

Audit logs are highly customizable. They allow you to track user activity on your Elasticsearch clusters, including authentication successes and failures, requests to OpenSearch, index changes, and incoming search queries.

Usage considerations

- This control requires that Elasticsearch domains must have advanced security options configured.
- To enable advanced security options on an Elasticsearch domain through the `AdvancedSecurityOptions` property, you must enable encryption of data at rest (by means of `EncryptionAtRestOptions`), node-to-node encryption (by means of `NodeToNodeEncryptionOptions`), and enforce HTTPS connections (by means of `DomainEndpointOptions`).

Remediation for rule failure

Within `LogPublishingOptions`, provide an `AUDIT_LOGS` configuration, set `Enabled` to `true`, and set `CloudWatchLogsLogGroupArn` to the ARN of a valid Amazon CloudWatch Logs log group.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured to send audit logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
```

```
            },
            "NodeToNodeEncryptionOptions": {
                "Enabled": true
            },
            "EncryptionAtRestOptions": {
                "Enabled": true
            },
            "DomainEndpointOptions": {
                "EnforceHTTPS": true
            },
            "AdvancedSecurityOptions": {
                "Enabled": true,
                "InternalUserDatabaseEnabled": false,
                "MasterUserOptions": {
                    "MasterUserARN": {
                        "Fn::GetAtt": [
                            "IAMRole",
                            "Arn"
                        ]
                    }
                }
            },
            "LogPublishingOptions": {
                "AUDIT_LOGS": {
                    "CloudWatchLogsLogGroupArn": {
                        "Fn::GetAtt": [
                            "LogGroup",
                            "Arn"
                        ]
                    },
                    "Enabled": true
                }
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
```

```
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
    EncryptionAtRestOptions:
      Enabled: true
    DomainEndpointOptions:
      EnforceHTTPS: true
    AdvancedSecurityOptions:
      Enabled: true
      InternalUserDatabaseEnabled: false
      MasterUserOptions:
        MasterUserARN: !GetAtt 'IAMRole.Arn'
    LogPublishingOptions:
      AUDIT_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt 'LogGroup.Arn'
        Enabled: true
```

CT.OPENSEARCH.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
```

```
# Rule Identifier:
#   elasticsearch_audit_logging_enabled_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured to send audit logs
 to an Amazon CloudWatch Logs log group.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been provided
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has not been provided or provided and set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has not been provided or
 provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 5
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has not been provided or
 provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has not been provided or provided and set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has been provided as a non-
empty string
#            or valid local reference
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has been provided as a non-
empty string
#            or valid local reference
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
```

```
#
rule elasticsearch_audit_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %elasticsearch_domains not empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.5]: Require an Elasticsearch domain to send audit logs to
 Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'AUDIT_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

rule elasticsearch_audit_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.5]: Require an Elasticsearch domain to send audit logs to
 Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'AUDIT_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        LogPublishingOptions exists
        LogPublishingOptions is_struct

        LogPublishingOptions {
            # Scenario 3
            AUDIT_LOGS exists
            AUDIT_LOGS is_struct

            AUDIT_LOGS {
                # Scenarios 4, 5, 6 and 7
                Enabled exists
                Enabled == true
```

```
                CloudWatchLogsLogGroupArn exists
                check_is_string_and_not_empty(CloudWatchLogsLogGroupArn) or
                check_local_references(%INPUT_DOCUMENT, CloudWatchLogsLogGroupArn,
 "AWS::Logs::LogGroup")
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
```

```
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.OPENSEARCH.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    DependsOn: LogGroupPolicy
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
```

```
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt:
            - IAMRole
            - Arn
      LogPublishingOptions:
        AUDIT_LOGS:
          CloudWatchLogsLogGroupArn:
            Fn::GetAtt:
            - LogGroup
            - Arn
          Enabled: true
  LogGroup:
    Type: AWS::Logs::LogGroup
  LogGroupPolicy:
    Type: AWS::Logs::ResourcePolicy
    Properties:
      PolicyName: AllowES
      PolicyDocument:
        Fn::Sub:
        - '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal": {"Service": ["es.amazonaws.com"]},"Action":
["logs:PutLogEvents","logs:CreateLogStream"],"Resource":"${LogGroupArn}","Condition":
{"StringEquals":{ "aws:SourceAccount": "${AWS::AccountId}"}}}]}'
        - LogGroupArn:
            Fn::GetAtt: [ LogGroup, Arn ]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
```

```
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt:
            - IAMRole
            - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
```

```
          AWS: '*'
        Action: es:*
        Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
    EncryptionAtRestOptions:
      Enabled: true
    DomainEndpointOptions:
      EnforceHTTPS: true
    AdvancedSecurityOptions:
      Enabled: true
      InternalUserDatabaseEnabled: false
      MasterUserOptions:
        MasterUserARN:
          Fn::GetAtt:
          - IAMRole
          - Arn
    LogPublishingOptions:
      AUDIT_LOGS:
        Enabled: false
```

[CT.OPENSEARCH.PR.6] Require an Elasticsearch domain to have zone awareness and at least three data nodes

This control checks whether ElasticSearch domains are configured with at least three data nodes and zone awareness enabled.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.6 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.6 example templates

Explanation

An Elasticsearch domain requires at least three data nodes for high availability and fault-tolerance. Deploying an Elasticsearch domain with at least three data nodes ensures that cluster operations can continue if a node fails.

Remediation for rule failure

Within `ElasticsearchClusterConfig`, set `ZoneAwarenessEnabled` to `true`, and set `InstanceCount` to an integer value greater than or equal to three.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured with three data nodes and zone awareness enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
```

```
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "ElasticsearchClusterConfig": {
                "InstanceType": "t3.small.elasticsearch",
                "InstanceCount": 3,
                "ZoneAwarenessEnabled": true,
                "ZoneAwarenessConfig": {
                    "AvailabilityZoneCount": 3
                }
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    ElasticsearchClusterConfig:
      InstanceType: t3.small.elasticsearch
      InstanceCount: 3
      ZoneAwarenessEnabled: true
```

```
      ZoneAwarenessConfig:
        AvailabilityZoneCount: 3
```

CT.OPENSEARCH.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticsearch_data_node_fault_tolerance_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured with at least
 three data nodes and zone awareness enabled.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
```

```
#       And: 'ZoneAwarenessEnabled' in 'ElasticsearchClusterConfig' has not been
 provided
#             or provided and set to a value other than bool(true)
#       And: 'InstanceCount' in 'ElasticsearchClusterConfig' has not been provided or
#             provided and set to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ElasticsearchClusterConfig' has been provided
#             and set to bool(true)
#       And: 'InstanceCount' in 'ElasticsearchClusterConfig' has not been provided or
#             provided and set to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ElasticsearchClusterConfig' has not been
 provided
#             or provided and set to a value other than bool(true)
#       And: 'InstanceCount' in 'ElasticsearchClusterConfig' has been provided and set
 to
#             an integer value greater than or equal to three (>= 3)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ElasticsearchClusterConfig' has been provided
#             and set to bool(true)
#       And: 'InstanceCount' in 'ElasticsearchClusterConfig' has been provided and set
 to
#             an integer value greater than or equal to three (>= 3)
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_data_node_fault_tolerance_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %elasticsearch_domains not
 empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.6]: Require an Elasticsearch domain to have zone awareness
 and at least three data nodes
            [FIX]: Within 'ElasticsearchClusterConfig', set 'ZoneAwarenessEnabled' to
 'true', and set 'InstanceCount' to an integer value greater than or equal to three.
        >>
}

rule elasticsearch_data_node_fault_tolerance_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.6]: Require an Elasticsearch domain to have zone awareness
 and at least three data nodes
            [FIX]: Within 'ElasticsearchClusterConfig', set 'ZoneAwarenessEnabled' to
 'true', and set 'InstanceCount' to an integer value greater than or equal to three.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        ElasticsearchClusterConfig exists
        ElasticsearchClusterConfig is_struct

        ElasticsearchClusterConfig {
            # Scenario 3, 4, 5 and 6
            ZoneAwarenessEnabled exists
            ZoneAwarenessEnabled == true
```

```
            InstanceCount exists
            InstanceCount >= 3
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
```

```
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      ElasticsearchClusterConfig:
        InstanceType: t3.small.elasticsearch
        InstanceCount: 3
        ZoneAwarenessEnabled: true
        ZoneAwarenessConfig:
          AvailabilityZoneCount: 3
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      ElasticsearchClusterConfig:
        InstanceType: t3.small.elasticsearch
        ZoneAwarenessEnabled: false
        InstanceCount: 2
```

[CT.OPENSEARCH.PR.7] Require an Elasticsearch domain to have at least three dedicated master nodes

This control checks whether Elasticsearch domains are configured with at least three dedicated master nodes.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.7rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.7rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.7 example templates

Explanation

An Elasticsearch domain requires at least three dedicated master nodes for high availability and fault-tolerance. Dedicated master node resources can be strained during data node blue/green deployments, because additional nodes must be managed. Deploying an Elasticsearch domain with at least three dedicated master nodes ensures that sufficient master node resource capacity exists, and that cluster operations can continue if a node fails.

Remediation for rule failure

Within `ElasticsearchClusterConfig`, set `DedicatedMasterEnabled` to `true`, and set `DedicatedMasterCount` to an integer value greater than or equal to three, or omit the `DedicatedMasterCount` property to adopt the default value of three.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example One

An Elasticsearch domain configured with three dedicated master nodes by means of the `DedicatedMasterCount` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "ElasticsearchClusterConfig": {
                "InstanceType": "t3.small.elasticsearch",
                "DedicatedMasterEnabled": true,
                "DedicatedMasterCount": 3
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    ElasticsearchClusterConfig:
      InstanceType: t3.small.elasticsearch
      DedicatedMasterEnabled: true
      DedicatedMasterCount: 3
```

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example Two

An Elasticsearch domain configured with three dedicated master nodes by means of the AWS CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
```

```
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "ElasticsearchClusterConfig": {
                "InstanceType": "t3.small.elasticsearch",
                "DedicatedMasterEnabled": true
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
```

```
          Action: es:*
          Resource: '*'
    ElasticsearchClusterConfig:
      InstanceType: t3.small.elasticsearch
      DedicatedMasterEnabled: true
```

CT.OPENSEARCH.PR.7rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticsearch_primary_node_fault_tolerance_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured with at least
 three dedicated master nodes.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'DedicatedMasterEnabled' in 'ElasticsearchClusterConfig' has not been
 provided
#             or provided and set to a value other than bool(true)
#       And: 'DedicatedMasterCount' in 'ElasticsearchClusterConfig' has not been
 provided or
#             provided and set to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'DedicatedMasterEnabled' in 'ElasticsearchClusterConfig' has not been
 provided
#             or provided and set to a value other than bool(true)
#       And: 'DedicatedMasterCount' in 'ElasticsearchClusterConfig' has been provided
 and set to
#             an integer value greater than or equal to three (>= 3)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'DedicatedMasterEnabled' in 'ElasticsearchClusterConfig' has been provided
#             and set to bool(true)
#       And: 'DedicatedMasterCount' in 'ElasticsearchClusterConfig' has been provided
 and set
#            to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'DedicatedMasterEnabled' in 'ElasticsearchClusterConfig' has been provided
#             and set to bool(true)
#       And: 'DedicatedMasterCount' in 'ElasticsearchClusterConfig' has not been
 provided
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'ElasticsearchClusterConfig' has been provided
#       And: 'DedicatedMasterEnabled' in 'ElasticsearchClusterConfig' has been provided
```

```
#             and set to bool(true)
#       And: 'DedicatedMasterCount' in 'ElasticsearchClusterConfig' has been provided
 and set to
#             an integer value greater than or equal to three (>= 3)
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_primary_node_fault_tolerance_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %elasticsearch_domains not
 empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.7]: Require an Elasticsearch domain to have at least three
 dedicated master nodes
            [FIX]: Within 'ElasticsearchClusterConfig', set 'DedicatedMasterEnabled'
 to 'true', and set 'DedicatedMasterCount' to an integer value greater than or equal
 to three, or omit the 'DedicatedMasterCount' property to adopt the default value of
 three.
        >>
}

rule elasticsearch_primary_node_fault_tolerance_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.7]: Require an Elasticsearch domain to have at least three
 dedicated master nodes
            [FIX]: Within 'ElasticsearchClusterConfig', set 'DedicatedMasterEnabled'
 to 'true', and set 'DedicatedMasterCount' to an integer value greater than or equal
```

```
 to three, or omit the 'DedicatedMasterCount' property to adopt the default value of
 three.
        >>
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        ElasticsearchClusterConfig exists
        ElasticsearchClusterConfig is_struct

        ElasticsearchClusterConfig {
            # Scenario 3, 4, 5 and 6
            DedicatedMasterEnabled exists
            DedicatedMasterEnabled == true

            DedicatedMasterCount not exists or
            DedicatedMasterCount >= 3
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceType: t3.small.elasticsearch
        DedicatedMasterEnabled: true
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceType: t3.small.elasticsearch
        DedicatedMasterEnabled: true
```

```
        DedicatedMasterCount: 3
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceType: t3.small.elasticsearch
        DedicatedMasterEnabled: false
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

[CT.OPENSEARCH.PR.8] Require an Elasticsearch Service domain to use TLSv1.2

This control checks whether Elasticsearch Service domains are configured to require HTTPS with a minimum TLS version of TLSv1.2.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Elasticsearch::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.8rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.8rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.8 example templates

Explanation

HTTPS (TLS) can help prevent potential attackers from using person-in-the-middle, or similar attacks, to eavesdrop on or manipulate network traffic. Only encrypted connections over HTTPS (TLS) should be allowed. Encrypting data in transit can affect performance. You should test your application with this feature to understand the performance profile and the effects of TLS. TLS 1.2 provides several security enhancements over previous versions of TLS.

Remediation for rule failure

Within `DomainEndpointOptions`, set `EnforceHTTPS` to `true` and set `TLSSecurityPolicy` to `Policy-Min-TLS-1-2-2019-07`.

The examples that follow show how to implement this remediation.

Elasticsearch Domain - Example

An Elasticsearch domain configured to require that all traffic to the domain arrives over HTTPS with a minimum TLS version of TLSv1.2. The example is shown in JSON and in YAML.

JSON example

```
{
    "ElasticsearchDomain": {
        "Type": "AWS::Elasticsearch::Domain",
        "Properties": {
            "ElasticsearchVersion": 7.1,
            "ElasticsearchClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.elasticsearch"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "DomainEndpointOptions": {
                "EnforceHTTPS": true,
                "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
            }
        }
    }
}
```

YAML example

```
ElasticsearchDomain:
  Type: AWS::Elasticsearch::Domain
  Properties:
    ElasticsearchVersion: 7.1
    ElasticsearchClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.elasticsearch
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    DomainEndpointOptions:
      EnforceHTTPS: true
      TLSSecurityPolicy: Policy-Min-TLS-1-2-2019-07
```

CT.OPENSEARCH.PR.8rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   elasticsearch_https_required_check
#
# Description:
#   This control checks whether Elasticsearch domains are configured to require HTTPS
 with a minimum TLS version of TLSv1.2.
#
# Reports on:
#   AWS::Elasticsearch::Domain
#
```

```
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Elasticsearch domain resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'DomainEndpointOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than 'Policy-Min-
TLS-1-2-2019-07'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has been provided and set to
 bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than 'Policy-Min-
TLS-1-2-2019-07'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has been provided and set
#            to 'Policy-Min-TLS-1-2-2019-07'
```

```
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Elasticsearch domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has been provided and set to
 bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has been provided and set
#            to 'Policy-Min-TLS-1-2-2019-07'
#      Then: PASS

#
# Constants
#
let ELASTICSEARCH_DOMAIN_TYPE = "AWS::Elasticsearch::Domain"
let ALLOWED_TLS_POLICIES = [ "Policy-Min-TLS-1-2-2019-07" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let elasticsearch_domains = Resources.*[ Type == %ELASTICSEARCH_DOMAIN_TYPE ]

#
# Primary Rules
#
rule elasticsearch_https_required_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %elasticsearch_domains not empty {
    check(%elasticsearch_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.8]: Require an Elasticsearch Service domain to use TLSv1.2
            [FIX]: Within 'DomainEndpointOptions', set 'EnforceHTTPS' to 'true' and set
 'TLSSecurityPolicy' to 'Policy-Min-TLS-1-2-2019-07'.
        >>
}

rule elasticsearch_https_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %ELASTICSEARCH_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%ELASTICSEARCH_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.8]: Require an Elasticsearch Service domain to use TLSv1.2
            [FIX]: Within 'DomainEndpointOptions', set 'EnforceHTTPS' to 'true' and set
 'TLSSecurityPolicy' to 'Policy-Min-TLS-1-2-2019-07'.
        >>
```

```
}

#
# Parameterized Rules
#
rule check(elasticsearch_domain) {
    %elasticsearch_domain {
        # Scenario 2
        DomainEndpointOptions exists
        DomainEndpointOptions is_struct

        DomainEndpointOptions {
            # Scenarios 3, 4, 5 and 6
            EnforceHTTPS exists
            EnforceHTTPS == true

            TLSSecurityPolicy exists
            TLSSecurityPolicy in %ALLOWED_TLS_POLICIES
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      DomainEndpointOptions:
        EnforceHTTPS: true
        TLSSecurityPolicy: Policy-Min-TLS-1-2-2019-07
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ElasticsearchDomain:
    Type: AWS::Elasticsearch::Domain
    Properties:
      ElasticsearchVersion: 7.1
      ElasticsearchClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.elasticsearch
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
```

```
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      DomainEndpointOptions:
        EnforceHTTPS: true
        TLSSecurityPolicy: Policy-Min-TLS-1-0-2019-07
```

[CT.OPENSEARCH.PR.9] Require an Amazon OpenSearch Service domain to encrypt data at rest

This control checks whether Amazon OpenSearch Service domains have encryption-at-rest enabled.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.9 example templates

Explanation

For an added layer of security for sensitive data, you should configure your OpenSearch Service domain to be encrypted at rest. When you configure encryption of data at rest, AWS KMS stores

and manages your encryption keys. To perform the encryption, AWS KMS uses the Advanced Encryption Standard algorithm with 256-bit keys (AES-256).

Remediation for rule failure

Within `EncryptionAtRestOptions`, set `Enabled` to `true`.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured with encryption-at-rest enabled The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
```

```
            },
            "EncryptionAtRestOptions": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    EncryptionAtRestOptions:
      Enabled: true
```

CT.OPENSEARCH.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   opensearch_encrypted_at_rest_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains have encryption-at-
rest enabled.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'EncryptionAtRestOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'EncryptionAtRestOptions' has been provided
#       And: In 'EncryptionAtRestOptions', 'Enabled' has not been provided or provided
#            and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'EncryptionAtRestOptions' has been provided
#       And: In 'EncryptionAtRestOptions','Enabled' has been provided and set to
 bool(true)
#      Then: PASS

#
```

```
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_encrypted_at_rest_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %opensearch_service_domains not empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.9]: Require an Amazon OpenSearch Service domain to encrypt
 data at rest
            [FIX]: Within 'EncryptionAtRestOptions', set 'Enabled' to 'true'.
        >>
}

rule opensearch_encrypted_at_rest_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.9]: Require an Amazon OpenSearch Service domain to encrypt
 data at rest
            [FIX]: Within 'EncryptionAtRestOptions', set 'Enabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        EncryptionAtRestOptions exists
        EncryptionAtRestOptions is_struct

        EncryptionAtRestOptions {
            # Scenarios 3 and 4
```

```
            Enabled exists
            Enabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
```

```
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      EncryptionAtRestOptions:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      EncryptionAtRestOptions:
        Enabled: false
```

[CT.OPENSEARCH.PR.10] Require an Amazon OpenSearch Service domain to be created in a user-specified Amazon VPC

This control checks whether Amazon OpenSearch Service domains are configured with VPC option settings that specify a target Amazon VPC.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.10 example templates

Explanation

Ensure that OpenSearch domains are not attached to public subnets. See `Resource-based policies` in the Amazon OpenSearch Service Developer Guide. Also ensure that your VPC is configured according to the recommended best practices. See `Security best practices for your VPC` in the Amazon VPC User Guide.

OpenSearch domains deployed within a VPC can communicate with VPC resources over the private AWS network, without the need to traverse the public internet. This configuration increases the security posture by limiting access to the data in transit. VPCs provide a number of network controls to secure access to OpenSearch domains, including network ACL and security groups. AWS Control Tower recommends that you migrate public OpenSearch domains to VPCs to take advantage of these controls.

Remediation for rule failure

Within `VPCOptions`, set `SubnetIds` to a list with one or more AAmazon EC2 subnet IDs.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured to deploy within an Amazon VPC by means of VPC option settings. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "VPCOptions": {
                "SubnetIds": [
                    {
                        "Ref": "Subnet"
                    }
                ]
            }
        }
```

```
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    VPCOptions:
      SubnetIds:
        - !Ref 'Subnet'
```

CT.OPENSEARCH.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_in_vpc_only_check
#
```

```
# Description:
#   This control checks whether Amazon OpenSearch Service domains are configured with
 VPC option settings that specify a target Amazon VPC.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'VPCOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'VPCOptions' has been provided
#       And: 'SubnetIds' in 'VPCOptions' has not been provided or has been provided
#            as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'VPCOptions' has been provided
#       And: 'SubnetIds' in 'VPCOptions' has been provided as a list with one or more
 values
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_in_vpc_only_check when is_cfn_template(%INPUT_DOCUMENT)
                                       %opensearch_service_domains not empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.10]: Require an Amazon OpenSearch Service domain to be
 created in a user-specified Amazon VPC
            [FIX]: Within 'VPCOptions', set 'SubnetIds' to a list with one or more
 Amazon EC2 subnet IDs.
        >>
}

rule opensearch_in_vpc_only_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.10]: Require an Amazon OpenSearch Service domain to be
 created in a user-specified Amazon VPC
            [FIX]: Within 'VPCOptions', set 'SubnetIds' to a list with one or more
 Amazon EC2 subnet IDs.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        VPCOptions exists
        VPCOptions is_struct

        VPCOptions {
            # Scenarios 3 and 4
            SubnetIds exists
            SubnetIds is_list
```

```
            SubnetIds not empty
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 10.0.0.0/16
  Subnet:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/16
      AvailabilityZone:
        Fn::Select:
        - 0
```

```
        - Fn::GetAZs: ''
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      VPCOptions:
        SubnetIds:
        - Ref: Subnet
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
```

```
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

[CT.OPENSEARCH.PR.11] Require an Amazon OpenSearch Service domain to encrypt data sent between nodes

This control checks whether Amazon OpenSearch Service domains have node-to-node encryption enabled.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.11 example templates

Explanation

HTTPS (TLS) can help prevent potential attackers from eavesdropping on or manipulating network traffic using person-in-the-middle, or similar, attacks. Only encrypted connections over HTTPS (TLS) should be allowed. Enabling node-to-node encryption for OpenSearch domains ensures that intra-cluster communications are encrypted in transit.

Usage considerations

- A performance penalty may be associated with this configuration. You should be aware of the performance trade-offs and test them before enabling this option.

Remediation for rule failure

Within `NodeToNodeEncryptionOptions`, set `Enabled` to `true`.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured with node-to-node encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
```

```
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "NodeToNodeEncryptionOptions": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
```

CT.OPENSEARCH.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_node_to_node_encryption_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains have node-to-node
 encryption enabled.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'NodeToNodeEncryptionOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'NodeToNodeEncryptionOptions' has been provided
#       And: In 'NodeToNodeEncryptionOptions', 'Enabled' has not been provided or has
 been provided and set to a
#            value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'NodeToNodeEncryptionOptions' has been provided
#       And: In 'NodeToNodeEncryptionOptions', 'Enabled' has been provided and set to a
#            value of bool(true)
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_node_to_node_encryption_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %opensearch_service_domains not
 empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.11]: Require an Amazon OpenSearch Service domain to encrypt
 data sent between nodes
            [FIX]: Within 'NodeToNodeEncryptionOptions', set 'Enabled' to 'true'.
        >>
}

rule opensearch_node_to_node_encryption_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.11]: Require an Amazon OpenSearch Service domain to encrypt
 data sent between nodes
            [FIX]: Within 'NodeToNodeEncryptionOptions', set 'Enabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
```

```
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        NodeToNodeEncryptionOptions exists
        NodeToNodeEncryptionOptions is_struct

        NodeToNodeEncryptionOptions {
            # Scenarios 3 and 4
            Enabled exists
            Enabled == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
```

```
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
```

```
          AWS: '*'
        Action: es:*
        Resource: '*'
    NodeToNodeEncryptionOptions: {}
```

[CT.OPENSEARCH.PR.12] Require an Amazon OpenSearch Service domain to send error logs to Amazon CloudWatch Logs

This control checks whether Amazon OpenSearch Service domains are configured to send error logs to an Amazon CloudWatch Logs log group.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.12 example templates

Explanation

Enable error logs (ES_APPLICATION_LOGS) for OpenSearch Service domains and send those logs to Amazon CloudWatch Logs for retention and response. Domain error logs can assist with security and access audits, and can help to diagnose availability issues.

Remediation for rule failure

Within `LogPublishingOptions`, provide an `ES_APPLICATION_LOGS` configuration, set `Enabled` to `true`, and set `CloudWatchLogsLogGroupArn` to the ARN of a valid Amazon CloudWatch Logs log group.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured to send error logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "LogPublishingOptions": {
                "ES_APPLICATION_LOGS": {
                    "CloudWatchLogsLogGroupArn": {
                        "Fn::GetAtt": [
                            "LogGroup",
                            "Arn"
```

```
                        ]
                    },
                    "Enabled": true
                }
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    LogPublishingOptions:
      ES_APPLICATION_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt 'LogGroup.Arn'
        Enabled: true
```

CT.OPENSEARCH.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_application_logging_enabled_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains are configured to
 send error logs to an Amazon CloudWatch Logs log group.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been provided
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has not been provided or provided and
 set to
```

```
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has not been provided
 or provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service Domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has not been provided
 or provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service Domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has not been provided or provided and
 set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has been provided as
 a non-empty string
#            or valid local reference
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service Domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'ES_APPLICATION_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'ES_APPLICATION_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'ES_APPLICATION_LOGS' has been provided as
 a non-empty string
#            or valid local reference
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_application_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %opensearch_service_domains not
 empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.12]: Require an Amazon OpenSearch Service domain to send
 error logs to Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'ES_APPLICATION_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

rule opensearch_application_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.12]: Require an Amazon OpenSearch Service domain to send
 error logs to Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'ES_APPLICATION_LOGS'
 configuration, set 'Enabled' to 'true', and set 'CloudWatchLogsLogGroupArn' to the ARN
 of a valid Amazon CloudWatch Logs log group.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        LogPublishingOptions exists
        LogPublishingOptions is_struct

        LogPublishingOptions {
            # Scenario 3
```

```
            ES_APPLICATION_LOGS exists
            ES_APPLICATION_LOGS is_struct

            ES_APPLICATION_LOGS {
                # Scenarios 4, 5, 6 and 7
                Enabled exists
                Enabled == true

                CloudWatchLogsLogGroupArn exists
                check_is_string_and_not_empty(CloudWatchLogsLogGroupArn) or
                check_local_references(%INPUT_DOCUMENT, CloudWatchLogsLogGroupArn,
 "AWS::Logs::LogGroup")
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
```

```
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.OPENSEARCH.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    DependsOn: LogGroupPolicy
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
```

```
            AWS: '*'
          Action: es:*
          Resource: '*'
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          CloudWatchLogsLogGroupArn:
            Fn::GetAtt:
            - LogGroup
            - Arn
          Enabled: true
  LogGroup:
    Type: AWS::Logs::LogGroup
  LogGroupPolicy:
    Type: AWS::Logs::ResourcePolicy
    Properties:
      PolicyName:
        Fn::Sub: ${AWS::StackName}-AllowOS
      PolicyDocument:
        Fn::Sub:
        - '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal": {"Service": ["es.amazonaws.com"]},"Action":
["logs:PutLogEvents","logs:CreateLogStream"],"Resource":"${LogGroupArn}","Condition":
{"StringEquals":{"aws:SourceAccount": "${AWS::AccountId}"}}}]}'
        - LogGroupArn:
            Fn::GetAtt: [LogGroup, Arn]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
```

```
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          Enabled: false
```

[CT.OPENSEARCH.PR.13] Require an Amazon OpenSearch Service domain to send audit logs to Amazon CloudWatch Logs

This control checks whether Amazon OpenSearch Service domains are configured to send audit logs to an Amazon CloudWatch Logs log group.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.13 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.13 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.13 example templates

Explanation

Audit logs are highly customizable. They allow you to track user activity on your OpenSearch clusters, including authentication successes and failures, requests to OpenSearch, index changes, and incoming search queries.

Usage considerations

- Audit log publishing requires advanced security options to be enabled on Amazon OpenSearch Service domains.
- To enable advanced security options on an Amazon OpenSearch Service domain, you must enable encryption of data at rest by means of the `EncryptionAtRestOptions` property, node-to-node encryption by means of the `NodeToNodeEncryptionOptions` property, and enforce HTTPS connections by means of the `EnforceHTTPS` property within `DomainEndpointOptions`.

Remediation for rule failure

Within `LogPublishingOptions`, provide an `AUDIT_LOGS` configuration, set `Enabled` to `true` and `CloudWatchLogsLogGroupArn` to the ARN of a valid Amazon CloudWatch Logs log group.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured to send audit logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "DependsOn": "LogGroupPolicy",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
```

```
            },
            "NodeToNodeEncryptionOptions": {
                "Enabled": true
            },
            "EncryptionAtRestOptions": {
                "Enabled": true
            },
            "DomainEndpointOptions": {
                "EnforceHTTPS": true
            },
            "AdvancedSecurityOptions": {
                "Enabled": true,
                "InternalUserDatabaseEnabled": false,
                "MasterUserOptions": {
                    "MasterUserARN": {
                        "Fn::GetAtt": [
                            "IAMRole",
                            "Arn"
                        ]
                    }
                }
            },
            "LogPublishingOptions": {
                "AUDIT_LOGS": {
                    "CloudWatchLogsLogGroupArn": {
                        "Fn::GetAtt": [
                            "LogGroup",
                            "Arn"
                        ]
                    },
                    "Enabled": true
                }
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
```

```
  DependsOn: LogGroupPolicy
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
    EncryptionAtRestOptions:
      Enabled: true
    DomainEndpointOptions:
      EnforceHTTPS: true
    AdvancedSecurityOptions:
      Enabled: true
      InternalUserDatabaseEnabled: false
      MasterUserOptions:
        MasterUserARN: !GetAtt 'IAMRole.Arn'
    LogPublishingOptions:
      AUDIT_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt 'LogGroup.Arn'
        Enabled: true
```

CT.OPENSEARCH.PR.13 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   opensearch_audit_logging_enabled_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains are configured to
 send audit logs to an Amazon CloudWatch Logs log group.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been provided
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has not been provided or provided and set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has not been provided or
 provided
#            as an empty string or invalid local reference
```

```
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has not been provided or
 provided
#            as an empty string or invalid local reference
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has not been provided or provided and set to
#            a value other than bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has been provided as a non-
empty string
#            or valid local reference
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'LogPublishingOptions' has been specified
#       And: 'AUDIT_LOGS' in 'LogPublishingOptions' has been provided
#       And: 'Enabled' in 'AUDIT_LOGS' has been provided and set to bool(true)
#       And: 'CloudWatchLogsLogGroupArn' in 'AUDIT_LOGS' has been provided as a non-
empty string
#            or valid local reference
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]
```

```
#
# Primary Rules
#
rule opensearch_audit_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %opensearch_service_domains not empty
 {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.13]: Require an Amazon OpenSearch Service domain to send
 audit logs to Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'AUDIT_LOGS'
 configuration, set 'Enabled' to 'true' and 'CloudWatchLogsLogGroupArn' to the ARN of a
 valid Amazon CloudWatch Logs log group.
        >>
}

rule opensearch_audit_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.13]: Require an Amazon OpenSearch Service domain to send
 audit logs to Amazon CloudWatch Logs
            [FIX]: Within 'LogPublishingOptions', provide an 'AUDIT_LOGS'
 configuration, set 'Enabled' to 'true' and 'CloudWatchLogsLogGroupArn' to the ARN of a
 valid Amazon CloudWatch Logs log group.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        LogPublishingOptions exists
        LogPublishingOptions is_struct

        LogPublishingOptions {
            # Scenario 3
            AUDIT_LOGS exists
            AUDIT_LOGS is_struct

            AUDIT_LOGS {
                # Scenarios 4, 5, 6 and 7
```

```
                Enabled exists
                Enabled == true

                CloudWatchLogsLogGroupArn exists
                check_is_string_and_not_empty(CloudWatchLogsLogGroupArn) or
                check_local_references(%INPUT_DOCUMENT, CloudWatchLogsLogGroupArn,
 "AWS::Logs::LogGroup")
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}
```

```
rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.OPENSEARCH.PR.13 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    DependsOn: LogGroupPolicy
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
```

```
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt:
            - IAMRole
            - Arn
      LogPublishingOptions:
        AUDIT_LOGS:
          CloudWatchLogsLogGroupArn:
            Fn::GetAtt:
            - LogGroup
            - Arn
          Enabled: true
  LogGroup:
    Type: AWS::Logs::LogGroup
  LogGroupPolicy:
    Type: AWS::Logs::ResourcePolicy
    Properties:
      PolicyName: AllowES
      PolicyDocument:
        Fn::Sub:
        - '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal": {"Service": ["es.amazonaws.com"]},"Action":
["logs:PutLogEvents","logs:CreateLogStream"],"Resource":"${LogGroupArn}","Condition":
{"StringEquals":{ "aws:SourceAccount": "${AWS::AccountId}"}}}]}'
        - LogGroupArn:
            Fn::GetAtt: [ LogGroup, Arn ]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
```

```
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt:
            - IAMRole
            - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
```

```
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt:
            - IAMRole
            - Arn
      LogPublishingOptions:
        AUDIT_LOGS:
          Enabled: false
```

[CT.OPENSEARCH.PR.14] Require an Amazon OpenSearch Service domain to have zone awareness and at least three data nodes

This control checks whether Amazon OpenSearch Service domains are configured with at least three data nodes and zone awareness enabled.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.14 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.14 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.14 example templates

Explanation

An OpenSearch domain requires at least three data nodes for high availability and fault-tolerance. Deploying an OpenSearch domain with at least three data nodes ensures that the cluster can remain operative if a node fails.

Remediation for rule failure

Within `ClusterConfig`, set `ZoneAwarenessEnabled` to `true` and `InstanceCount` to an integer value greater than or equal to three.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured with three data nodes and zone awareness enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
```

```
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "ClusterConfig": {
                "InstanceType": "t3.small.search",
                "InstanceCount": 3,
                "ZoneAwarenessEnabled": true,
                "ZoneAwarenessConfig": {
                    "AvailabilityZoneCount": 3
                }
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    ClusterConfig:
```

```
      InstanceType: t3.small.search
      InstanceCount: 3
      ZoneAwarenessEnabled: true
      ZoneAwarenessConfig:
        AvailabilityZoneCount: 3
```

CT.OPENSEARCH.PR.14 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_data_node_fault_tolerance_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains are configured with
 at least three data nodes and zone awareness enabled.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'ClusterConfig' has not been provided
#      Then: FAIL
#   Scenario: 3
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'ClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ClusterConfig' has not been provided
#             or provided and set to a value other than bool(true)
#       And: 'InstanceCount' in 'ClusterConfig' has not been provided or
#             provided and set to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'ClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ClusterConfig' has been provided
#             and set to bool(true)
#       And: 'InstanceCount' in 'ClusterConfig' has not been provided or
#             provided and set to an integer value less than three (< 3)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'ClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ClusterConfig' has not been provided
#             or provided and set to a value other than bool(true)
#       And: 'InstanceCount' in 'ClusterConfig' has been provided and set to
#             an integer value greater than or equal to three (>= 3)
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'ClusterConfig' has been provided
#       And: 'ZoneAwarenessEnabled' in 'ClusterConfig' has been provided
#             and set to bool(true)
#       And: 'InstanceCount' in 'ClusterConfig' has been provided and set to
#             an integer value greater than or equal to three (>= 3)
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
```

```
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_data_node_fault_tolerance_check when is_cfn_template(%INPUT_DOCUMENT)
                                                     %opensearch_service_domains not
 empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.14]: Require an Amazon OpenSearch Service domain to have zone
 awareness and at least three data nodes
            [FIX]: Within 'ClusterConfig', set 'ZoneAwarenessEnabled' to 'true' and
 'InstanceCount' to an integer value greater than or equal to three.
        >>
}

rule opensearch_data_node_fault_tolerance_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.14]: Require an Amazon OpenSearch Service domain to have zone
 awareness and at least three data nodes
            [FIX]: Within 'ClusterConfig', set 'ZoneAwarenessEnabled' to 'true' and
 'InstanceCount' to an integer value greater than or equal to three.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        ClusterConfig exists
        ClusterConfig is_struct

        ClusterConfig {
            # Scenario 3, 4, 5 and 6
            ZoneAwarenessEnabled exists
            ZoneAwarenessEnabled == true

            InstanceCount exists
```

```
            InstanceCount >= 3
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.14 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
```

```
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      ClusterConfig:
        InstanceType: t3.small.search
        InstanceCount: 3
        ZoneAwarenessEnabled: true
        ZoneAwarenessConfig:
          AvailabilityZoneCount: 3
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceType: t3.small.search
        ZoneAwarenessEnabled: false
        InstanceCount: 2
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
```

[CT.OPENSEARCH.PR.15] Require an Amazon OpenSearch Service domain to use fine-grained access control

This control checks whether Amazon OpenSearch Service domains have fine-grained access control enabled.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.15 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.15 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.15 example templates

Explanation

Fine-grained access control offers additional ways of controlling access to your data on Amazon OpenSearch Service.

Usage considerations

- Fine-grained access control requires that advanced security options must be enabled on Amazon OpenSearch Service domains.
- To enable advanced security options on an Amazon OpenSearch Service domain, you must enable encryption of data at rest by means of the `EncryptionAtRestOptions` property, node-to-node encryption by means of the `NodeToNodeEncryptionOptions` property, and enforce HTTPS connections by means of the `EnforceHTTPS` property within `DomainEndpointOptions`.

Remediation for rule failure

Within `AdvancedSecurityOptions`, set `Enabled` to `true`, set `InternalUserDatabaseEnabled` to `true` or `false`, and set `MasterUserOptions` with an options configuration for your master user.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured with fine-grained access control. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
```

```
            },
            "NodeToNodeEncryptionOptions": {
                "Enabled": true
            },
            "EncryptionAtRestOptions": {
                "Enabled": true
            },
            "DomainEndpointOptions": {
                "EnforceHTTPS": true
            },
            "AdvancedSecurityOptions": {
                "Enabled": true,
                "InternalUserDatabaseEnabled": false,
                "MasterUserOptions": {
                    "MasterUserARN": {
                        "Fn::GetAtt": [
                            "IAMRole",
                            "Arn"
                        ]
                    }
                }
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
```

```
      Version: '2012-10-17       '
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    NodeToNodeEncryptionOptions:
      Enabled: true
    EncryptionAtRestOptions:
      Enabled: true
    DomainEndpointOptions:
      EnforceHTTPS: true
    AdvancedSecurityOptions:
      Enabled: true
      InternalUserDatabaseEnabled: false
      MasterUserOptions:
        MasterUserARN: !GetAtt 'IAMRole.Arn'
```

CT.OPENSEARCH.PR.15 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_fine_grained_access_control_enabled_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains have fine-grained
 access control enabled.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has been provided
#       And: 'Enabled' in 'AdvancedSecurityOptions' has not been provided or
#            has been provided and set to a value other than bool(true)
#       And: 'InternalUserDatabaseEnabled' in 'AdvancedSecurityOptions' has not been
 provided or provided and set to a
#             non boolean value
#       And: 'MasterUserOptions' in 'AdvancedSecurityOptions' has not been provided or
 provided and set to a value
#            other than a struct
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has been provided
#       And: 'Enabled' in 'AdvancedSecurityOptions' has been provided and set to
 bool(true)
#       And: 'InternalUserDatabaseEnabled' in 'AdvancedSecurityOptions' has not been
 provided or provided and set to a
#             non boolean value
#       And: 'MasterUserOptions' in 'AdvancedSecurityOptions' has not been provided or
 provided and set to a value
#            other than a struct
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has been provided
#       And: 'Enabled' in 'AdvancedSecurityOptions' has been provided and set to
 bool(true)
```

```
#       And: 'InternalUserDatabaseEnabled' in 'AdvancedSecurityOptions' has been
 provided and set to a
#            boolean value
#       And: 'MasterUserOptions' in 'AdvancedSecurityOptions' has not been provided or
 provided and set to a value
#            other than a struct
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has been provided
#       And: 'Enabled' in 'AdvancedSecurityOptions' has been provided and set to
 bool(true)
#       And: 'InternalUserDatabaseEnabled' in 'AdvancedSecurityOptions' has not been
 provided or provided and set to a
#             non boolean value
#       And: 'MasterUserOptions' in 'AdvancedSecurityOptions' has been provided and set
 to a struct
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'AdvancedSecurityOptions' has been provided
#       And: 'Enabled' in 'AdvancedSecurityOptions' has been provided and set to
 bool(true)
#       And: 'InternalUserDatabaseEnabled' in 'AdvancedSecurityOptions' has been
 provided and set to a
#            boolean value
#       And: 'MasterUserOptions' in 'AdvancedSecurityOptions' has been provided and set
 to a struct
#      Then: PASS

#
# Constants
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let INPUT_DOCUMENT = this

#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
```

```
# Primary Rules
#
rule opensearch_fine_grained_access_control_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %opensearch_service_domains not empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.15]: Require an Amazon OpenSearch Service domain to use fine-
grained access control
            [FIX]: Within 'AdvancedSecurityOptions', set 'Enabled' to 'true', set
 'InternalUserDatabaseEnabled' to 'true' or 'false', and set 'MasterUserOptions' with
 an options configuration for your master user.
        >>
}

rule opensearch_fine_grained_access_control_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.15]: Require an Amazon OpenSearch Service domain to use fine-
grained access control
            [FIX]: Within 'AdvancedSecurityOptions', set 'Enabled' to 'true', set
 'InternalUserDatabaseEnabled' to 'true' or 'false', and set 'MasterUserOptions' with
 an options configuration for your master user.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        AdvancedSecurityOptions exists
        AdvancedSecurityOptions is_struct

        AdvancedSecurityOptions {
            # Scenarios 3, 4, 5, 6 and 7
            Enabled exists
            Enabled == true

            InternalUserDatabaseEnabled exists
            InternalUserDatabaseEnabled in [ true, false ]
```

```
            MasterUserOptions exists
            MasterUserOptions is_struct
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.15 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action:
```

```
          - sts:AssumeRole
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: true
        InternalUserDatabaseEnabled: false
        MasterUserOptions:
          MasterUserARN:
            Fn::GetAtt: [IAMRole, Arn]
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
```

```
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      NodeToNodeEncryptionOptions:
        Enabled: true
      EncryptionAtRestOptions:
        Enabled: true
      DomainEndpointOptions:
        EnforceHTTPS: true
      AdvancedSecurityOptions:
        Enabled: false
```

[CT.OPENSEARCH.PR.16] Require an Amazon OpenSearch Service domain to use TLSv1.2

This control checks whether Amazon OpenSearch Service domains are configured to require HTTPS with a minimum TLS version of TLSv1.2.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::OpenSearchService::Domain`
- **CloudFormation guard rule:** CT.OPENSEARCH.PR.16 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.OPENSEARCH.PR.16 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.OPENSEARCH.PR.16 example templates

Explanation

HTTPS (TLS) can help prevent potential attackers from using person-in-the-middle, or similar attacks, to eavesdrop on or manipulate network traffic. Only encrypted connections over HTTPS (TLS) should be allowed. Encrypting data in transit can affect performance. You should test your application with this feature to understand the performance profile and the effects of TLS. TLS 1.2 provides several security enhancements over previous versions of TLS.

Remediation for rule failure

Within `DomainEndpointOptions`, set `EnforceHTTPS` to `true` and set `TLSSecurityPolicy` to `Policy-Min-TLS-1-2-2019-07`.

The examples that follow show how to implement this remediation.

Amazon OpenSearch Service Domain - Example

An Amazon OpenSearch Service domain configured to require all traffic to the domain arrive over HTTPS with a minimum TLS version of TLSv1.2. The example is shown in JSON and in YAML.

JSON example

```
{
    "OpenSearchServiceDomain": {
        "Type": "AWS::OpenSearchService::Domain",
        "Properties": {
            "EngineVersion": "OpenSearch_1.3",
            "ClusterConfig": {
                "InstanceCount": "1",
                "InstanceType": "t3.small.search"
            },
            "EBSOptions": {
                "EBSEnabled": true,
```

```
                "Iops": "3000",
                "VolumeSize": "10",
                "VolumeType": "gp3"
            },
            "AccessPolicies": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Principal": {
                            "AWS": "*"
                        },
                        "Action": "es:*",
                        "Resource": "*"
                    }
                ]
            },
            "DomainEndpointOptions": {
                "EnforceHTTPS": true,
                "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
            }
        }
    }
}
```

YAML example

```
OpenSearchServiceDomain:
  Type: AWS::OpenSearchService::Domain
  Properties:
    EngineVersion: OpenSearch_1.3
    ClusterConfig:
      InstanceCount: '1'
      InstanceType: t3.small.search
    EBSOptions:
      EBSEnabled: true
      Iops: '3000'
      VolumeSize: '10'
      VolumeType: gp3
    AccessPolicies:
      Version: '2012-10-17       '
```

```
      Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
    DomainEndpointOptions:
      EnforceHTTPS: true
      TLSSecurityPolicy: Policy-Min-TLS-1-2-2019-07
```

CT.OPENSEARCH.PR.16 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   opensearch_https_required_check
#
# Description:
#   This control checks whether Amazon OpenSearch Service domains are configured to
 require HTTPS with a minimum TLS version of TLSv1.2.
#
# Reports on:
#   AWS::OpenSearchService::Domain
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any OpenSearch Service domain
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'DomainEndpointOptions' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set ot a value other than 'Policy-Min-
TLS-1-2-2019-07'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has been provided and set to
 bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set ot a value other than 'Policy-Min-
TLS-1-2-2019-07'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has not been provided or
#            has been provided and set to a value other than bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has been provided and set
#            to 'Policy-Min-TLS-1-2-2019-07'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an OpenSearch Service domain resource
#       And: 'DomainEndpointOptions' has been provided
#       And: 'EnforceHTTPS' in 'DomainEndpointOptions' has been provided and set to
 bool(true)
#       And: 'TLSSecurityPolicy' in 'DomainEndpointOptions' has been provided and set
#            to 'Policy-Min-TLS-1-2-2019-07'
#      Then: PASS

#
# Constants
```

```
#
let OPENSEARCH_SERVICE_DOMAIN_TYPE = "AWS::OpenSearchService::Domain"
let ALLOWED_TLS_POLICIES = [ "Policy-Min-TLS-1-2-2019-07" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let opensearch_service_domains = Resources.*[ Type == %OPENSEARCH_SERVICE_DOMAIN_TYPE ]

#
# Primary Rules
#
rule opensearch_https_required_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %opensearch_service_domains not empty {
    check(%opensearch_service_domains.Properties)
        <<
        [CT.OPENSEARCH.PR.16]: Require an Amazon OpenSearch Service domain to use
 TLSv1.2
            [FIX]: Within 'DomainEndpointOptions', set 'EnforceHTTPS' to 'true' and set
 'TLSSecurityPolicy' to 'Policy-Min-TLS-1-2-2019-07'.
        >>
}

rule opensearch_https_required_check when is_cfn_hook(%INPUT_DOCUMENT,
 %OPENSEARCH_SERVICE_DOMAIN_TYPE) {
    check(%INPUT_DOCUMENT.%OPENSEARCH_SERVICE_DOMAIN_TYPE.resourceProperties)
        <<
        [CT.OPENSEARCH.PR.16]: Require an Amazon OpenSearch Service domain to use
 TLSv1.2
            [FIX]: Within 'DomainEndpointOptions', set 'EnforceHTTPS' to 'true' and set
 'TLSSecurityPolicy' to 'Policy-Min-TLS-1-2-2019-07'.
        >>
}

#
# Parameterized Rules
#
rule check(opensearch_service_domain) {
    %opensearch_service_domain {
        # Scenario 2
        DomainEndpointOptions exists
        DomainEndpointOptions is_struct
```

```
        DomainEndpointOptions {
            # Scenarios 3, 4, 5 and 6
            EnforceHTTPS exists
            EnforceHTTPS == true

            TLSSecurityPolicy exists
            TLSSecurityPolicy in %ALLOWED_TLS_POLICIES
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.OPENSEARCH.PR.16 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
```

```
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
          Resource: '*'
      DomainEndpointOptions:
        EnforceHTTPS: true
        TLSSecurityPolicy: Policy-Min-TLS-1-2-2019-07
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  OpenSearchServiceDomain:
    Type: AWS::OpenSearchService::Domain
    Properties:
      EngineVersion: OpenSearch_1.3
      ClusterConfig:
        InstanceCount: '1'
        InstanceType: t3.small.search
      EBSOptions:
        EBSEnabled: true
        Iops: '3000'
        VolumeSize: '10'
        VolumeType: gp3
      AccessPolicies:
        Version: '2012-10-17       '
        Statement:
        - Effect: Deny
          Principal:
            AWS: '*'
          Action: es:*
```

```
        Resource: '*'
    DomainEndpointOptions:
      EnforceHTTPS: true
      TLSSecurityPolicy: Policy-Min-TLS-1-0-2019-07
```

Amazon Relational Database Service (Amazon RDS) controls

Topics

- [CT.RDS.PR.1] Require that an Amazon RDS database instance is configured with multiple Availability Zones
- [CT.RDS.PR.2] Require an Amazon RDS database instance to have enhanced monitoring configured
- [CT.RDS.PR.3] Require an Amazon RDS cluster to have deletion protection configured
- [CT.RDS.PR.4] Require an Amazon RDS database cluster to have AWS IAM database authentication configured
- [CT.RDS.PR.5] Require an Amazon RDS database instance to have minor version upgrades configured
- [CT.RDS.PR.6] Require an Amazon RDS database cluster to have backtracking configured
- [CT.RDS.PR.7] Require Amazon RDS database instances to have IAM authentication configured
- [CT.RDS.PR.8] Require an Amazon RDS database instance to have automatic backups configured
- [CT.RDS.PR.9] Require an Amazon RDS database cluster to copy tags to snapshots
- [CT.RDS.PR.10] Require an Amazon RDS database instance to copy tags to snapshots
- [CT.RDS.PR.11] Require an Amazon RDS database instance to have a VPC configuration
- [CT.RDS.PR.12] Require an Amazon RDS event subscription to have critical cluster events configured
- [CT.RDS.PR.13] Require any Amazon RDS instance to have deletion protection configured
- [CT.RDS.PR.14] Require an Amazon RDS database instance to export logs to Amazon CloudWatch Logs by means of the EnableCloudwatchLogsExports property
- [CT.RDS.PR.15] Require that an Amazon RDS instance does not create DB security groups
- [CT.RDS.PR.16] Require an Amazon RDS database cluster to have encryption at rest configured
- [CT.RDS.PR.17] Require an Amazon RDS event notification subscription to have critical database instance events configured

- [CT.RDS.PR.18] Require an Amazon RDS event notification subscription to have critical database parameter group events configured
- [CT.RDS.PR.19] Require an Amazon RDS event notifications subscription to have critical database security group events configured
- [CT.RDS.PR.20] Require an Amazon RDS database instance not to use a database engine default port
- [CT.RDS.PR.21] Require an Amazon RDS DB cluster to have a unique administrator username
- [CT.RDS.PR.22] Require an Amazon RDS database instance to have a unique administrator username
- [CT.RDS.PR.23] Require an Amazon RDS database instance to not be publicly accessible
- [CT.RDS.PR.24] Require an Amazon RDS database instance to have encryption at rest configured
- [CT.RDS.PR.25] Require an Amazon RDS database cluster to export logs to Amazon CloudWatch Logs by means of the EnableCloudwatchLogsExports property
- [CT.RDS.PR.26] Require an Amazon Relational Database Service DB Proxy to require Transport Layer Security (TLS) connections
- [CT.RDS.PR.27] Require an Amazon Relational Database Service DB cluster parameter group to require Transport Layer Security (TLS) connections for supported engine types
- [CT.RDS.PR.28] Require an Amazon Relational Database Service DB parameter group to require Transport Layer Security (TLS) connections for supported engine types
- [CT.RDS.PR.29] Require an Amazon RDS cluster not be configured to be publicly accessible by means of the 'PubliclyAccessible' property
- [CT.RDS.PR.30] Require that an Amazon RDS database instance has encryption at rest configured to use a KMS key that you specify for supported engine types

[CT.RDS.PR.1] Require that an Amazon RDS database instance is configured with multiple Availability Zones

This control checks whether high availability is configured for your Amazon Relational Database Service (RDS) database instances.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`

- **CloudFormation guard rule:** CT.RDS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.1 example templates

Explanation

Amazon RDS database (DB) instances should be configured for multiple Availability Zones (AZs). This configuration increases the availability of the stored data. Deployment into multiple Availability Zones allows for automated failover, in case an Availability Zone has an outage, and during regular RDS maintenance.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set `MultiAZ` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS database instance configured with multiple Availability Zones. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "MultiAZ": true
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    MultiAZ: true
  DeletionPolicy: Delete
```

CT.RDS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_multi_az_support_check
#
# Description:
#   This control checks whether high availability is configured for your Amazon
 Relational Database Service (RDS) database instances.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'MultiAZ' has not been specified
```

```
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'MultiAZ' has been specified
#       And: 'MultiAZ' has been set to bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'MultiAZ' has been specified
#       And: 'MultiAZ' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_multi_az_support_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
```

```
        <<
        [CT.RDS.PR.1]: Require that an Amazon RDS database instance is configured with
 multiple Availability Zones
        [FIX]: Set 'MultiAZ' to 'true'.
        >>
}

rule rds_instance_multi_az_support_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.1]: Require that an Amazon RDS database instance is configured with
 multiple Availability Zones
        [FIX]: Set 'MultiAZ' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
       # Scenario 3
       MultiAZ exists
       # Scenario 4 and 5
       MultiAZ == true
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        # Scenario 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
```

```
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      MultiAZ: true
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      MultiAZ: false
    DeletionPolicy: Delete
```

[CT.RDS.PR.2] Require an Amazon RDS database instance to have enhanced monitoring configured

This control checks whether enhanced monitoring is activated for an Amazon Relational Database Service (RDS) instances.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.2 example templates

Explanation

In Amazon RDS, enhanced monitoring facilitates a more rapid response to performance changes in underlying infrastructure. These performance changes could result in a lack of availability of the data. Enhanced monitoring provides real-time metrics of the operating system on which your RDS DB instance runs. An agent, installed on the instance, can obtain metrics more accurately than is possible from the hypervisor layer.

Enhanced monitoring metrics are useful when you want to see how different processes or threads on a database (DB) instance use the CPU.

Usage considerations

- This control applies only to Amazon RDS DB engine types `aurora`, `aurora-mysql`, `aurora-postgresql`, `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`
- This control applies to a standalone RDS instance and an RDS instance that is part of a DB cluster. Do not enable this control if you want to manage enhanced monitoring on the cluster level instead of requiring it to be specified on each RDS DB instance within the cluster. For more information on configuring enahanced monitoring at the cluster level, refer to Setting up and enabling Enhanced Monitoring in the *Amazon Aurora User Guide*.

Remediation for rule failure

Set `MonitoringInterval` to a supported value (1, 5, 10, 15, 30, 60), and set `MonitoringRoleArn` to the ARN of an AWS IAM role.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured with enhanced monitoring. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "MonitoringInterval": 30,
            "MonitoringRoleArn": {
                "Fn::GetAtt": [
                    "MonitoringIAMRole",
                    "Arn"
                ]
            }
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    MonitoringInterval: 30
    MonitoringRoleArn: !GetAtt 'MonitoringIAMRole.Arn'
  DeletionPolicy: Delete
```

CT.RDS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_enhanced_monitoring_enabled_check
#
# Description:
#   This control checks whether enhanced monitoring is activated for Amazon Relational
 Database Service (RDS) instances.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#       And: 'MonitoringInterval' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#       And: 'MonitoringInterval' has been specified
#       And: 'MonitoringInterval' has been set to '0'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#       And: 'MonitoringInterval' has been specified
#       And: 'MonitoringInterval' has not been set to a value from the list 1, 5, 10,
 15, 30, 60
#      Then: FAIL
```

```
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#       And: 'MonitoringInterval' has been specified
#       And: 'MonitoringInterval' has been set to a value from the list 1, 5, 10, 15,
 30, 60
#       And: 'MonitoringRoleArn' has not been specified or specified as an empty string
#      Then: FAIL
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-se1', 'oracle-se', 'postgres',
 'sqlserver-ee',
#            'sqlserver-se', 'sqlserver-ex', 'sqlserver-web'
#       And: 'MonitoringInterval' has been specified
#       And: 'MonitoringInterval' has been set to a value from the list 1, 5, 10, 15,
 30, 60
#       And: 'MonitoringRoleArn' has been specified with a non-empty string or valid
 local reference
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "aurora", "aurora-mysql", "aurora-postgresql", "mariadb", "mysql",
    "oracle-ee", "oracle-ee-cdb", "oracle-se2", "oracle-se2-cdb",
    "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]
let ALLOWED_EM_VALUES = [1, 5, 10, 15, 30, 60]

#
# Assignments
#
```

```
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_enhanced_monitoring_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.2]: Require an Amazon RDS database instance or cluster to have
 enhanced monitoring configured
        [FIX]: Set 'MonitoringInterval' to a supported value (1, 5, 10, 15, 30, 60),
 and set 'MonitoringRoleArn' to the ARN of an AWS IAM role.
        >>
}

rule rds_instance_enhanced_monitoring_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.2]: Require an Amazon RDS database instance or cluster to have
 enhanced monitoring configured
        [FIX]: Set 'MonitoringInterval' to a supported value (1, 5, 10, 15, 30, 60),
 and set 'MonitoringRoleArn' to the ARN of an AWS IAM role.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
        # Scenario: 3, 4, 5, 6 and 7
        MonitoringInterval exists
        MonitoringInterval in %ALLOWED_EM_VALUES
        # Scenario: 6 and 7
        MonitoringRoleArn exists
        check_for_valid_monitor_role_arn(MonitoringRoleArn)
    }
}

rule filter_engine(db_properties) {
    %db_properties {
```

```
        # Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

rule check_for_valid_monitor_role_arn(iam_role_arn) {
   %iam_role_arn {
      check_is_string_and_not_empty(this) or
      check_local_references(%INPUT_DOCUMENT, this, "AWS::IAM::Role")
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
```

```
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.RDS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  MonitoringIAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "monitoring.rds.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonRDSEnhancedMonitoringRole
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
```

```
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      MonitoringInterval: 30
      MonitoringRoleArn:
        Fn::GetAtt: ["MonitoringIAMRole", "Arn"]
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
```

```
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      MonitoringInterval: 0
    DeletionPolicy: Delete
```

[CT.RDS.PR.3] Require an Amazon RDS cluster to have deletion protection configured

This control checks whether your Amazon Relational Database Service (Amazon RDS) cluster has deletion protection activated.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.3 example templates

Explanation

Enabling cluster deletion protection is an additional layer of protection against accidental database deletion or deletion by an unauthorized entity.

When deletion protection is enabled, an Amazon RDS cluster cannot be deleted. Before a deletion request can succeed, deletion protection must be deactivated.

Remediation for rule failure

Set the value of the `DeletionProtection` parameter to true.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster with deletion protection enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RDSClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RDSClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "TestDBSubnetGroup"
            },
            "DeletionProtection": true
        }
    }
}
```

YAML example

```
RDSDBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora
    MasterUsername: !Sub '{{resolve:secretsmanager:${RDSClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RDSClusterSecret}::password}}'
```

```
    DBSubnetGroupName: !Ref 'TestDBSubnetGroup'
    DeletionProtection: true
```

CT.RDS.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_deletion_protection_enabled_check
#
# Description:
#   Checks if an Amazon Relational Database Service (Amazon RDS) cluster has deletion
 protection enabled.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#  None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'DeletionProtection' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'DeletionProtection' has been specified
#       And: 'DeletionProtection' has been set to bool(false)
```

```
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'DeletionProtection' has been specified
#       And: 'DeletionProtection' has been set to bool(true)
#      Then: PASS


#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_deletion_protection_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %db_clusters not empty {
    check(%db_clusters.Properties)
        <<
        [CT.RDS.PR.3]: Require an Amazon RDS cluster to have deletion protection
 configured
        [FIX]: Set the value of the 'DeletionProtection' parameter to true.
        >>
}

rule rds_cluster_deletion_protection_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.3]: Require an Amazon RDS cluster to have deletion protection
 configured
        [FIX]: Set the value of the 'DeletionProtection' parameter to true.
        >>
}
```

```
rule check(properties) {
    %properties {
        # Scenario 2
        DeletionProtection exists
        # Scenario 3 and 4
        DeletionProtection == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
```

```
      Ref: VPC
    CidrBlock: 10.0.0.0/25
    AvailabilityZone:
      Fn::Select:
      - 0
      - Fn::GetAZs: ''
SubnetTwo:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock: 10.0.0.128/25
    AvailabilityZone:
      Fn::Select:
      - 1
      - Fn::GetAZs: ''
    VpcId:
      Ref: VPC
DBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: Example DB subnet group
    SubnetIds:
    - Ref: SubnetOne
    - Ref: SubnetTwo
RDSClusterSecret:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: RDS cluster secret
    GenerateSecretString:
      SecretStringTemplate: '{"username": "exampleuser"}'
      GenerateStringKey: password
      PasswordLength: 32
      ExcludeCharacters: "/@\""
RDSCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername:
      Fn::Sub: "{{resolve:secretsmanager:${RDSClusterSecret}::username}}"
    MasterUserPassword:
      Fn::Sub: "{{resolve:secretsmanager:${RDSClusterSecret}::password}}"
    DBSubnetGroupName:
      Ref: DBSubnetGroup
    DeletionProtection: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Example DB subnet group
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  RDSClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
```

```
      Description: RDS cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\""
  RDSCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: "{{resolve:secretsmanager:${RDSClusterSecret}::username}}"
      MasterUserPassword:
        Fn::Sub: "{{resolve:secretsmanager:${RDSClusterSecret}::password}}"
      DBSubnetGroupName:
        Ref: DBSubnetGroup
      DeletionProtection: false
```

[CT.RDS.PR.4] Require an Amazon RDS database cluster to have AWS IAM database authentication configured

This control checks whether an Amazon Relational Database Service (RDS) database (DB) cluster has AWS IAM database authentication activated.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.4 example templates

Explanation

IAM database authentication allows for password-free authentication to database instances. The authentication uses an authentication token. Network traffic to and from the database is encrypted using SSL.

Usage considerations

- This control applies only to Amazon RDS DB cluster engine types `aurora`, `aurora-mysql` and `aurora-postgresql`.

Remediation for rule failure

Set `EnableIAMDatabaseAuthentication` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster configured with AWS IAM database authentication. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora-mysql",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "EnableIAMDatabaseAuthentication": true
        }
```

```
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    EnableIAMDatabaseAuthentication: true
```

CT.RDS.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_iam_authentication_enabled_check
#
# Description:
#   This control checks whether an Amazon Relational Database Service (RDS) database
 (DB) cluster has AWS IAM database authentication activated.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain any RDS DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is not one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'EnableIAMDatabaseAuthentication' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'EnableIAMDatabaseAuthentication' has been provided
#       And: 'EnableIAMDatabaseAuthentication' has been set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'EnableIAMDatabaseAuthentication' has been provided
#       And: 'EnableIAMDatabaseAuthentication' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let SUPPORTED_DB_CLUSTER_ENGINES = ["aurora", "aurora-mysql","aurora-postgresql"]
```

```
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_iam_authentication_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %db_clusters not empty {
    check(%db_clusters.Properties)
        <<
        [CT.RDS.PR.4]: Require an Amazon RDS database cluster to have AWS IAM database
 authentication configured
            [FIX]: Set 'EnableIAMDatabaseAuthentication' to 'true'.
        >>
}

rule rds_cluster_iam_authentication_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.4]: Require an Amazon RDS database cluster to have AWS IAM database
 authentication configured
            [FIX]: Set 'EnableIAMDatabaseAuthentication' to 'true'.
        >>
}

rule check(db_cluster) {
    %db_cluster [
        # Scenario 2
        filter_engine(this)
    ] {
       # Scenario 3
       EnableIAMDatabaseAuthentication exists
       # Scenario 4 and 5
       EnableIAMDatabaseAuthentication == true
    }
}

rule filter_engine(cluster_properties) {
    %cluster_properties {
```

```
        Engine exists
        Engine in %SUPPORTED_DB_CLUSTER_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
```

```
      - 0
      - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
      EnableIAMDatabaseAuthentication: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
```

```
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
```

[CT.RDS.PR.5] Require an Amazon RDS database instance to have minor version upgrades configured

This control checks whether automatic minor version upgrades are enabled for an Amazon Relational Database Service (RDS) database instance.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.5 example templates

Explanation

By activating automatic minor version upgrades, you can ensure that the latest minor version updates to the relational database management system (RDBMS) are installed. These upgrades might include security patches and bug fixes. Keeping up to date with patch installation is an important step in securing systems.

Usage considerations

- This control applies only to Amazon RDS DB engine types `aurora`, `aurora-mysql`, `aurora-postgresql`, `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Omit the `AutoMinorVersionUpgrade` property or set it to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example One

Amazon RDS DB instance configured with automatic minor version upgrades, enabled by means of AWS CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
```

```
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            }
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
  DeletionPolicy: Delete
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example Two

Amazon RDS DB instance configured with automatic minor version upgrades, enabled by means of the `AutoMinorVersionUpgrade` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
```

```
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "AutoMinorVersionUpgrade": true
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    AutoMinorVersionUpgrade: true
  DeletionPolicy: Delete
```

CT.RDS.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_automatic_minor_version_upgrade_enabled_check
```

```
#
# Description:
#   This control checks whether automatic minor version upgrades are enabled for an
 Amazon Relational Database Service (RDS) database instance.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-ee-cdb', 'oracle-se2', 'oracle-se2-cdb',
#            'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-web',
#            'postgres'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-ee-cdb', 'oracle-se2', 'oracle-se2-cdb',
#            'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-web',
#            'postgres'
#       And: 'AutoMinorVersionUpgrade' has been specified
#       And: 'AutoMinorVersionUpgrade' has been set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
```

```
#            'oracle-ee', 'oracle-ee-cdb', 'oracle-se2', 'oracle-se2-cdb',
#            'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-web',
#            'postgres'
#       And: 'AutoMinorVersionUpgrade' has not been specified
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-ee-cdb', 'oracle-se2', 'oracle-se2-cdb',
#            'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-web',
#            'postgres'
#       And: 'AutoMinorVersionUpgrade' has been specified
#       And: 'AutoMinorVersionUpgrade' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "aurora", "aurora-mysql", "aurora-postgresql", "mariadb", "mysql",
    "oracle-ee", "oracle-ee-cdb", "oracle-se2", "oracle-se2-cdb",
    "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_automatic_minor_version_upgrade_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                              %rds_db_instances not
 empty {
    check(%rds_db_instances.Properties)
        <<
```

```
        [CT.RDS.PR.5]: Require an Amazon RDS database instance to have minor version
 upgrades configured
        [FIX]: Omit the 'AutoMinorVersionUpgrade' property or set it to 'true'.
        >>
}

rule rds_instance_automatic_minor_version_upgrade_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.5]: Require an Amazon RDS database instance to have minor version
 upgrades configured
        [FIX]: Omit the 'AutoMinorVersionUpgrade' property or set it to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [ filter_engine(this) ] {
        # Scenario: 4
        AutoMinorVersionUpgrade not exists or
        # Scenario: 3 and 5
        AutoMinorVersionUpgrade == true
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        # Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      AutoMinorVersionUpgrade: false
    DeletionPolicy: Delete
```

[CT.RDS.PR.6] Require an Amazon RDS database cluster to have backtracking configured

This control checks whether an Amazon Relational Database Service (RDS) database (DB) cluster has backtracking enabled.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`

- **CloudFormation guard rule:** CT.RDS.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.6 example templates

Explanation

Backups help you to recover more quickly from a security incident. Backups also strengthen the resilience of your systems. Aurora backtracking reduces the time required to recover a database for a specific point in time, and the recovery does not require a database restore.

Usage considerations

- This control applies only to Amazon RDS DB cluster engine types `aurora` and `aurora-mysql`, and to DB cluster engine modes `provisioned` and `parallelquery`
- This control does not apply to Amazon RDS DB clusters that support Aurora Serverless V2 database instances (For example, RDS DB clusters configured with a `ServerlessV2ScalingConfiguration` and Aurora Serverless V2 compatible `EngineVersion`.)

Remediation for rule failure

Set `BacktrackWindow` to a number between `1` and `259200`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster configured with a backtrack window of 720 seconds (12 minutes). The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora-mysql",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "BacktrackWindow": 720
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    BacktrackWindow: 720
```

CT.RDS.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   aurora_cluster_backtracking_enabled_check
```

```
#
# Description:
#   This control checks whether an Amazon Relational Database Service (RDS) database
 (DB) cluster has backtracking enabled.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#    Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any RDS DB cluster resources
#       Then: SKIP
#    Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'Engine' provided is not one of 'aurora' or 'aurora-mysql'
#       Then: SKIP
#    Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'Engine' provided is one of 'aurora' or 'aurora-mysql'
#        And: 'EngineMode' provided is not one of 'provisioned' or 'parallelquery'
#       Then: SKIP
#    Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'ServerlessV2ScalingConfiguration' is provided
#        And: 'Engine' provided is 'aurora-mysql'
#        And: 'EngineVersion' provided is '8.0.mysql_aurora.3.02.0' or higher
#       Then: SKIP
#    Scenario: 5
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'Engine' provided is one of 'aurora' or 'aurora-mysql'
#        And: 'EngineMode' is not provided or 'EngineMode' provided is one of
 'provisioned' or 'parallelquery'
#        And: 'BacktrackWindow' has not been provided
```

```
#       Then: FAIL
#    Scenario: 6
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'Engine' provided is one of 'aurora' or 'aurora-mysql'
#        And: 'EngineMode' is not provided or 'EngineMode' provided is one of
 'provisioned' or 'parallelquery'
#        And: 'BacktrackWindow' has been provided and is set to 0
#       Then: FAIL
#    Scenario: 7
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'Engine' provided is one of 'aurora' or 'aurora-mysql'
#        And: 'EngineMode' is not provided or 'EngineMode' provided is one of
 'provisioned' or 'parallelquery'
#        And: 'BacktrackWindow' has been provided and is set to a value > 0
#       Then: PASS

#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let SUPPORTED_DB_CLUSTER_ENGINES = ["aurora", "aurora-mysql"]
let SUPPORTED_DB_CLUSTER_ENGINE_MODES = ["provisioned", "parallelquery"]
let AURORA_SERVERLESS_V2_SUPPORTED_ENGINE = ["aurora-mysql"]
let AURORA_V3_SERVERLESS_V2_NOT_SUPPORTED_PATTERN = /^8\.0\.mysql_aurora\.3\.01\./
let AURORA_V3_SERVERLESS_V2_SUPPORTED_PATTERN = /^8\.0\.mysql_aurora\.3/
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule aurora_cluster_backtracking_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %db_clusters not empty {
    check(%db_clusters.Properties)
        <<
        [CT.RDS.PR.6]: Require an Amazon RDS database cluster to have backtracking
 configured
            [FIX]: Set 'BacktrackWindow' to a number between '1' and '259200'.
```

```
        >>
}

rule aurora_cluster_backtracking_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.6]: Require an Amazon RDS database cluster to have backtracking
 configured
            [FIX]: Set 'BacktrackWindow' to a number between '1' and '259200'.
        >>
}

rule check(db_cluster) {
    %db_cluster [
        filter_engine_enginemode_and_serverless_v2(this)
    ] {
        # Scenario 5
        BacktrackWindow exists
        # Scenario 6 and 7
        BacktrackWindow > 0
    }
}

rule filter_engine_enginemode_and_serverless_v2(db_cluster) {
    # Scenario 2 and 3
    %db_cluster {
        Engine exists
        Engine in %SUPPORTED_DB_CLUSTER_ENGINES
        EngineMode not exists or
        EngineMode in %SUPPORTED_DB_CLUSTER_ENGINE_MODES
    }
    #Scenario 4
    %db_cluster [
        ServerlessV2ScalingConfiguration exists
        Engine in %AURORA_SERVERLESS_V2_SUPPORTED_ENGINE
    ] {
        EngineVersion in %AURORA_V3_SERVERLESS_V2_NOT_SUPPORTED_PATTERN or
        EngineVersion not in %AURORA_V3_SERVERLESS_V2_SUPPORTED_PATTERN
    }
}

#
# Utility Rules
```

```
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
```

```
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Example DB subnet group
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
      BacktrackWindow: 720
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
```

```
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Example DB subnet group
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
```

```
      EngineMode: provisioned
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
```

[CT.RDS.PR.7] Require Amazon RDS database instances to have IAM authentication configured

This control checks whether an Amazon RDS database (DB) instance has AWS Identity and Access Management (IAM) database authentication activated.

- **Control objective:** Use strong authentication
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.7 example templates

Explanation

IAM database authentication allows authentication to database instances with an authentication token instead of a password. Network traffic to and from the database is encrypted with SSL.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql` and `postgres`.

- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set `EnableIAMDatabaseAuthentication` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured with IAM database authentication. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "EnableIAMDatabaseAuthentication": true
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    EnableIAMDatabaseAuthentication: true
  DeletionPolicy: Delete
```

CT.RDS.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_iam_authentication_enabled_check
#
# Description:
#   This control checks whether an Amazon RDS database (DB) instance has AWS Identity
 and Access Management (IAM) database authentication activated.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
```

```
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not in-scope database engines - 'mariadb', 'mysql', 'postgres'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is in-scope database engines - 'mariadb', 'mysql', 'postgres'
#       And: 'EnableIAMDatabaseAuthentication' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is in-scope database engines - 'mariadb', 'mysql', 'postgres'
#       And: 'EnableIAMDatabaseAuthentication' has been specified
#       And: 'EnableIAMDatabaseAuthentication' has been set to bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is in-scope database engines - 'mariadb', 'mysql', 'postgres'
#       And: 'EnableIAMDatabaseAuthentication' has been specified
#       And: 'EnableIAMDatabaseAuthentication' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = ["mariadb", "mysql", "postgres"]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_iam_authentication_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
```

```
                                                        %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.7]: Require Amazon RDS database instances to have AWS IAM
 authentication configured
        [FIX]: Set 'EnableIAMDatabaseAuthentication' to 'true'.
        >>
}

rule rds_instance_iam_authentication_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.7]: Require Amazon RDS database instances to have AWS IAM
 authentication configured
        [FIX]: Set 'EnableIAMDatabaseAuthentication' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
        #Scenario: 3
        EnableIAMDatabaseAuthentication exists
        #Scenario: 4 and 5
        EnableIAMDatabaseAuthentication == true
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        #Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
```

```
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      EnableIAMDatabaseAuthentication: true
```

```
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      EnableIAMDatabaseAuthentication: false
    DeletionPolicy: Delete
```

[CT.RDS.PR.8] Require an Amazon RDS database instance to have automatic backups configured

This control checks whether Amazon RDS database (DB) instances have automated backups enabled, and verifies that the backup retention period is greater than or equal to seven (7) days.

- **Control objective:** Improve resiliency

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.8 example templates

Explanation

Backups help you recover more quickly from a security incident, and they strengthen the resilience of your systems. Amazon RDS provides an easy way to configure daily, full-instance volume snapshots.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set `BackupRetentionPeriod` to an integer value between 7 and 35 days (inclusive).

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured with automated backups configured and a backup retention period of 14 days. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "BackupRetentionPeriod": 14
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    BackupRetentionPeriod: 14
  DeletionPolicy: Delete
```

CT.RDS.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_backup_enabled_check
#
# Description:
#   This control checks whether Amazon RDS database (DB) instances have automated
 backups enabled, and verifies that the backup retention period is greater than or
 equal to seven (7) days.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
```

```
#       And: 'BackupRetentionPeriod' has been specified
#       And: 'BackupRetentionPeriod' has been set to 0 (backup disabled)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'BackupRetentionPeriod' has not been specified
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'BackupRetentionPeriod' has been specified
#       And: 'BackupRetentionPeriod' has been set to < 7
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'BackupRetentionPeriod' has been specified
#       And: 'BackupRetentionPeriod' has been set to an integer >= 7
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]
```

```
#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_backup_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.8]: Require an Amazon RDS database instance to have automatic
 backups configured
        [FIX]: Set 'BackupRetentionPeriod' to an integer value between 7 and 35 days
 (inclusive).
        >>
}

rule rds_instance_backup_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.8]: Require an Amazon RDS database instance to have automatic
 backups configured
        [FIX]: Set 'BackupRetentionPeriod' to an integer value between 7 and 35 days
 (inclusive).
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [ filter_db_identifier_and_engine(this) ] {
        # Scenario: 3, 4, 5 and 6
        BackupRetentionPeriod exists
        BackupRetentionPeriod >= 7
    }
}

rule filter_db_identifier_and_engine(db_properties) {
    %db_properties {
        # Scenario: 2
```

```
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}


#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
```

```
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      BackupRetentionPeriod: 14
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      BackupRetentionPeriod: 4
    DeletionPolicy: Delete
```

[CT.RDS.PR.9] Require an Amazon RDS database cluster to copy tags to snapshots

This control checks whether an Amazon RDS database (DB) cluster is configured to copy all tags to snapshots created.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.9 example templates

Explanation

Identification and inventory of your infrastructure assets is a crucial aspect of governance and security. With visibility into all your Amazon RDS DB clusters, you can assess their security posture and take action on potential areas of weakness. We recommend that you tag snapshots in the same way as their parent RDS database clusters. Activating this setting ensures that snapshots inherit the tags of their parent database clusters.

Usage considerations

- This control applies only to Amazon RDS DB cluster engine types `aurora`, `aurora-mysql`, and `aurora-postgresql`.

Remediation for rule failure

Set `CopyTagsToSnapshot` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster configured to copy tags to snapshots. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora-mysql",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "CopyTagsToSnapshot": true
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
```

```
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    CopyTagsToSnapshot: true
```

CT.RDS.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   rds_cluster_copy_tags_to_snapshots_enabled_check
#
# Description:
#   This control checks whether an Amazon RDS DB cluster is configured to copy all tags
 to snapshots created.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is not one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
```

```
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'CopyTagsToSnapshot' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'CopyTagsToSnapshot' has been provided
#       And: 'CopyTagsToSnapshot' has been set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' provided is one of 'aurora' or 'aurora-mysql' or 'aurora-
postgresql'
#       And: 'CopyTagsToSnapshot' has been provided
#       And: 'CopyTagsToSnapshot' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let SUPPORTED_DB_CLUSTER_ENGINES = ["aurora", "aurora-mysql","aurora-postgresql"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_copy_tags_to_snapshots_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %db_clusters not empty {
    check(%db_clusters.Properties)
        <<
        [CT.RDS.PR.9]: Require an Amazon RDS database cluster to copy tags to snapshots
            [FIX]: Set 'CopyTagsToSnapshot' to 'true'.
        >>
```

```
}

rule rds_cluster_copy_tags_to_snapshots_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.9]: Require an Amazon RDS database cluster to copy tags to snapshots
            [FIX]: Set 'CopyTagsToSnapshot' to 'true'.
        >>
}

rule check(db_cluster) {
    %db_cluster [
        # Scenario 2
        filter_engine(this)
    ] {
       # Scenario 3
       CopyTagsToSnapshot exists
       # Scenario 4 and 5
       CopyTagsToSnapshot == true
    }
}

rule filter_engine(cluster_properties) {
    %cluster_properties {
        Engine exists
        Engine in %SUPPORTED_DB_CLUSTER_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
```

```
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
      CopyTagsToSnapshot: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
```

```
      - 0
      - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
```

[CT.RDS.PR.10] Require an Amazon RDS database instance to copy tags to snapshots

This control checks whether Amazon RDS database (DB) instances are configured to copy all tags to snapshots created.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.10 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.10 example templates

Explanation

Identification and inventory of your IT assets is a crucial aspect of governance and security. With visibility of all your RDS DB instances, you can assess their security posture and take action on potential areas of weakness. Snapshots should be tagged to match their parent RDS database instances. Enabling this setting ensures that snapshots inherit the tags from their parent database instances.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set `CopyTagsToSnapshot` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured to copy all tags to snapshots created. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "CopyTagsToSnapshot": true
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
```

```
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    CopyTagsToSnapshot: true
  DeletionPolicy: Delete
```

CT.RDS.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_copy_tags_to_snapshots_enabled_check
#
# Description:
#   This control checks whether Amazon RDS database (DB) instances are configured to
 copy all tags to snapshots created.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
```

```
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'CopyTagsToSnapshot' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'CopyTagsToSnapshot' has been specified
#       And: 'CopyTagsToSnapshot' has been set to bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'CopyTagsToSnapshot' has been specified
#       And: 'CopyTagsToSnapshot' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
```

```
]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_copy_tags_to_snapshots_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                    %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.10]: Require an Amazon RDS database instance to copy tags to
 snapshots
        [FIX]: Set 'CopyTagsToSnapshot' to 'true'.
        >>
}

rule rds_instance_copy_tags_to_snapshots_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.10]: Require an Amazon RDS database instance to copy tags to
 snapshots
        [FIX]: Set 'CopyTagsToSnapshot' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
       # Scenario: 3
       CopyTagsToSnapshot exists
       # Scenario: 4 and 5
       CopyTagsToSnapshot == true
    }
}

rule filter_engine(db_properties) {
```

```
    %db_properties {
        # Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
```

```
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      CopyTagsToSnapshot: true
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      CopyTagsToSnapshot: false
```

```
    DeletionPolicy: Delete
```

[CT.RDS.PR.11] Require an Amazon RDS database instance to have a VPC configuration

This control checks whether an Amazon RDS database (DB) instance is deployed in a VPC (that is, with an EC2-VPC instance).

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.11 example templates

Explanation

Amazon Virtual Private Cloud (Amazon VPC) provides a number of network controls to create secure access to RDS resources. These controls include VPC endpoints, network ACLs, and security groups. To take advantage of these controls, create your Amazon RDS instances as EC2 VPC instances.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.

- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set a `DBSubnetGroupName`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured to deploy in an Amazon VPC with an RDS DB subnet group. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            }
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
  DeletionPolicy: Delete
```

CT.RDS.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_deployed_in_vpc_check
#
# Description:
#   This control checks whether an Amazon RDS database (DB) instance is deployed in a
 VPC (that is, an EC2 VPC instance).
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DBSubnetGroupName' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DBSubnetGroupName' has been specified but is an empty string
#            or invalid local reference to a DB Subnet Group
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DBSubnetGroupName' has been specified but is a non-empty string
#            or valid local reference to a DB Subnet Group
#      Then: PASS

#
# Constants
#
```

```
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]
let INPUT_DOCUMENT = this

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_deployed_in_vpc_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.11]: Require an Amazon RDS database instance to have a VPC
 configuration
        [FIX]: Set a 'DBSubnetGroupName'.
        >>
}

rule rds_instance_deployed_in_vpc_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.11]: Require an Amazon RDS database instance to have a VPC
 configuration
        [FIX]: Set a 'DBSubnetGroupName'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
       # Scenario: 3
       DBSubnetGroupName exists
```

```
       # Scenario: 4 and 5
       check_db_subnet_group(DBSubnetGroupName)
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        # Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

rule check_db_subnet_group(db_subnet_group) {
   %db_subnet_group {
      check_is_string_and_not_empty(this) or
      check_local_references(%INPUT_DOCUMENT, this, "AWS::RDS::DBSubnetGroup")
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
```

```
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.RDS.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.96.0/19
      AvailabilityZone:
        Fn::Select:
        - '0'
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId:
```

```
      Ref: VPC
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.128.0/19
      AvailabilityZone:
        Fn::Select:
        - '1'
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Test DB subnet group
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DeletionPolicy: Delete
```

[CT.RDS.PR.12] Require an Amazon RDS event subscription to have critical cluster events configured

This control checks whether your Amazon RDS event subscriptions for RDS clusters are configured to notify on event categories of `maintenance` and `failure`.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::RDS::EventSubscription`
- **CloudFormation guard rule:** CT.RDS.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.12 example templates

Explanation

Amazon RDS event notifications uses Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response.

Usage considerations

- This control applies only to Amazon RDS event subscriptions for RDS clusters (`SourceType` of `db-cluster`).

Remediation for rule failure

When `SourceType` is set to `db-cluster`, set `Enabled` to true and ensure that `EventCategories` contains both `maintenance` and `failure` values.

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example One

Amazon RDS event subscription for RDS clusters configured to notify on all event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
```

```
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "SourceType": "db-cluster",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    SourceType: db-cluster
    Enabled: true
```

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example Two

Amazon RDS event subscription for RDS clusters configured to notify on `maintenance` and `failure` event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "EventCategories": [
                "maintenance",
                "failure"
            ],
```

```
            "SourceType": "db-cluster",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    EventCategories:
      - maintenance
      - failure
    SourceType: db-cluster
    Enabled: true
```

CT.RDS.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_event_notifications_configured_check
#
# Description:
#   Checks whether an Amazon RDS event subscriptions for RDS clusters is configured to
 notify on event categories of "maintenance" and "failure".
#
# Reports on:
#   AWS::RDS::EventSubscription
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS event subscription
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is not 'db-cluster'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is 'db-cluster'
#       And: 'Enabled' is not provided or set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-cluster'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not contain both 'maintenance' and 'failure'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-cluster'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not exist or is an empty list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-cluster'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' contains both 'maintenance' and 'failure'
#      Then: PASS

#
# Constants
#
```

```
let RDS_EVENTSUBSCRIPTION_TYPE = "AWS::RDS::EventSubscription"
let INPUT_DOCUMENT = this
let EVENT_CATEGORIES = ["maintenance","failure"]
let EVENT_SOURCE_TYPE = "db-cluster"

#
# Assignments
#
let rds_event_subscriptions = Resources.*[ Type == %RDS_EVENTSUBSCRIPTION_TYPE ]

#
# Primary Rules
#
rule rds_cluster_event_notifications_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %rds_event_subscriptions not
 empty {
    check(%rds_event_subscriptions.Properties)
        <<
        [CT.RDS.PR.12]: Require an Amazon RDS event subscription to have critical
 cluster events configured
        [FIX]: When 'SourceType' is set to 'db-cluster', set 'Enabled' to true and
 ensure that 'EventCategories' contains both 'maintenance' and 'failure' values.
        >>
}

rule rds_cluster_event_notifications_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_EVENTSUBSCRIPTION_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_EVENTSUBSCRIPTION_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.12]: Require an Amazon RDS event subscription to have critical
 cluster events configured
        [FIX]: When 'SourceType' is set to 'db-cluster', set 'Enabled' to true and
 ensure that 'EventCategories' contains both 'maintenance' and 'failure' values.
        >>
}

#
# Parameterized Rules
#
rule check(resource) {
    %resource [ SourceType == %EVENT_SOURCE_TYPE ] {
        Enabled exists
        # Scenario 4
```

```
        Enabled == true
        # Scenario 5
        EventCategories not exists or
        # Scenario 6
        check_event_categories_for_required_events(EventCategories)
    }
}

rule check_event_categories_for_required_events(event_categories) {
    %event_categories {
        this exists
        this is_list
        this empty or
        %EVENT_CATEGORIES.* in this
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
```

```
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      SourceType: db-cluster
      Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      EventCategories:
      - maintenance
      - deletion
      SourceType: db-cluster
      Enabled: true
```

[CT.RDS.PR.13] Require any Amazon RDS instance to have deletion protection configured

This control checks whether an Amazon Relational Database Service (Amazon RDS) instance has deletion protection activated.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive

- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.13 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.13 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.13 example templates

Explanation

When active, instance deletion protection provides an additional layer of protection against accidental database deletion, or deletion by an unauthorized entity.

While deletion protection is active, an RDS DB instance cannot be deleted. Before a deletion request can succeed, deletion protection must be turned off.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set `DeletionProtection` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB instance - Example

Amazon RDS DB instance configured with deletion protection active. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 5.7,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true,
            "DeletionProtection": true
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 5.7
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    StorageEncrypted: true
    DeletionProtection: true
  DeletionPolicy: Delete
```

CT.RDS.PR.13 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_deletion_protection_enabled_check
#
# Description:
#   This control checks whether an Amazon Relational Database Service (Amazon RDS)
 instance has deletion protection activated.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
```

```
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DeletionProtection' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DeletionProtection' has been specified
#       And: 'DeletionProtection' has been set to bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'DeletionProtection' has been specified
#       And: 'DeletionProtection' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
```

```
rule rds_instance_deletion_protection_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.13]: Require any Amazon RDS instance to have deletion protection
 configured
        [FIX]: Set 'DeletionProtection' to 'true'.
        >>
}

rule rds_instance_deletion_protection_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.13]: Require any Amazon RDS instance to have deletion protection
 configured
        [FIX]: Set 'DeletionProtection' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
       #Scenario: 3
       DeletionProtection exists
       #Scenario: 4 and 5
       DeletionProtection == true
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        #Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}


#
```

```
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.13 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
```

```
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      DeletionProtection: true
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      DeletionProtection: false
    DeletionPolicy: Delete
```

[CT.RDS.PR.14] Require an Amazon RDS database instance to export logs to Amazon CloudWatch Logs by means of the EnableCloudwatchLogsExports property

This rule checks whether Amazon Relational Database Service (RDS) instances have all available log types configured for export to Amazon CloudWatch Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.14 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.14 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.14 example templates

Explanation

AWS Control Tower recommends that you enable th export of relevant logs for all Amazon RDS databases to Amazon CloudWatch Logs. Database logging provides detailed records of requests made to RDS. Database logs can assist with security and access audits, and they can help you diagnose availability issues.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex`, `sqlserver-web`, `oracle-ee`, `oracle-se2`, `oracle-se1`, and `oracle-se`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

- Additional prerequisites may exist for enabling logging based on your selected database engine type. Refer to Monitoring Amazon RDS log files in the *Amazon RDS User Guide* for more information.

Remediation for rule failure

Specify `EnableCloudwatchLogsExports` with a list of all supported log types for the Amazon RDS database instance engine.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example One

Amazon RDS DB instance configured with an engine type of `mysql` and all supported log types, for the `mysql` engine type. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "mysql",
            "EngineVersion": 5.7,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true,
            "EnableCloudwatchLogsExports": [
                "error",
                "general",
                "slowquery",
                "audit"
            ]
        },
```

```
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: mysql
    EngineVersion: 5.7
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    StorageEncrypted: true
    EnableCloudwatchLogsExports:
      - error
      - general
      - slowquery
      - audit
  DeletionPolicy: Delete
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example Two

Amazon RDS DB instance configured with an engine type of `postgres` and all supported log types, for the `postgres` engine type. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
```

```
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "EnableCloudwatchLogsExports": [
                "postgresql",
                "upgrade"
            ]
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    EnableCloudwatchLogsExports:
      - postgresql
      - upgrade
  DeletionPolicy: Delete
```

CT.RDS.PR.14 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_logging_enabled_check
#
# Description:
#   This rules checks whether Amazon Relational Database Service (RDS) instances have
 all available log types configured for export to Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#       And: 'EnableCloudwatchLogsExports' has not been specified or has been specified
#             and is an empty list
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#       And: 'EnableCloudwatchLogsExports' has been specified and is a non-empty list
#       And: One or more log types in 'EnableCloudwatchLogsExports' are not supported
 by the specified 'Engine'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#       And: 'EnableCloudwatchLogsExports' has been specified and is a non-empty list
#       And: 'EnableCloudwatchLogsExports' does not contain all log types supported by
 the specified 'Engine'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine'  is one of 'mariadb', 'mysql'
#       And: 'EnableCloudwatchLogsExports' has been specified
#       And: 'EnableCloudwatchLogsExports' value is a non-empty and all supported log
 types
#             are enabled - 'audit', 'error', 'general', 'slowquery'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is 'postgres'
#       And: 'EnableCloudwatchLogsExports' has been specified
#       And: 'EnableCloudwatchLogsExports' value is a non-empty and all supported log
 types
#             are enabled - 'postgresql', 'upgrade'
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex',
#            'sqlserver-web'
```

```
#       And: 'EnableCloudwatchLogsExports' has been specified
#       And: 'EnableCloudwatchLogsExports' value is a non-empty and all supported log
 types
#             are enabled - 'agent', 'error'
#      Then: PASS
#   Scenario: 9
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-
se2-cdb',
#       And: 'EnableCloudwatchLogsExports' has been specified
#       And: 'EnableCloudwatchLogsExports' value is a non-empty and all supported log
 types
#             are enabled - 'alert', 'audit', 'listener', 'oemagent', 'trace'
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]

let MYSQL_OR_MARIA_ENGINES_SUBTYPES = [ "mariadb", "mysql" ]
let POSTGRES_ENGINES_SUBTYPES = [ "postgres" ]
let SQLSERVER_ENGINES_SUBTYPES = [ "sqlserver-ee", "sqlserver-se", "sqlserver-ex",
 "sqlserver-web" ]
let ORACLE_ENGINES_SUBTYPES = [ "oracle-ee", "oracle-se2", "oracle-se1", "oracle-se" ]

let MYSQL_OR_MARIA_SUPPORTED_LOG_TYPES = [ "audit", "error", "general", "slowquery" ]
let POSTGRES_SUPPORTED_LOG_TYPES = [ "postgresql", "upgrade" ]
let SQLSERVER_SUPPORTED_LOG_TYPES = [ "agent", "error" ]
let ORACLE_SUPPORTED_LOG_TYPES = [ "alert", "audit", "listener", "oemagent", "trace" ]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
```

```
# Primary Rules
#
rule rds_instance_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.14]: Require an Amazon RDS database instance to have logging
 configured
            [FIX]: Specify 'EnableCloudwatchLogsExports' with a list of all supported
 log types for the Amazon RDS database instance engine.
        >>
}

rule rds_instance_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.14]: Require an Amazon RDS database instance to have logging
 configured
            [FIX]: Specify 'EnableCloudwatchLogsExports' with a list of all supported
 log types for the Amazon RDS database instance engine.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_engine(this)] {
        # Scenario: 3
        EnableCloudwatchLogsExports exists
        check_is_list_and_not_empty(EnableCloudwatchLogsExports)

        # Scenario: 4 and 5
        when Engine IN %MYSQL_OR_MARIA_ENGINES_SUBTYPES {
            %MYSQL_OR_MARIA_SUPPORTED_LOG_TYPES.* IN EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %MYSQL_OR_MARIA_SUPPORTED_LOG_TYPES[*]
        }

        # Scenario: 4 and 6
        when Engine IN %POSTGRES_ENGINES_SUBTYPES {
            %POSTGRES_SUPPORTED_LOG_TYPES.* IN EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %POSTGRES_SUPPORTED_LOG_TYPES[*]
        }
```

```
        # Scenario: 4 and 7
        when Engine IN %SQLSERVER_ENGINES_SUBTYPES {
            %SQLSERVER_SUPPORTED_LOG_TYPES.* in EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %SQLSERVER_SUPPORTED_LOG_TYPES[*]
        }

        # Scenario: 4 and 8
        when Engine IN %ORACLE_ENGINES_SUBTYPES {
            %ORACLE_SUPPORTED_LOG_TYPES.* in EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %ORACLE_SUPPORTED_LOG_TYPES[*]
        }
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        # Scenario: 2
        Engine exists
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule check_is_list_and_not_empty(value) {
    %value {
        this is_list
        this not empty
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.14 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: true
      EnableCloudwatchLogsExports:
      - error
      - general
      - slowquery
      - audit
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: true
      EnableCloudwatchLogsExports:
      - error
      - general
      - slowquery
    DeletionPolicy: Delete
```

[CT.RDS.PR.15] Require that an Amazon RDS instance does not create DB security groups

This control checks whether any Amazon Relational Database Service (RDS) database (DB) security groups are created by, or associated to, an RDS DB instance, because DB security groups are intended for the EC2-Classic platform only.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`, `AWS::RDS::DBSecurityGroup`
- **CloudFormation guard rule:** CT.RDS.PR.15 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.15 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.15 example templates

Explanation

We recommend that all Amazon Relational Database Service (RDS) databases use Amazon VPC security groups to secure their access. Amazon DB security groups are for the EC2-Classic platform only, and they are not recommended for use.

Remediation for rule failure

Omit the `DBSecurityGroups` property. Instead, configure Amazon VPC security groups by means of the `VPCSecurityGroups` property.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured with an Amazon VPC security group. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "mysql",
```

```
            "EngineVersion": 5.7,
            "DBInstanceClass": "db.t3.small",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true,
            "Port": 6733,
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "VPCSecurityGroups": [
                {
                    "Ref": "SecurityGroup"
                }
            ]
        }
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: mysql
    EngineVersion: 5.7
    DBInstanceClass: db.t3.small
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    StorageEncrypted: true
    Port: 6733
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    VPCSecurityGroups:
      - !Ref 'SecurityGroup'
```

CT.RDS.PR.15 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_db_security_group_not_allowed_check
#
# Description:
#   This control checks whether any Amazon Relational Database Service (RDS) database
 (DB) security groups are created by, or associated to, an RDS DB instance, because DB
 security groups are intended for the EC2-Classic platform only.
#
# Reports on:
#   AWS::RDS::DBSecurityGroup, AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any DB security group resources
#        And: The input document does not contain any RDS DB instance resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains a DB security group resource
#       Then: FAIL
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any DBsecurity group resources
#        And: The input document contains an RDS DB instance resource
#        And: 'DBSecurityGroups' has been specified on the RDS DB instance as a non
 empty list
```

```
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any DB security group resources
#        And: The input document contains an RDS DB instance resource
#        And: 'DBSecurityGroups' has not been specified on the RDS DB instance or
 specified as an empty list
#       Then: PASS

#
# Constants
#
let DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let DB_SECURITY_GROUP_TYPE = "AWS::RDS::DBSecurityGroup"
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_instances = Resources.*[ Type == %DB_INSTANCE_TYPE ]
let db_security_groups = Resources.*[ Type == %DB_SECURITY_GROUP_TYPE ]

#
# Primary Rules
#
rule rds_db_security_group_not_allowed_check when is_cfn_template(this)
                                                  %db_security_groups not empty {
    check_db_security_group(%db_security_groups)
        <<
        [CT.RDS.PR.15]: Require that an Amazon RDS instance does not create DB security
 groups
            [FIX]: Omit the 'DBSecurityGroups' property. Instead, configure Amazon VPC
 security groups by means of the 'VPCSecurityGroups' property.
        >>
}

rule rds_db_security_group_not_allowed_check when is_cfn_template(this)
                                                  %db_instances not empty {
    check_db_instance(%db_instances.Properties)
        <<
        [CT.RDS.PR.15]: Require that an Amazon RDS instance does not create DB security
 groups
            [FIX]: Omit the 'DBSecurityGroups' property. Instead, configure Amazon VPC
 security groups by means of the 'VPCSecurityGroups' property.
```

```
        >>
}

rule rds_db_security_group_not_allowed_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DB_SECURITY_GROUP_TYPE) {
    check_db_security_group(%INPUT_DOCUMENT.%DB_SECURITY_GROUP_TYPE)
        <<
        [CT.RDS.PR.15]: Require that an Amazon RDS instance does not create DB security
 groups
            [FIX]: Omit the 'DBSecurityGroups' property. Instead, configure Amazon VPC
 security groups by means of the 'VPCSecurityGroups' property.
        >>
}

rule rds_db_security_group_not_allowed_check when is_cfn_hook(%INPUT_DOCUMENT,
 %DB_INSTANCE_TYPE) {
    check_db_instance(%INPUT_DOCUMENT.%DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.15]: Require that an Amazon RDS instance does not create DB security
 groups
            [FIX]: Omit the 'DBSecurityGroups' property. Instead, configure Amazon VPC
 security groups by means of the 'VPCSecurityGroups' property.
        >>
}

#
# Parameterized Rules
#
rule check_db_security_group(db_security_group) {
    # Scenario 2
    %db_security_group empty
}

rule check_db_instance(db_instance) {
    %db_instance {
        # Scenario 3 and 4
        DBSecurityGroups not exists or
        check_is_empty_list(this)
    }
}

rule check_is_empty_list(db_instance_configuration) {
    %db_instance_configuration {
        DBSecurityGroups is_list
```

```
        DBSecurityGroups empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.15 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.96.0/19
      AvailabilityZone:
        Fn::Select:
        - '0'
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId:
```

```
      Ref: VPC
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.128.0/19
      AvailabilityZone:
        Fn::Select:
        - '1'
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Example Security Group
      VpcId:
        Ref: VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 6733
        ToPort: 6733
        CidrIp: 10.0.0.0/16
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.t3.small
```

```
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: true
      Port: 6733
      DBSubnetGroupName:
        Ref: DBSubnetGroup
      VPCSecurityGroups:
      - Ref: SecurityGroup
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBSecurityGroup:
    Type: AWS::RDS::DBSecurityGroup
    Properties:
      DBSecurityGroupIngress:
      - CIDRIP: "0.0.0.0/0"
      GroupDescription: "Ingress for Amazon EC2 security group"
```

[CT.RDS.PR.16] Require an Amazon RDS database cluster to have encryption at rest configured

This control checks whether the storage encryption is configured on Amazon Relational Database Service (RDS) database (DB) clusters that are not being restored from an existing cluster.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`

- **CloudFormation guard rule:** CT.RDS.PR.16 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.16 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.16 example templates

Explanation

We recommend that you configure your Amazon RDS DB clusters to be encrypted at rest, to give an added layer of security for your sensitive data. To encrypt your RDS DB clusters and snapshots at rest, enable the encryption option for your RDS DB clusters. Data that is encrypted at rest includes the underlying storage for DB clusters, its automated backups, read replicas, and snapshots.

Encrypted RDS DB clusters use the open standard AES-256 encryption algorithm to encrypt your data on the server that hosts the clusters. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data with a minimal impact on performance. You do not need to modify your database client applications to use encryption.

Usage considerations

- This control applies only to Amazon RDS DB clusters that are not being restored from an existing cluster or created as a read replica. (For example, when `SourceDBClusterIdentifier` or `ReplicationSourceIdentifier` properties have been provided.)

Remediation for rule failure

Set `StorageEncrypted` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster configured with storage encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora-mysql",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "StorageEncrypted": true
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    StorageEncrypted: true
```

CT.RDS.PR.16 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   rds_cluster_storage_encrypted_check
#
```

```
# Description:
#   This control checks whether the storage encryption is configured on Amazon
 Relational Database Service (RDS) database (DB) clusters that are not being restored
 from an existing cluster.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any RDS DB cluster resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'SourceDBClusterIdentifier' or 'ReplicationSourceIdentifier' has been
 provided
#       Then: SKIP
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'SourceDBClusterIdentifier' or 'ReplicationSourceIdentifier' has not been
 provided
#        And: 'StorageEncrypted' has not been provided
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'SourceDBClusterIdentifier' or 'ReplicationSourceIdentifier' has not been
 provided
#        And: 'StorageEncrypted' has been provided and set to bool(false)
#       Then: FAIL
#   Scenario: 5
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB cluster resource
#        And: 'SourceDBClusterIdentifier' or 'ReplicationSourceIdentifier' has not been
 provided
```

```
#        And: 'StorageEncrypted' has been provided and set to bool(true)
#       Then: PASS

#
# Constants
#
let RDS_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let rds_cluster = Resources.*[ Type == %RDS_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_storage_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %rds_cluster not empty {
    check(%rds_cluster.Properties)
        <<
        [CT.RDS.PR.16]: Require an Amazon RDS database cluster to have encryption at
 rest configured
            [FIX]: Set 'StorageEncrypted' to 'true'.
        >>
}

rule rds_cluster_storage_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.16]: Require an Amazon RDS database cluster to have encryption at
 rest configured
            [FIX]: Set 'StorageEncrypted' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_cluster) {
    %rds_cluster [
        # Scenario 2
        filter_sources(this)
```

```
    ] {
        # Scenario 3
        StorageEncrypted exists

        # Scenario 4 and 5
        StorageEncrypted == true
    }
}

rule filter_sources(rds_cluster) {
    %rds_cluster {
        # Scenario 2
        SourceDBClusterIdentifier not exists or
        filter_property_is_empty_string(SourceDBClusterIdentifier) or
        filter_is_not_valid_local_reference(%INPUT_DOCUMENT, SourceDBClusterIdentifier,
 "AWS::RDS::DBCluster")

        ReplicationSourceIdentifier not exists or
        filter_property_is_empty_string(ReplicationSourceIdentifier) or
        filter_replication_source_identifier(ReplicationSourceIdentifier)
    }
}

rule filter_property_is_empty_string(value) {
    %value {
        this is_string
        this == /\A\s*\z/
    }
}

rule filter_is_not_valid_local_reference(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        this not is_string
        this is_struct

        when this.'Ref' exists {
            'Ref' {
                when query_for_resource(%doc, this, %referenced_resource_type) {
                    this not exists
                }
                this exists
            }
        }
```

```
        when this.'Ref' not exists {
            this exists
        }
    }
}

rule filter_replication_source_identifier(reference_properties) {
    filter_is_not_valid_local_reference_via_join(%INPUT_DOCUMENT,
 %reference_properties, "AWS::RDS::DBCluster")
    filter_is_not_valid_local_reference_via_join(%INPUT_DOCUMENT,
 %reference_properties, "AWS::RDS::DBInstance")
}

rule filter_is_not_valid_local_reference_via_join(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        this not is_string
        this is_struct

        when this.'Fn::Join' exists {
            'Fn::Join' {
                when filter_list_contains_valid_local_reference(this[1], %doc,
 %referenced_resource_type) {
                    this not exists
                }
                this exists
            }
        }
        when this.'Fn::Join' not exists {
            this exists
        }
    }
}

rule filter_list_contains_valid_local_reference(list, doc, referenced_resource_type) {
     some %list.* {
         check_local_references(%doc, this, %referenced_resource_type)
     }
 }

#
# Utility Rules
#
rule is_cfn_template(doc) {
```

```
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.RDS.PR.16 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
```

```
      Description: RDS Cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      StorageEncrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS Cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      StorageEncrypted: false
```

[CT.RDS.PR.17] Require an Amazon RDS event notification subscription to have critical database instance events configured

This control checks whether your Amazon RDS event subscriptions for RDS instances are configured to notify on event categories of `maintenance`, `failure`, and `configuration change`.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::EventSubscription`
- **CloudFormation guard rule:** CT.RDS.PR.17 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.17 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.17 example templates

Explanation

Amazon RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response.

Usage considerations

- This control applies only to Amazon RDS Event Subscriptions for RDS instances (`SourceType` of `db-instance`).

Remediation for rule failure

When `SourceType` is set to `db-instance`, set `Enabled` to true and ensure that the parameter `EventCategories` contains `maintenance`, `failure`, and `configuration change` values.

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example One

Amazon RDS Event Subscription for RDS instances configured to notify on all event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "SourceType": "db-instance",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    SourceType: db-instance
    Enabled: true
```

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example Two

Amazon RDS Event Subscription for RDS instances configured to notify on `maintenance`, `failure`, and `configuration change` event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "EventCategories": [
                "maintenance",
                "failure",
                "configuration change"
            ],
            "SourceType": "db-instance",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    EventCategories:
      - maintenance
      - failure
      - configuration change
    SourceType: db-instance
    Enabled: true
```

CT.RDS.PR.17 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   rds_instance_event_notifications_configured_check
#
# Description:
#   Checks whether Amazon RDS event subscriptions for RDS instances are configured to
 notify on event categories of 'maintenance', 'failure', and 'configuration change'.
#
# Reports on:
#   AWS::RDS::EventSubscription
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS event subscription
 Resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is not 'db-instance'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is 'db-instance'
#       And: 'Enabled' is not provided or set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-instance'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not contain 'maintenance', 'failure', and
 'configuration change'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-instance'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not exist or is an empty list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-instance'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' contains 'maintenance', 'failure', and 'configuration
 change'
#      Then: PASS

#
# Constants
#
let RDS_EVENTSUBSCRIPTION_TYPE = "AWS::RDS::EventSubscription"
let INPUT_DOCUMENT = this
let EVENT_CATEGORIES = ["maintenance","failure","configuration change"]
let EVENT_SOURCE_TYPE = "db-instance"

#
# Assignments
#
let rds_event_subscriptions = Resources.*[ Type == %RDS_EVENTSUBSCRIPTION_TYPE ]

#
# Primary Rules
#
rule rds_instance_event_notifications_configured_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                          %rds_event_subscriptions
 not empty  {
    check(%rds_event_subscriptions.Properties)
        <<
        [CT.RDS.PR.17]: Require an Amazon RDS event notification subscription to have
 critical database instance events configured
        [FIX]: When 'SourceType' is set to 'db-instance', set 'Enabled' to true and
 ensure that the parameter 'EventCategories' contains 'maintenance', 'failure', and
 'configuration change' values.
        >>
}
```

```
rule rds_instance_event_notifications_configured_check when
 is_cfn_hook(%INPUT_DOCUMENT, %RDS_EVENTSUBSCRIPTION_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_EVENTSUBSCRIPTION_TYPE.resourceProperties)
       <<
        [CT.RDS.PR.17]: Require an Amazon RDS event notification subscription to have
 critical database instance events configured
        [FIX]: When 'SourceType' is set to 'db-instance', set 'Enabled' to true and
 ensure that the parameter 'EventCategories' contains 'maintenance', 'failure', and
 'configuration change' values.
        >>
}

#
# Parameterized Rules
#
rule check(resource) {
    %resource [ SourceType == %EVENT_SOURCE_TYPE ] {
        Enabled exists
        # Scenario 4
        Enabled == true
        # Scenario 5
        EventCategories not exists or
        # Scenario 6
        check_event_categories_for_required_events(EventCategories)
    }
}

rule check_event_categories_for_required_events(event_categories) {
    %event_categories {
        this exists
        this is_list
        this empty or
        %EVENT_CATEGORIES.* in this
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
```

```
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.17 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      SourceType: db-instance
      Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
```

```
      Ref: SNSTopic
    EventCategories:
    - maintenance
    - failure
    SourceType: db-instance
    Enabled: true
```

[CT.RDS.PR.18] Require an Amazon RDS event notification subscription to have critical database parameter group events configured

This control checks whether your Amazon RDS event subscriptions for RDS parameter groups are configured to notify on event categories of `configuration change`.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::EventSubscription`
- **CloudFormation guard rule:** CT.RDS.PR.18 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.18 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.18 example templates

Explanation

Amazon RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response.

Usage considerations

- This control applies only to Amazon RDS event subscriptions for RDS parameter groups (`SourceType` of `db-parameter-group`).

Remediation for rule failure

When `SourceType` is set to `db-parameter-group`, set `Enabled` to true and ensure that the parameter `EventCategories` contains `configuration change` as a value.

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example One

Amazon RDS event subscription for RDS parameter groups configured to notify on all event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "SourceType": "db-parameter-group",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    SourceType: db-parameter-group
    Enabled: true
```

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example Two

Amazon RDS event subscription for RDS parameter groups configured to notify on the `configuration change` event category. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "EventCategories": [
                "configuration change"
            ],
            "SourceType": "db-parameter-group",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    EventCategories:
      - configuration change
    SourceType: db-parameter-group
    Enabled: true
```

CT.RDS.PR.18 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   rds_pg_event_notifications_configured_check
#
# Description:
#   Checks whether Amazon RDS event subscriptions for RDS parameter groups are
 configured to notify on event categories of 'configuration change'.
#
# Reports on:
#   AWS::RDS::EventSubscription
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS event subscription resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS event subscription resource
#       And: 'SourceType' is provided and is not 'db-parameter-group'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS event subscription resource
#       And: 'SourceType' is 'db-parameter-group'
#       And: 'Enabled' is not provided or set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-parameter-group'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not contain 'configuration change'
#      Then: FAIL
#   Scenario: 5
```

```
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-parameter-group'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not exist or is an empty list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an AWS CloudFormation or CloudFormation hook
 document
#       And: The input document contains an RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-parameter-group'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' contains 'configuration change'
#      Then: PASS

#
# Constants
#
let RDS_EVENTSUBSCRIPTION_TYPE = "AWS::RDS::EventSubscription"
let INPUT_DOCUMENT = this
let EVENT_CATEGORIES = ["configuration change"]
let EVENT_SOURCE_TYPE = "db-parameter-group"

#
# Assignments
#
let rds_event_subscriptions = Resources.*[ Type == %RDS_EVENTSUBSCRIPTION_TYPE ]

#
# Primary Rules
#
rule rds_pg_event_notifications_configured_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %rds_event_subscriptions not
 empty  {
    check(%rds_event_subscriptions.Properties)
        <<
        [CT.RDS.PR.18]: Require an Amazon RDS event notification subscription to have
 critical database parameter group events configured
        [FIX]: When 'SourceType' is set to 'db-parameter-group', set 'Enabled' to true
 and ensure that the parameter 'EventCategories' contains 'configuration change' as a
 value.
        >>
}
```

```
rule rds_pg_event_notifications_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_EVENTSUBSCRIPTION_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_EVENTSUBSCRIPTION_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.18]: Require an Amazon RDS event notification subscription to have
 critical database parameter group events configured
        [FIX]: When 'SourceType' is set to 'db-parameter-group', set 'Enabled' to true
 and ensure that the parameter 'EventCategories' contains 'configuration change' as a
 value.
        >>
}

#
# Parameterized Rules
#
rule check(resource) {
    %resource [ SourceType == %EVENT_SOURCE_TYPE ] {
        Enabled exists
        # Scenario 4
        Enabled == true
        # Scenario 5
        EventCategories not exists or
        # Scenario 6
        check_event_categories_for_required_events(EventCategories)
    }
}

rule check_event_categories_for_required_events(event_categories) {
    %event_categories {
        this exists
        this is_list
        this empty or
        %EVENT_CATEGORIES.* in this
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
```

```
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.18 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      SourceType: db-parameter-group
      Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
```

```
      SnsTopicArn:
        Ref: SNSTopic
      Enabled: false
      SourceType: db-parameter-group
```

[CT.RDS.PR.19] Require an Amazon RDS event notifications subscription to have critical database security group events configured

This control checks whether your Amazon RDS event subscriptions for RDS security groups are configured to notify on event categories of `failure` and `configuration change`

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::EventSubscription`
- **CloudFormation guard rule:** CT.RDS.PR.19 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.19 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.19 example templates

Explanation

Amazon RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for a rapid response.

Usage considerations

- This control applies only to Amazon RDS Event Subscriptions for RDS security groups (`SourceType` of `db-security-group`)

Remediation for rule failure

When `SourceType` is set to `db-security-group`, set `Enabled` to `true` and ensure that the parameter `EventCategories` contains both `failure` and `configuration change` values.

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example One

Amazon RDS Event Subscription for RDS security groups configured to notify on all event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "SourceType": "db-security-group",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    SourceType: db-security-group
    Enabled: true
```

The examples that follow show how to implement this remediation.

Amazon RDS Event Subscription - Example Two

Amazon RDS Event Subscription for RDS security groups configured to notify on `failure` and `configuration change` event categories. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSEventSubscription": {
        "Type": "AWS::RDS::EventSubscription",
        "Properties": {
            "SnsTopicArn": {
                "Ref": "SnsTopic"
            },
            "EventCategories": [
                "failure",
                "configuration change"
            ],
            "SourceType": "db-security-group",
            "Enabled": true
        }
    }
}
```

YAML example

```
RDSEventSubscription:
  Type: AWS::RDS::EventSubscription
  Properties:
    SnsTopicArn: !Ref 'SnsTopic'
    EventCategories:
      - failure
      - configuration change
    SourceType: db-security-group
    Enabled: true
```

CT.RDS.PR.19 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_sg_event_notifications_configured_check
#
# Description:
#   Checks whether an Amazon RDS event subscription for RDS security groups are
 configured to notify on event categories of 'failure' and 'configuration change'.
#
# Reports on:
#   AWS::RDS::EventSubscription
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS event subscription
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription Resource
#       And: 'SourceType' is provided and is not 'db-security-group'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is 'db-security-group'
#       And: 'Enabled' is not provided or set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-security-group'
```

```
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not contain both 'failure' and 'configuration
 change'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-security-group'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' does not exist or is an empty list
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS event subscription resource
#       And: 'SourceType' is provided and is 'db-security-group'
#       And: 'Enabled' is provided and set to bool(true)
#       And: 'EventCategories' contains both 'failure' and 'configuration change'
#      Then: PASS

#
# Constants
#
let RDS_EVENTSUBSCRIPTION_TYPE = "AWS::RDS::EventSubscription"
let INPUT_DOCUMENT = this
let EVENT_CATEGORIES = ["failure","configuration change"]
let EVENT_SOURCE_TYPE = "db-security-group"

#
# Assignments
#
let rds_event_subscriptions = Resources.*[ Type == %RDS_EVENTSUBSCRIPTION_TYPE ]

#
# Primary Rules
#
rule rds_sg_event_notifications_configured_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %rds_event_subscriptions not
 empty  {
    check(%rds_event_subscriptions.Properties)
        <<
        [CT.RDS.PR.19]: Require an Amazon RDS event notifications subscription to have
 critical database security group events configured
```

```
        [FIX]: When 'SourceType' is set to 'db-security-group', set 'Enabled' to
 true and ensure that the parameter 'EventCategories' contains both 'failure' and
 'configuration change' values.
        >>
}

rule rds_sg_event_notifications_configured_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_EVENTSUBSCRIPTION_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_EVENTSUBSCRIPTION_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.19]: Require an Amazon RDS event notifications subscription to have
 critical database security group events configured
        [FIX]: When 'SourceType' is set to 'db-security-group', set 'Enabled' to
 true and ensure that the parameter 'EventCategories' contains both 'failure' and
 'configuration change' values.
        >>
}

#
# Parameterized Rules
#
rule check(resource) {
    %resource [ SourceType == %EVENT_SOURCE_TYPE ] {
        Enabled exists
        # Scenario 4
        Enabled == true
        # Scenario 5
        EventCategories not exists or
        # Scenario 6
        check_event_categories_for_required_events(EventCategories)
    }
}

rule check_event_categories_for_required_events(event_categories) {
    %event_categories {
        this exists
        this is_list
        this empty or
        %EVENT_CATEGORIES.* in this
    }
}

#
# Utility Rules
```

```
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.19 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      SourceType: db-security-group
      Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
```

```
    Properties: {}
  RDSEventSubscription:
    Type: AWS::RDS::EventSubscription
    Properties:
      SnsTopicArn:
        Ref: SNSTopic
      EventCategories:
      - failure
      SourceType: db-security-group
      Enabled: true
```

[CT.RDS.PR.20] Require an Amazon RDS database instance not to use a database engine default port

This control checks whether Amazon Relational Database Service (RDS) database instances are configured for default database port for their specific engine types.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.20 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.20 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.20 example templates

Explanation

If you use a known port to deploy an Amazon RDS cluster or instance, an attacker can guess information about the cluster or instance. The attacker can use this information in conjunction with other information to connect to an Amazon RDS cluster or instance, or to gain additional information about your application.

When you change the port, you must also update the existing connection strings that were used to connect to the old port. You also should check the security group of the DB instance to ensure that it includes an ingress rule that allows connectivity on the new port.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-se2`, `oracle-ee-cdb`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`.
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

Set a value for `Port` that is different than the default value for the Amazon RDS DB instance engine type.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example One

Amazon RDS DB instance configured with a port that's different than the `mysql` engine default port. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "mysql",
            "EngineVersion": 5.7,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
```

```
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true,
            "Port": 6733
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: mysql
    EngineVersion: 5.7
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    StorageEncrypted: true
    Port: 6733
  DeletionPolicy: Delete
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example Two

Amazon RDS DB instance configured with a port that's different than the `postgres` engine default port. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
```

```
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "Port": 5723
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    Port: 5723
  DeletionPolicy: Delete
```

CT.RDS.PR.20 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   rds_instance_no_default_ports_check
#
# Description:
#   This control checks whether Amazon Relational Database Service (RDS) database
 instances are configured for default database port for their specific engine types.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of  'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of  'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#       And: 'Port' has not been specified
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of  'mariadb', 'mysql',
#            'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-se2-cdb',
```

```
#            'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex', 'sqlserver-
web'
#       And: 'Port' has been specified
#       And: 'Port' value is default port (includes mysql/mariadb port '3306',
 sqlserver
#            port '1433', postgres port '5432', and oracle port '1521')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine'  is one of 'mariadb', 'mysql'
#       And: 'Port' has been specified
#       And: 'Port' value is not equal to '3306'
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is 'postgres'
#       And: 'Port' has been specified
#       And: 'Port' value is not equal to '5432'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'sqlserver-ee', 'sqlserver-se', 'sqlserver-ex',
 'sqlserver-web'
#       And: 'Port' has been specified
#       And: 'Port' value is not equal to '1433'
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'oracle-ee', 'oracle-se2', 'oracle-ee-cdb', 'oracle-
se2-cdb',
#       And: 'Port' has been specified
#       And: 'Port' value is not equal to '1521'
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let ORACLE_ENGINES = [ "oracle-ee", "oracle-se2", "oracle-se1", "oracle-se" ]
```

```
let SQLSERVER_ENGINES = [ "sqlserver-ee", "sqlserver-se", "sqlserver-ex", "sqlserver-
web" ]
let MYSQL_OR_MARIA_ENGINES = [ "mariadb", "mysql" ]
let POSTGRES_ENGINES = [ "postgres" ]
let MYSQL_MARIA_DEFAULT_PORTS = [3306, "3306"]
let POSTGRES_DEFAULT_PORTS = [5432, "5432"]
let SQL_DEFAULT_PORTS = [1433, "1433"]
let ORACLE_DEFAULT_PORTS = [1521, "1521"]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_no_default_ports_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.20]: Require an Amazon RDS database instance not to use a database
 engine default port
        [FIX]: Set a value for 'Port' that is different than the default value for the
 Amazon RDS DB instance engine type.
        >>
}

rule rds_instance_no_default_ports_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.20]: Require an Amazon RDS database instance not to use a database
 engine default port
        [FIX]: Set a value for 'Port' that is different than the default value for the
 Amazon RDS DB instance engine type.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    # Scenario: 4 and 5
```

```
    %rds_db_instance[ filter_engine(this, %MYSQL_OR_MARIA_ENGINES) ] {
        check_port(Port, %MYSQL_MARIA_DEFAULT_PORTS)
    }
    # Scenario: 4 and 6
    %rds_db_instance[ filter_engine(this, %POSTGRES_ENGINES) ] {
        check_port(Port, %POSTGRES_DEFAULT_PORTS)
    }
    # Scenario: 4 and 7
    %rds_db_instance[ filter_engine(this, %SQLSERVER_ENGINES) ] {
        check_port(Port, %SQL_DEFAULT_PORTS)
    }
    # Scenario: 4 and 8
    %rds_db_instance[ filter_engine(this, %ORACLE_ENGINES) ] {
        check_port(Port, %ORACLE_DEFAULT_PORTS)
    }
}

rule filter_engine(db_properties, engine) {
    %db_properties {
        # Scenario: 2
        Engine exists
        Engine is_string
        Engine in %engine
    }
}

rule check_port(port, default_ports) {
    # Scenario: 3
    %port exists
    %port not in %default_ports
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
```

```
}
```

CT.RDS.PR.20 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      Port: 6733
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      Port: 3306
    DeletionPolicy: Delete
```

[CT.RDS.PR.21] Require an Amazon RDS DB cluster to have a unique administrator username

This control checks whether an Amazon Relational Database Service (RDS) database (DB) cluster has changed the administrator username from its default value.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.21 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.21 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.21 example templates

Explanation

When you create an Amazon RDS database, we recommend that you change the default administrator username to a unique value. Default user names are public knowledge, and they should be changed, because changing these user names reduces the risk of unintended access.

Usage considerations

- This control applies only to Amazon RDS DB clusters that set the `MasterUsername` property.

Remediation for rule failure

Set `MasterUsername` to a value other than `admin` or `postgres`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

Amazon RDS DB cluster configured with a unique administrator username. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora-mysql",
            "MasterUsername": "samplemasteruser",
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RDSClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
```

```
                "Ref": "DBSubnetGroup"
            }
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-mysql
    MasterUsername: samplemasteruser
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RDSClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
```

CT.RDS.PR.21 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_default_admin_check
#
# Description:
#   This control checks whether an Amazon Relational Database Service (RDS) database
 (DB) cluster has changed the administrator username from its default value.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'MasterUsername' has not been provided
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'MasterUsername' has been provided and it is set to 'admin' or 'postgres'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'MasterUsername' has been provided and is not set to 'admin' or 'postgres'
#      Then: PASS

#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let DISALLOWED_MASTER_USERNAMES = ["admin", "postgres"]
let INPUT_DOCUMENT = this

#
# Assignments
#
let db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_default_admin_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %db_clusters not empty {
    check(%db_clusters.Properties)
        <<
        [CT.RDS.PR.21]: Require an Amazon RDS DB cluster to have a unique administrator
 username
            [FIX]: Set 'MasterUsername' to a value other than 'admin' or 'postgres'.
        >>
```

```
}

rule rds_cluster_default_admin_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.21]: Require an Amazon RDS DB cluster to have a unique administrator
 username
            [FIX]: Set 'MasterUsername' to a value other than 'admin' or 'postgres'.
        >>
}

rule check(db_cluster) {
    %db_cluster [
        # scenario 2
        filter_master_username_provided(this)
    ] {
        # scenario 3 and 4
        MasterUsername not in %DISALLOWED_MASTER_USERNAMES
    }
}

#
# Utility Rules
#
rule filter_master_username_provided(dbcluster_properties) {
    %dbcluster_properties{
        MasterUsername exists
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists

}
```

CT.RDS.PR.21 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
```

```
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername: examplemasteruser
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/25
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
```

```
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.128/25
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
      VpcId:
        Ref: VPC
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: DB subnet group for DBCluster
      SubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 32
        ExcludeCharacters: "/@\"'\\"
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      Engine: aurora-mysql
      MasterUsername: admin
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
      DBSubnetGroupName:
        Ref: DBSubnetGroup
```

[CT.RDS.PR.22] Require an Amazon RDS database instance to have a unique administrator username

This control checks whether an Amazon Relational Database Service (RDS) database has changed the adminstrator username from its default value.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.22 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.22 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.22 example templates

Explanation

Default administrative usernames on Amazon RDS databases are public knowledge. When creating an Amazon RDS database, you should change the default administrative username to a unique value, thereby reducing the risk of unintended access.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex`, and `sqlserver-web`.

Remediation for rule failure

Set `MasterUsername` to a value other than `postgres` or `admin`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured with a custom administrator username. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": "testUser",
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            }
        },
        "DeletionPolicy": "Delete"
    }
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: testUser
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
  DeletionPolicy: Delete
```

CT.RDS.PR.22 rule specification

```
# ###################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   rds_instance_default_admin_check
#
# Description:
#   This control checks whether an Amazon Relational Database Service (RDS) database
 has changed the adminstrator username from its default value.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'MasterUsername' has been specified and is one of 'postgres' or 'admin'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
```

```
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#       And: 'MasterUsername' has been specified and is not one of 'postgres' or
 'admin'
#      Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]
let RDS_DEFAULT_USERNAMES = [ "postgres", "admin" ]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_default_admin_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.22]: Require an Amazon RDS database instance to have a unique
 administrator username
        [FIX]: Set 'MasterUsername' to a value other than 'postgres' or 'admin'.
        >>
}

rule rds_instance_default_admin_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.22]: Require an Amazon RDS database instance to have a unique
 administrator username
```

```
        [FIX]: Set 'MasterUsername' to a value other than 'postgres' or 'admin'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [ filter_engine_and_master_username_provided(this) ] {
        # Scenario: 3 and 4
        MasterUsername not in %RDS_DEFAULT_USERNAMES
    }
}

rule filter_engine_and_master_username_provided(db_properties) {
    %db_properties {
        # Scenario: 2
        MasterUsername exists
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.22 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername: testUser
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Test RDS DB Instance secret
      GenerateSecretString:
```

```
        SecretStringTemplate: '{"username": "testUser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername: postgres
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DeletionPolicy: Delete
```

[CT.RDS.PR.23] Require an Amazon RDS database instance to not be publicly accessible

This rule checks whether Amazon Relational Database Service (RDS) database (DB) instances are publicly accessible, as determined by checking the `PubliclyAccessible` configuration property.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.23 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.23 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.23 example templates

Explanation

The `PubliclyAccessible` property in the RDS instance CloudFormation resource indicates whether the DB instance is publicly accessible. When the DB instance is configured with `PubliclyAccessible` set to `true`, it is an internet-facing instance with a publicly resolvable DNS name, which resolves to a public IP address. When the DB instance isn't publicly accessible, it is an internal instance with a DNS name that resolves to a private IP address.

Unless you intend for your RDS instance to be publicly accessible, do not configure the RDS instance with the `PubliclyAccessible` value set to `true`, because this configuration may allow unwanted traffic to your database instance.

Remediation for rule failure

Set the value of `PubliclyAccessible` to `false`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Instance - Example

Amazon RDS DB instance configured as an internal instance, by mmeans of a publicly accessible configuration. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "PubliclyAccessible": false
        },
        "DeletionPolicy": "Delete"
    }
```

```
}
```

YAML example

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    PubliclyAccessible: false
  DeletionPolicy: Delete
```

CT.RDS.PR.23 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_public_access_check
#
# Description:
#   This rule checks whether Amazon Relational Database Service (RDS) database (DB)
 instances are publicly accessible, as determined by checking the 'PubliclyAccessible'
 configuration property.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document does not contain any RDS DB instance resources
#       Then: SKIP
#   Scenario: 2
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB instance resource
#        And: 'PubliclyAccessible' has not been specified
#       Then: FAIL
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB instance resource
#        And: 'PubliclyAccessible' is present and is a value other than bool(false)
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB instance resource
#        And: 'PubliclyAccessible' has been specified and set to bool(false)
#       Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_instance_public_access_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.23]: Require an Amazon RDS database instance to not be publicly
 accessible
            [FIX]: Set the value of 'PubliclyAccessible' to 'false'.
```

```
        >>
}

rule rds_instance_public_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.23]: Require an Amazon RDS database instance to not be publicly
 accessible
            [FIX]: Set the value of 'PubliclyAccessible' to 'false'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance{
       #Scenario: 2
       PubliclyAccessible exists
       #Scenario: 3 and 4
       PubliclyAccessible == false

    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.23 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      PubliclyAccessible: false
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
```

```
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      PubliclyAccessible: true
    DeletionPolicy: Delete
```

[CT.RDS.PR.24] Require an Amazon RDS database instance to have encryption at rest configured

This control checks whether storage encryption is enabled for your Amazon RDS database (DB) instance.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.24 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.24 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.24 example templates

Explanation

For an added layer of security for your sensitive data in Amazon RDS DB instances, you should configure your RDS DB instances to be encrypted at rest. To encrypt your RDS DB instances and snapshots at rest, enable the encryption option for your RDS DB instances. Data that is encrypted at rest includes the underlying storage for DB instances, its automated backups, read replicas, and snapshots.

Encrypted Amazon RDS DB instances use the open standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You do not need to modify your database client applications to use encryption.

Amazon RDS encryption currently is available for all database engines and storage types. Amazon RDS encryption is available for most DB instance classes.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, `sqlserver-ex` and `sqlserver-web`
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.

Remediation for rule failure

The parameter `StorageEncrypted` must be set to true for RDS DB Instances.

The examples that follow show how to implement this remediation.

Amazon RDS DB instance - Example

Amazon RDS DB instance with storage encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RDSDBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RDSDBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true
        }
    }
}
```

YAML example

```
RDSDBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
    MasterUsername: !Sub '{{resolve:secretsmanager:${RDSDBInstanceSecret}::username}}'
```

```
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${RDSDBInstanceSecret}::password}}'
    StorageEncrypted: true
```

CT.RDS.PR.24 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_instance_storage_encrypted_check
#
# Description:
#   Checks whether storage encryption is enabled for your Amazon RDS DB instances.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#            'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#      Then: SKIP
#   Scenario: 3
#      Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#        And: The input document contains an RDS DB instance resource
#        And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#        And: 'StorageEncrypted' has not been provided
#       Then: FAIL
#   Scenario: 4
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB instance resource
#        And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#        And: 'StorageEncrypted' has been provided and set to bool(false)
#       Then: FAIL
#   Scenario: 5
#      Given: The input document is an CloudFormation or CloudFormation hook document
#        And: The input document contains an RDS DB instance resource
#        And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
 'oracle-se2',
#             'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-se', 'sqlserver-
ex', 'sqlserver-web'
#        And: 'StorageEncrypted' has been provided and set to bool(true)
#       Then: PASS

#
# Constants
#
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let INPUT_DOCUMENT = this
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-ex", "sqlserver-web"
]

#
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
```

```
#
rule rds_instance_storage_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.24]: Require an Amazon RDS database instance to have encryption at
 rest configured
        [FIX]: The parameter 'StorageEncrypted' must be set to true for RDS DB
 Instances.
        >>
}

rule rds_instance_storage_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.24]: Require an Amazon RDS database instance to have encryption at
 rest configured
        [FIX]: The parameter 'StorageEncrypted' must be set to true for RDS DB
 Instances.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [filter_restore_and_engine(this)] {
        #Scenario: 3
        StorageEncrypted exists
        #Scenario: 4 and 5
        StorageEncrypted == true
    }
}

rule filter_restore_and_engine(db_properties) {
    %db_properties {
        #Scenario: 2
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.24 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: "/@\""
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
```

```
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: true
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: "/@\""
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 14.2
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: false
    DeletionPolicy: Delete
```

[CT.RDS.PR.25] Require an Amazon RDS database cluster to export logs to Amazon CloudWatch Logs by means of the EnableCloudwatchLogsExports property

This control checks whether Amazon RDS database clusters have all available log types enabled for export to Amazon CloudWatch Logs.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.25 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.25 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.25 example templates

Explanation

AWS Control Tower recommends that you enable the export of relevant logs for all Amazon RDS database clusters to Amazon CloudWatch Logs. Database logging provides detailed records of requests made to RDS. Database logs can assist with security and access audits, and they can help you diagnose availability issues.

Usage considerations

- This control applies only to Amazon RDS DB cluster engine types `aurora`, `aurora-mysql`, `aurora-postgresql`, `mysql` and `postgres`.
- Additional prerequisites may exist for enabling logging based on your selected database engine type. Refer to Monitoring Amazon Aurora log files in the *Amazon Aurora User Guide* for more information.

Remediation for rule failure

Specify `EnableCloudwatchLogsExports` with a list of all supported log types for the Amazon RDS database cluster engine.

The examples that follow show how to implement this remediation.

Amazon RDS database (DB) Cluster - Example One

Amazon RDS Aurora DB cluster configured with all available log types enabled for export to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "EnableCloudwatchLogsExports": [
                "audit",
                "error",
                "general",
                "slowquery"
            ]
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    EnableCloudwatchLogsExports:
      - audit
      - error
      - general
      - slowquery
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example Two

Amazon RDS Multi-AZ Postgres DB cluster configured with all available log types enabled for export to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "aurora",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "EnableCloudwatchLogsExports": [
                "audit",
                "error",
                "general",
                "slowquery"
```

```
            ]
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    EnableCloudwatchLogsExports:
      - audit
      - error
      - general
      - slowquery
```

CT.RDS.PR.25 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_logging_enabled_check
#
# Description:
#   This control checks whether Amazon RDS database clusters have all available log
 types enabled for export to Amazon CloudWatch Logs.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
```

```
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any RDS DB cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster Resource
#       And: 'Engine' is not one of 'aurora', 'aurora-mysql', 'aurora-postgresql',
 'mysql' or 'postgres'
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql', 'mysql'
 or 'postgres'
#       And: 'EnableCloudwatchLogsExports' has not been specified or has been specified
 as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql', 'mysql'
 or 'postgres'
#       And: 'EnableCloudwatchLogsExports' has been specified and is a non-empty list
#       And: One or more log types in 'EnableCloudwatchLogsExports' are not supported
 by the specified 'Engine'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' is one of 'aurora', 'aurora-mysql', 'aurora-postgresql', 'mysql'
 or 'postgres'
#       And: 'EnableCloudwatchLogsExports' has been specified and is a non-empty list
#       And: 'EnableCloudwatchLogsExports' does not contain all log types supported by
 the specified 'Engine'
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
```

```
#       And: 'Engine' is one of 'aurora', 'aurora-mysql' or 'mysql'
#       And: 'EnableCloudwatchLogsExports' has been specified as a list with all
 supported log types
#             for the 'Engine' ('audit', 'error', 'general' and 'slowquery')
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' is 'aurora-postgresql'
#       And: 'EnableCloudwatchLogsExports' has been specified as a list with all
 supported log types
#             for the 'Engine' ('postgresql')
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an RDS DB cluster resource
#       And: 'Engine' is 'postgres'
#       And: 'EnableCloudwatchLogsExports' has been specified as a list with all
 supported log types
#             for the 'Engine' ('postgresql', 'upgrade')
#      Then: PASS

#
# Constants
#
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let INPUT_DOCUMENT = this

let SUPPORTED_RDS_CLUSTER_ENGINES = [
    "aurora", "aurora-mysql", "aurora-postgresql", "mysql", "postgres"
]

let MYSQL_ENGINE_SUBTYPES = [ "aurora", "aurora-mysql", "mysql" ]
let AURORA_POSTGRES_ENGINE_SUBTYPES = [ "aurora-postgresql" ]
let POSTGRES_ENGINE_SUBTYPES = [ "postgres" ]

let MYSQL_SUPPORTED_LOG_TYPES = [ "audit", "error", "general", "slowquery" ]
let AURORA_POSTGRES_SUPPORTED_LOG_TYPES  = [ "postgresql" ]
let POSTGRES_SUPPORTED_LOG_TYPES = [ "postgresql", "upgrade" ]

#
# Assignments
#
let rds_db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]
```

```
#
# Primary Rules
#
rule rds_cluster_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %rds_db_clusters not empty {
    check(%rds_db_clusters.Properties)
        <<
        [CT.RDS.PR.25]: Require an Amazon RDS database cluster to have logging
 configured
            [FIX]: Specify 'EnableCloudwatchLogsExports' with a list of all supported
 log types for the Amazon RDS database cluster engine.
        >>
}

rule rds_cluster_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.25]: Require an Amazon RDS database cluster to have logging
 configured
            [FIX]: Specify 'EnableCloudwatchLogsExports' with a list of all supported
 log types for the Amazon RDS database cluster engine.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_cluster) {
    %rds_db_cluster [
        filter_engine(this)
    ] {
        # Scenario 3
        EnableCloudwatchLogsExports exists
        check_is_list_and_not_empty(EnableCloudwatchLogsExports)

        # Scenario 4 and 6
        when Engine IN %MYSQL_ENGINE_SUBTYPES {
            %MYSQL_SUPPORTED_LOG_TYPES.* IN EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %MYSQL_SUPPORTED_LOG_TYPES[*]
        }

        # Scenario 4 and 7
```

```
        when Engine IN %AURORA_POSTGRES_ENGINE_SUBTYPES {
            %AURORA_POSTGRES_SUPPORTED_LOG_TYPES.* IN EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %AURORA_POSTGRES_SUPPORTED_LOG_TYPES[*]
        }

        # Scenario 4 and 8
        when Engine IN %POSTGRES_ENGINE_SUBTYPES {
            %POSTGRES_SUPPORTED_LOG_TYPES.* in EnableCloudwatchLogsExports[*]
            EnableCloudwatchLogsExports.* IN %POSTGRES_SUPPORTED_LOG_TYPES[*]
        }
    }
}

rule filter_engine(db_properties) {
    %db_properties {
        # Scenario 2
        Engine exists
        Engine in %SUPPORTED_RDS_CLUSTER_ENGINES
    }
}

#
# Utility Rules
#
rule check_is_list_and_not_empty(value) {
    %value {
        this is_list
        this not empty
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.25 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      MasterUsername: exampleusername
      MasterUserPassword: example-password
      DBSubnetGroupName: example-db-subnet-group
      Engine: aurora
      EnableCloudwatchLogsExports:
        - audit
        - error
        - general
        - slowquery
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      DBClusterInstanceClass: db.m6gd.large
      MasterUsername: exampleusername
      MasterUserPassword: example-password
      DBSubnetGroupName: example-db-subnet-group
      Engine: postgres
      AllocatedStorage: 100
      StorageType: io1
      Iops: 3000
      EnableCloudwatchLogsExports:
        - postgresql
        - upgrade
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      MasterUsername: exampleusername
      MasterUserPassword: example-password
      DBSubnetGroupName: example-db-subnet-group
      Engine: aurora
```

[CT.RDS.PR.26] Require an Amazon Relational Database Service DB Proxy to require Transport Layer Security (TLS) connections

This control checks whether an Amazon Relational Database Service DB Proxy is configured to require Transport Layer Security (TLS) for connections to the proxy.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBProxy`
- **CloudFormation guard rule:** CT.RDS.PR.26 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.26 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.26 example templates

Explanation

Amazon RDS Proxy can act as an additional layer of security between client applications and the underlying database. For example, you can connect to the proxy using TLS 1.2, even if the underlying DB instance supports an older version of TLS. You can connect to the proxy using an IAM role, even if the proxy connects to the database with the native user and password authentication method. With this technique, you can enforce strong authentication requirements for database applications without a costly migration effort for the DB instances themselves.

Usage considerations

- For general information about Amazon RDS proxy limitations, see Quotas and limitations for RDS Proxy in the *Amazon Relational Database Service User Guide*.

Remediation for rule failure

Set the value of the RequireTLS property to true.

The examples that follow show how to implement this remediation.

Amazon RDS DB Proxy - Example

An Amazon RDS DB Proxy configured to require TLS connections to the proxy. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBProxy": {
        "Type": "AWS::RDS::DBProxy",
        "Properties": {
            "DBProxyName": "sample-db-proxy",
            "EngineFamily": "MYSQL",
            "IdleClientTimeout": 120,
            "RoleArn": {
                "Fn::GetAtt": "ProxySecretAccessRole.Arn"
            },
            "Auth": [
                {
                    "AuthScheme": "SECRETS",
                    "SecretArn": {
                        "Ref": "DBInstanceSecret"
```

```
                    },
                    "IAMAuth": "DISABLED"
                }
            ],
            "VpcSubnetIds": [
                {
                    "Ref": "SubnetOne"
                },
                {
                    "Ref": "SubnetTwo"
                }
            ],
            "RequireTLS": true
        }
    }
}
```

YAML example

```
DBProxy:
  Type: AWS::RDS::DBProxy
  Properties:
    DBProxyName: sample-db-proxy
    EngineFamily: MYSQL
    IdleClientTimeout: 120
    RoleArn: !GetAtt 'ProxySecretAccessRole.Arn'
    Auth:
      - AuthScheme: SECRETS
        SecretArn: !Ref 'DBInstanceSecret'
        IAMAuth: DISABLED
    VpcSubnetIds:
      - !Ref 'SubnetOne'
      - !Ref 'SubnetTwo'
    RequireTLS: true
```

CT.RDS.PR.26 rule specification

```
# ####################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   rds_proxy_tls_check
#
# Description:
#   This control checks whether an Amazon RDS DB Proxy is configured to require
 Transport Layer Security (TLS) for connections to the proxy.
#
# Reports on:
#   AWS::RDS::DBProxy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS DB proxy resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB proxy resource
#       And: 'RequireTLS' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB proxy resource
#       And: 'RequireTLS' has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB proxy resource
#       And: 'RequireTLS' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
```

```
let RDS_DB_PROXY_TYPE = "AWS::RDS::DBProxy"

#
# Assignments
#
let rds_db_proxies = Resources.*[ Type == %RDS_DB_PROXY_TYPE ]

#
# Primary Rules
#
rule rds_proxy_tls_check when is_cfn_template(%INPUT_DOCUMENT)
                              %rds_db_proxies not empty {
    check(%rds_db_proxies.Properties)
        <<
        [CT.RDS.PR.26]: Require an Amazon RDS DB Proxy to require Transport Layer
 Security (TLS) connections
        [FIX]: Set the value of the RequireTLS property to true.
        >>
}

rule rds_proxy_tls_check when is_cfn_hook(%INPUT_DOCUMENT, %RDS_DB_PROXY_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_PROXY_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.26]: Require an Amazon RDS DB Proxy to require Transport Layer
 Security (TLS) connections
        [FIX]: Set the value of the RequireTLS property to true.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_proxy) {
    %rds_db_proxy {
        # Scenarios 2
        RequireTLS exists
        # Scenarios 3 and 4
        RequireTLS == true
    }
}

#
# Utility Rules
#
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.26 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
```

```
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  ProxySecretAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: rds.amazonaws.com
          Action: sts:AssumeRole
      Policies:
      - PolicyName: SecretAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - secretsmanager:GetSecretValue
            Resource:
              Ref: DBInstanceSecret
          - Effect: Allow
            Action:
            - kms:Decrypt
            Resource:
              Fn::Sub: arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/*
            Condition:
              StringEquals:
                kms:ViaService:
                  Fn::Sub: secretsmanager.${AWS::Region}.amazonaws.com
              ForAnyValue:StringEquals:
                kms:ResourceAliases: alias/aws/secretsmanager
```

```
  DBProxy:
    Type: AWS::RDS::DBProxy
    Properties:
      DBProxyName:
        Fn::Sub: ${AWS::StackName}-example
      EngineFamily: MYSQL
      IdleClientTimeout: 120
      RoleArn:
        Fn::GetAtt: ProxySecretAccessRole.Arn
      Auth:
      - AuthScheme: SECRETS
        SecretArn:
          Ref: DBInstanceSecret
        IAMAuth: DISABLED
      VpcSubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      RequireTLS: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  SubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SubnetTwo:
```

```
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: ''
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: '"@/\'
  ProxySecretAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service: rds.amazonaws.com
          Action: sts:AssumeRole
      Policies:
      - PolicyName: SecretAccessPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - secretsmanager:GetSecretValue
            Resource:
              Ref: DBInstanceSecret
          - Effect: Allow
            Action:
            - kms:Decrypt
            Resource:
              Fn::Sub: arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/*
            Condition:
```

```
            StringEquals:
              kms:ViaService:
                Fn::Sub: secretsmanager.${AWS::Region}.amazonaws.com
            ForAnyValue:StringEquals:
              kms:ResourceAliases: alias/aws/secretsmanager
  DBProxy:
    Type: AWS::RDS::DBProxy
    Properties:
      DBProxyName:
        Fn::Sub: ${AWS::StackName}-example
      EngineFamily: MYSQL
      IdleClientTimeout: 120
      RoleArn:
        Fn::GetAtt: ProxySecretAccessRole.Arn
      Auth:
      - AuthScheme: SECRETS
        SecretArn:
          Ref: DBInstanceSecret
        IAMAuth: DISABLED
      VpcSubnetIds:
      - Ref: SubnetOne
      - Ref: SubnetTwo
      RequireTLS: false
```

[CT.RDS.PR.27] Require an Amazon Relational Database Service DB cluster parameter group to require Transport Layer Security (TLS) connections for supported engine types

This control checks whether an Amazon Relational Database Service DB cluster parameter group requires Transport Layer Security (TLS) connections for supported engine types.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBClusterParameterGroup`
- **CloudFormation guard rule:** CT.RDS.PR.27 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.27 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.27 example templates

Explanation

You can use Secure Socket Layer (SSL) or Transport Layer Security (TLS) from your application to encrypt a connection to a DB cluster running Aurora MySQL, Aurora PostgreSQL, MySQL, or PostgreSQL. SSL/TLS connections provide a layer of security by encrypting data that moves between your client and DB cluster.

Usage considerations

- This control applies only to Amazon RDS DB cluster parameter groups with families `aurora-mysql`, `aurora-postgresql`, `postgres`, or `mysql`.

Remediation for rule failure

For Amazon RDS DB cluster parameter groups with `aurora-mysql` and `mysql` families, in the Parameters property, set the value of `require_secure_transport` to true. For Amazon RDS DB cluster parameter groups with `aurora-postgresql` amd `postgres` families, in the Parameters property, set the value of `rds.force_ssl` to true.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster Parameter Group - Example One

An Amazon RDS DB cluster parameter group configured to require TLS/SSL for all connections to Aurora MySQL DB clusters. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBClusterParameterGroup": {
        "Type": "AWS::RDS::DBClusterParameterGroup",
        "Properties": {
            "Description": "sample-db-parameter-group",
```

```
            "Family": "aurora-mysql5.7",
            "Parameters": {
                "require_secure_transport": "ON"
            }
        }
    }
}
```

YAML example

```
RDSDBClusterParameterGroup:
  Type: AWS::RDS::DBClusterParameterGroup
  Properties:
    Description: sample-db-parameter-group
    Family: aurora-mysql5.7
    Parameters:
      require_secure_transport: 'ON'
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster Parameter Group - Example Two

An Amazon RDS DB cluster parameter group configured to require TLS/SSL for all connections to PostgreSQL DB clusters. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBClusterParameterGroup": {
        "Type": "AWS::RDS::DBClusterParameterGroup",
        "Properties": {
            "Description": "sample-db-parameter-group",
            "Family": "postgres14",
            "Parameters": {
                "rds.force_ssl": true
            }
        }
    }
```

```
}
```

YAML example

```
RDSDBClusterParameterGroup:
  Type: AWS::RDS::DBClusterParameterGroup
  Properties:
    Description: sample-db-parameter-group
    Family: postgres14
    Parameters:
      rds.force_ssl: true
```

CT.RDS.PR.27 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_db_cluster_parameter_group_tls_check
#
# Description:
#   This control checks whether an Amazon RDS DB cluster parameter group requires
 Transport Layer Security (TLS) connections for supported engine types.
#
# Reports on:
#   AWS::RDS::DBClusterParameterGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any Amazon RDS DB cluster parameter
 group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has not been provided or has been provided and set to an RDS DB
 cluster
#            parameter group family other than 'aurora-mysql', 'aurora-postgresql',
 'postgres',
#            or 'mysql' families
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to an 'aurora-mysql' Amazon RDS DB
 cluster parameter
#            group family
#       And: In 'Parameters', 'require_secure_transport' has not been provided, or
#            has been provided and set to a value other than 'ON'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to a 'mysql' Amazon RDS DB cluster
 parameter
#            group family
#       And: In 'Parameters', 'require_secure_transport' has not been provided, or
#            has been provided and set to a value other than a boolean true value
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to an 'aurora-postgresql' or 'postgres'
#            Amazon RDS DB cluster parameter group family
#       And: In 'Parameters', 'rds.force_ssl' has not been provided, or has been
 provided
#            and set to a value other than a boolean true value
#      Then: FAIL
#   Scenario: 6
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to an 'aurora-mysql' RDS DB cluster
 parameter
#            group family
#       And: In 'Parameters', 'require_secure_transport' has been provided and set
#            to 'ON'
#      Then: PASS
#   Scenario: 7
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to a 'mysql' Amazon RDS DB cluster
 parameter group family
#       And: In 'Parameters', 'require_secure_transport' has been provided and set
#            to a boolean true value
#      Then: PASS
#   Scenario: 8
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster parameter group
 resource
#       And: 'Family' has been provided and set to an 'aurora-postgresql' or 'postgres'
 RDS DB
#            cluster parameter group family
#       And: In 'Parameters', 'rds.force_ssl' has been provided and set
#            to a boolean true value
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let RDS_DB_CLUSTER_PARAMETER_GROUP_TYPE = "AWS::RDS::DBClusterParameterGroup"

let AURORA_MYSQL_PG_FAMILY = /^aurora-mysql/
let AURORA_POSTGRES_PG_FAMILY = /^aurora-postgresql/
let MYSQL_PG_FAMILY = /^mysql/
let POSTGRES_PG_FAMILY = /^postgres/

let BOOLEAN_TRUE_VALUES = [
    true,
    1, "1",
    "true", "True", "TRUE",
```

```
    "on", "On", "ON"
]
let AURORA_MSQL_ON_PATTERN = /(?i)^on$/

#
# Assignments
#
let rds_db_cluster_parameter_groups = Resources.*[ Type ==
 %RDS_DB_CLUSTER_PARAMETER_GROUP_TYPE ]

#
# Primary Rules
#
rule rds_db_cluster_parameter_group_tls_check when is_cfn_template(%INPUT_DOCUMENT)
                                                  %rds_db_cluster_parameter_groups not
 empty {
    check(%rds_db_cluster_parameter_groups.Properties)
        <<
        [CT.RDS.PR.27]: Require an Amazon RDS DB cluster parameter group to require
 Transport Layer Security (TLS) connections for supported engine types
        [FIX]: For RDS DB cluster parameter groups with 'aurora-mysql' and 'mysql'
 families, in the Parameters property, set the value of 'require_secure_transport'
 to true. For RDS DB cluster parameter groups with 'aurora-postgresql' amd 'postgres'
 families, in  the Parameters property, set the value of 'rds.force_ssl' to true.
        >>
}

rule rds_db_cluster_parameter_group_tls_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_PARAMETER_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_PARAMETER_GROUP_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.27]: Require an Amazon RDS DB cluster parameter group to require
 Transport Layer Security (TLS) connections for supported engine types
        [FIX]: For RDS DB cluster parameter groups with 'aurora-mysql' and 'mysql'
 families, in the Parameters property, set the value of 'require_secure_transport'
 to true. For Amazon RDS DB cluster parameter groups with 'aurora-postgresql' amd
 'postgres' families, in  the Parameters property, set the value of 'rds.force_ssl' to
 true.
        >>
}

#
# Parameterized Rules
#
```

```
rule check(rds_parameter_group) {
    %rds_parameter_group [
        # Scenario 2
        filter_pg_aurora_mysql_families(this)
    ] {
        # Scenarios 3 and 5
        Parameters exists
        Parameters is_struct
        Parameters {
            require_secure_transport exists
            require_secure_transport in %AURORA_MSQL_ON_PATTERN
        }
    }

    %rds_parameter_group [
        # Scenario 2
        filter_pg_mysql_families(this)
    ] {
        # Scenarios 3 and 5
        Parameters exists
        Parameters is_struct
        Parameters {
            require_secure_transport exists
            require_secure_transport in %BOOLEAN_TRUE_VALUES
        }
    }

    %rds_parameter_group [
        # Scenario 2
        filter_pg_postgres_families(this)
    ] {
        # Scenarios 4 and 6
        Parameters exists
        Parameters is_struct
        Parameters {
            "rds.force_ssl" exists
            "rds.force_ssl" in %BOOLEAN_TRUE_VALUES
        }
    }
}

rule filter_pg_aurora_mysql_families(parameter_group) {
    %parameter_group {
        Family exists
```

```
        Family in %AURORA_MYSQL_PG_FAMILY
    }
}

rule filter_pg_mysql_families(parameter_group) {
    %parameter_group {
        Family exists
        Family in %MYSQL_PG_FAMILY
    }
}

rule filter_pg_postgres_families(parameter_group) {
    %parameter_group {
        Family exists
        Family in %AURORA_POSTGRES_PG_FAMILY or
        Family in %POSTGRES_PG_FAMILY
    }
}

#
# Utility Rules
#
rule check_is_list_and_not_empty(value) {
    %value {
        this is_list
        this not empty
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.27 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RDSDBClusterParameterGroup:
    Type: AWS::RDS::DBClusterParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: aurora-mysql5.7
      Parameters:
        require_secure_transport: 'ON'
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RDSDBClusterParameterGroup:
    Type: AWS::RDS::DBClusterParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: postgres14
      Parameters:
        rds.force_ssl: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RDSDBClusterParameterGroup:
```

```
    Type: AWS::RDS::DBClusterParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: aurora-mysql5.7
      Parameters:
        require_secure_transport: 'OFF'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RDSDBClusterParameterGroup:
    Type: AWS::RDS::DBClusterParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: mysql8.0
      Parameters:
        require_secure_transport: 0
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RDSDBClusterParameterGroup:
    Type: AWS::RDS::DBClusterParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: postgres15
      Parameters:
        rds.force_ssl: false
```

[CT.RDS.PR.28] Require an Amazon Relational Database Service DB parameter group to require Transport Layer Security (TLS) connections for supported engine types

This control checks whether an Amazon Relational Database Service DB parameter group requires Transport Layer Security (TLS) connections, for supported engine types.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBParameterGroup`
- **CloudFormation guard rule:** CT.RDS.PR.28 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.28 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.28 example templates

Explanation

You can use Secure Socket Layer (SSL) or Transport Layer Security (TLS) from your application to encrypt a connection to a DB instance running MariaDB, Microsoft SQL Server, MySQL, Oracle, or PostgreSQL. SSL/TLS connections provide a layer of security by encrypting data that moves between your client and DB instance.

Usage considerations

- This control applies only to Amazon RDS DB parameter groups with families `postgres`, `sqlserver`, `mariadb` (excluding `mariadb10.0` to `mariadb10.4`), and `mysql` (excluding `mysql5.5` to `mysql5.6`)

Remediation for rule failure

For Amazon RDS DB instance parameter groups with `mysql` and `mariadb` families, in `Parameters`, set `require_secure_transport` to `true`. For Amazon RDS DB instance parameter groups with `postgres` and `sqlserver` families, in `Parameters`, set `rds.force_ssl` to `true`.

The examples that follow show how to implement this remediation.

Amazon RDS DB Parameter Group - Example One

An Amazon RDS DB parameter group configured to require TLS/SSL for all connections to MariaDB DB instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBParameterGroup": {
        "Type": "AWS::RDS::DBParameterGroup",
        "Properties": {
            "Description": "sample-db-parameter-group",
            "Family": "mariadb10.6",
            "Parameters": {
                "require_secure_transport": true
            }
        }
    }
}
```

YAML example

```
RDSDBParameterGroup:
  Type: AWS::RDS::DBParameterGroup
  Properties:
    Description: sample-db-parameter-group
    Family: mariadb10.6
    Parameters:
      require_secure_transport: true
```

The examples that follow show how to implement this remediation.

Amazon RDS DB Parameter Group - Example Two

An Amazon RDS DB parameter group configured to require TLS/SSL for all connections to PostgreSQL DB instances. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBParameterGroup": {
        "Type": "AWS::RDS::DBParameterGroup",
        "Properties": {
            "Description": "sample-db-parameter-group",
            "Family": "postgres14",
            "Parameters": {
                "rds.force_ssl": true
            }
        }
    }
}
```

YAML example

```
RDSDBParameterGroup:
  Type: AWS::RDS::DBParameterGroup
  Properties:
    Description: sample-db-parameter-group
    Family: postgres14
    Parameters:
      rds.force_ssl: true
```

CT.RDS.PR.28 rule specification

```
# ###################################
```

```
##       Rule Specification        ##
#####################################
#
# Rule Identifier:
#   rds_db_parameter_group_tls_check
#
# Description:
#   This control checks whether an Amazon RDS DB parameter group requires Transport
 Layer Security (TLS) connections, for supported engine types.
#
# Reports on:
#   AWS::RDS::DBParameterGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS DB parameter group
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB parameter group resource
#       And: 'Family' has not been provided or has been provided and set to an Amazon
 RDS DB
#            parameter group family other than one with support for requiring TLS
 connections
#            ('mariadb' - excluding mariadb families 10.0 to 10.4, 'mysql' - excluding
 mysql
#            families 5.5 to 5.6, 'postgres' or 'sqlserver')
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB parameter group resource
#       And: 'Family' has been provided and set to Amazon RDS parameter group families
 'mariadb'
#            (excluding families 10.0 to 10.4) or 'mysql' (excluding families 5.5 to
 5.6)
#       And: In 'Parameters', 'require_secure_transport' has not been provided, or
```

```
#            has been provided and set to a value other than a boolean true value
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB parameter group resource
#       And: 'Family' has been provided and set to Amazon RDS parameter group families
 'sqlserver' or 'postgres'
#       And: In 'Parameters', 'rds.force_ssl' has not been provided, or has been
 provided
#            and set to a value other than a boolean true value
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB parameter group resource
#       And: 'Family' has been provided and set to Amazon RDS parameter group families
 'mariadb'
#            (excluding families 10.0 to 10.4) or 'mysql' (excluding families 5.5 to
 5.6)
#       And: In 'Parameters', 'require_secure_transport' has been provided and set to
#            a boolean true value
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB parameter group resource
#       And: 'Family' has been provided and set to Amazon RDS parameter group families
 'sqlserver' or 'postgres'
#       And: In 'Parameters', 'rds.force_ssl' has been provided and set to
#            a boolean true value
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let RDS_DB_PARAMETER_GROUP_TYPE = "AWS::RDS::DBParameterGroup"

let MYSQL_PG_FAMILY = /^mysql/
let MARIADB_PG_FAMILY = /^mariadb/
let POSTGRES_PG_FAMILY = /^postgres/
let SQLSERVER_PG_FAMILY = /^sqlserver/

let MYSQL_FAMILIES_WITH_NO_SECURE_TRANSPORT_SUPPORT = [
    "mysql5.5",
    "mysql5.6"
```

```
]
let MARIADB_FAMILIES_WITH_NO_SECURE_TRANSPORT_SUPPORT = [
    "mariadb10.0",
    "mariadb10.1",
    "mariadb10.2",
    "mariadb10.3",
    "mariadb10.4"
]
let BOOLEAN_TRUE_VALUES = [
    true,
    1, "1",
    "true", "True", "TRUE",
    "on", "On", "ON"
]

#
# Assignments
#
let rds_db_parameter_groups = Resources.*[ Type == %RDS_DB_PARAMETER_GROUP_TYPE ]

#
# Primary Rules
#
rule rds_db_parameter_group_tls_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %rds_db_parameter_groups not empty {
    check(%rds_db_parameter_groups.Properties)
        <<
        [CT.RDS.PR.28]: Require an Amazon RDS DB parameter group to require Transport
 Layer Security (TLS) connections for supported engine types
        [FIX]: For Amazon RDS DB instance parameter groups with 'mysql' and 'mariadb'
 families, in 'Parameters', set 'require_secure_transport' to 'true'. For Amazon RDS DB
 instance parameter groups with 'postgres' and 'sqlserver' families, in 'Parameters',
 set 'rds.force_ssl' to 'true'.
        >>
}

rule rds_db_parameter_group_tls_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_PARAMETER_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_PARAMETER_GROUP_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.28]: Require an Amazon RDS DB parameter group to require Transport
 Layer Security (TLS) connections for supported engine types
        [FIX]: For Amazon RDS DB instance parameter groups with 'mysql' and 'mariadb'
 families, in 'Parameters', set 'require_secure_transport' to 'true'. For Amazon RDS DB
```

```
 instance parameter groups with 'postgres' and 'sqlserver' families, in 'Parameters',
 set 'rds.force_ssl' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(rds_parameter_group) {
    %rds_parameter_group [
        # Scenario 2
        filter_pg_mysql_maria_families(this)
    ] {
        # Scenarios 3 and 5
        Parameters exists
        Parameters is_struct
        Parameters {
            require_secure_transport exists
            require_secure_transport in %BOOLEAN_TRUE_VALUES
        }
    }

    %rds_parameter_group [
        # Scenario 2
        filter_pg_postgres_sqlserver_families(this)
    ] {
        # Scenarios 4 and 6
        Parameters exists
        Parameters is_struct
        Parameters {
            "rds.force_ssl" exists
            "rds.force_ssl" in %BOOLEAN_TRUE_VALUES
        }
    }
}

rule filter_pg_mysql_maria_families(parameter_group) {
    %parameter_group {
        Family exists
        filter_mysql_family(this) or
        filter_mariadb_family(this)
    }
}
```

```
rule filter_mysql_family(parameter_group) {
    %parameter_group {
        Family in %MYSQL_PG_FAMILY
        Family not in %MYSQL_FAMILIES_WITH_NO_SECURE_TRANSPORT_SUPPORT
    }
}

rule filter_mariadb_family(parameter_group) {
    %parameter_group {
        Family in %MARIADB_PG_FAMILY
        Family not in %MARIADB_FAMILIES_WITH_NO_SECURE_TRANSPORT_SUPPORT
    }
}

rule filter_pg_postgres_sqlserver_families(parameter_group) {
    %parameter_group {
        Family exists
        Family in %POSTGRES_PG_FAMILY or
        Family in %SQLSERVER_PG_FAMILY
    }
}

#
# Utility Rules
#
rule check_is_list_and_not_empty(value) {
    %value {
        this is_list
        this not empty
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.28 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RDSDBParameterGroup:
    Type: AWS::RDS::DBParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: mariadb10.6
      Parameters:
        require_secure_transport: true
```

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RDSDBParameterGroup:
    Type: AWS::RDS::DBParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: postgres14
      Parameters:
        rds.force_ssl: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
```

```
  RDSDBParameterGroup:
    Type: AWS::RDS::DBParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: mariadb10.6
      Parameters:
        require_secure_transport: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RDSDBParameterGroup:
    Type: AWS::RDS::DBParameterGroup
    Properties:
      Description:
        Fn::Sub: ${AWS::StackName}-example
      Family: postgres15
      Parameters:
        rds.force_ssl: false
```

[CT.RDS.PR.29] Require an Amazon RDS cluster not be configured to be publicly accessible by means of the 'PubliclyAccessible' property

This control checks whether an Amazon Relational Database Service database cluster is configured to be publicly accessible, or not, as determined by the setting of the PubliclyAccessible property.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBCluster`
- **CloudFormation guard rule:** CT.RDS.PR.29 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.29 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.29 example templates

Explanation

The PubliclyAccessible property in the Amazon RDS DB cluster CloudFormation resource indicates whether the DB cluster is publicly accessible. When the DB instance is configured with the PubliclyAccessible property set to true, its Domain Name System (DNS) endpoint resolves to the public IP address from outside of the DB cluster's virtual private cloud (VPC), and it also resolves to the private IP address from within the DB cluster's VPC.

Unless you intend for your Amazon RDS DB cluster to be publicly accessible, do not configure the Amazon RDS DB cluster with the PubliclyAccessible value set to true, because this configuration may allow unwanted traffic to your database instance.

Remediation for rule failure

Set the value of the PubliclyAccessible property to false.

The examples that follow show how to implement this remediation.

Amazon RDS DB Cluster - Example

An Amazon RDS Multi-AZ Postgres DB cluster configured not to be publicly accessible. The example is shown in JSON and in YAML.

JSON example

```
{
    "DBCluster": {
        "Type": "AWS::RDS::DBCluster",
        "Properties": {
            "Engine": "postgres",
            "DBClusterInstanceClass": "db.m6gd.large",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
```

```
            },
            "DBSubnetGroupName": {
                "Ref": "DBSubnetGroup"
            },
            "AllocatedStorage": 100,
            "StorageType": "io1",
            "Iops": 3000,
            "PubliclyAccessible": false
        }
    }
}
```

YAML example

```
DBCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: postgres
    DBClusterInstanceClass: db.m6gd.large
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBClusterSecret}::password}}'
    DBSubnetGroupName: !Ref 'DBSubnetGroup'
    AllocatedStorage: 100
    StorageType: io1
    Iops: 3000
    PubliclyAccessible: false
```

CT.RDS.PR.29 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_cluster_public_access_check
#
# Description:
```

```
#   This control checks whether an Amazon RDS database cluster is configured to be
 publicly accessible, or not, as determined by the setting of the PubliclyAccessible
 property.
#
# Reports on:
#   AWS::RDS::DBCluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS DB cluster resources
#       Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster resource
#       And: 'Engine' has been provided and set to a database engine type other than a
#            Multi-AZ database engine (type other than 'mysql' or 'postgres')
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster resource
#       And: 'Engine' has been provided and set to a Multi-AZ database engine
#            ('mysql', 'postgres')
#       And: 'PubliclyAccessible' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster resource
#       And: 'Engine' has been provided and set to a Multi-AZ database engine
#            ('mysql', 'postgres')
#       And: 'PubliclyAccessible' has been provided and set to a value other than
 bool(false)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon RDS DB cluster resource
#       And: 'Engine' has been provided and set to a Multi-AZ database engine
#            ('mysql', 'postgres')
```

```
#       And: 'PubliclyAccessible' has been provided and set to bool(false)
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let RDS_DB_CLUSTER_TYPE = "AWS::RDS::DBCluster"
let MULTI_AZ_ENGINE_TYPES = [ "mysql", "postgres" ]

#
# Assignments
#
let rds_db_clusters = Resources.*[ Type == %RDS_DB_CLUSTER_TYPE ]

#
# Primary Rules
#
rule rds_cluster_public_access_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %rds_db_clusters not empty {
    check(%rds_db_clusters.Properties)
        <<
        [CT.RDS.PR.29]: Require an Amazon RDS cluster not be configured to be publicly
 accessible by means of the 'PubliclyAccessible' property
        [FIX]: Set the value of the PubliclyAccessible property to false.
        >>
}

rule rds_cluster_public_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.29]: Require an Amazon RDS cluster not be configured to be publicly
 accessible by means of the 'PubliclyAccessible' property
        [FIX]: Set the value of the PubliclyAccessible property to false.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_cluster) {
    %rds_db_cluster[
        filter_multi_az_engine(this)
```

```
    ] {
        # Scenario 2
        PubliclyAccessible exists

        # Scenarios 3 and 4
        PubliclyAccessible == false
    }
}

rule filter_multi_az_engine(rds_db_cluster) {
    %rds_db_cluster {
        Engine exists
        Engine in %MULTI_AZ_ENGINE_TYPES
    }
}



#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.RDS.PR.29 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  DBClusterSecret:
```

```
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      DBClusterIdentifier: example-db-cluster
      DBClusterInstanceClass: db.m5d.large
      MasterUsername:
        Fn::Sub: "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
      MasterUserPassword:
        Fn::Sub: "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
      Engine: mysql
      AllocatedStorage: 100
      StorageType: io1
      Iops: 1000
      PubliclyAccessible: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBClusterSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
  DBCluster:
    Type: AWS::RDS::DBCluster
    Properties:
      DBClusterIdentifier: example-db-cluster-public
```

```
      DBClusterInstanceClass: db.m5d.large
      MasterUsername:
        Fn::Sub: "{{resolve:secretsmanager:${DBClusterSecret}::username}}"
      MasterUserPassword:
        Fn::Sub: "{{resolve:secretsmanager:${DBClusterSecret}::password}}"
      Engine: mysql
      AllocatedStorage: 100
      StorageType: io1
      Iops: 1000
      PubliclyAccessible: true
```

[CT.RDS.PR.30] Require that an Amazon RDS database instance has encryption at rest configured to use a KMS key that you specify for supported engine types

This control checks whether storage encryption is enabled for your Amazon RDS database (DB) instance, and that the encryption uses a KMS key that you specify for supported engine types.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::RDS::DBInstance`
- **CloudFormation guard rule:** CT.RDS.PR.30 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.RDS.PR.30 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.RDS.PR.30 example templates

Explanation

As an added layer of security for your sensitive data in Amazon RDS DB instances, you can configure your Amazon RDS DB instances to be encrypted at rest. To encrypt your Amazon RDS DB instances and snapshots at rest, enable the encryption option for your Amazon RDS DB instances.

Data that is encrypted at rest includes the underlying storage for DB instances, its automated backups, read replicas, and snapshots.

Amazon RDS-encrypted DB instances use the open standard AES-256 encryption algorithm to encrypt your data residing on the server that hosts your Amazon RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently, with a minimal impact on performance. You do not need to modify your database client applications to use encryption.

Amazon RDS encryption is available for all database engines and storage types. Amazon RDS encryption is available for most DB instance classes.

Usage considerations

- This control applies only to Amazon RDS DB engine types `mariadb`, `mysql`, `oracle-ee`, `oracle-ee-cdb`, `oracle-se2`, `oracle-se2-cdb`, `postgres`, `sqlserver-ee`, `sqlserver-se`, and `sqlserver-web`
- This control applies only when the `Engine` property is provided. It does not apply when restoring from a DB snapshot or cluster snapshot where an `Engine` has not been set explicitly.
- This control requires that a KMS key is specified for Amazon RDS DB instance resources. It does not check the properties of the KMS key used, such as whether the KMS key is customer-managed or service-managed.
- Consider using a customer-managed key if you want full control over the KMS key, which includes establishing and maintaining the key's policies, IAM policies, and grants, as well as enabling and disabling the key, rotating its cryptographic material, adding tags, creating aliases that refer to the KMS key, and scheduling the KMS key for deletion.

Remediation for rule failure

Set the KmsKeyId property to the ARN of an AWS KMS key that is configured to grant key usage permissions to Amazon RDS.

The examples that follow show how to implement this remediation.

Amazon RDS DB instance - Example

An Amazon RDS DB instance with storage encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "RDSDBInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "Engine": "postgres",
            "EngineVersion": 14.2,
            "DBInstanceClass": "db.m5.large",
            "StorageType": "gp2",
            "AllocatedStorage": 5,
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${DBInstanceSecret}::password}}"
            },
            "StorageEncrypted": true,
            "KmsKeyId": {
                "Ref": "KMSKey"
            }
        }
    }
}
```

YAML example

```
RDSDBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: 14.2
    DBInstanceClass: db.m5.large
    StorageType: gp2
    AllocatedStorage: 5
```

```
    MasterUsername: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    StorageEncrypted: true
    KmsKeyId: !Ref 'KMSKey'
```

CT.RDS.PR.30 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   rds_storage_encrypted_kms_key_check
#
# Description:
#   This control checks whether storage encryption is enabled for your Amazon RDS
 database (DB) instance, and that the encryption uses a KMS key that you specify for
 supported engine types.
#
# Reports on:
#   AWS::RDS::DBInstance
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon RDS DB instance resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon RDS DB instance resource
#       And: 'Engine' is not one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
#            'oracle-se2', 'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-
se',
#            'sqlserver-web'
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
#            'oracle-se2', 'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-
se',
#            'sqlserver-web'
#       And: 'KmsKeyId' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
#            'oracle-se2', 'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-
se',
#            'sqlserver-web'
#       And: 'KmsKeyId' has been provided as an empty string or invalid local reference
#            to a KMS key ID or alias or ARN
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Amazon RDS DB instance resource
#       And: 'Engine' is one of 'mariadb', 'mysql', 'oracle-ee', 'oracle-ee-cdb',
#            'oracle-se2', 'oracle-se2-cdb', 'postgres', 'sqlserver-ee', 'sqlserver-
se',
#            'sqlserver-web'
#       And: 'KmsKeyId' has been provided as a non-empty string or valid local
 reference
#            to a KMS key ID or alias or ARN
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let RDS_DB_INSTANCE_TYPE = "AWS::RDS::DBInstance"
let SUPPORTED_RDS_INSTANCE_ENGINES = [
    "mariadb", "mysql", "oracle-ee", "oracle-ee-cdb", "oracle-se2",
    "oracle-se2-cdb", "postgres", "sqlserver-ee", "sqlserver-se",
    "sqlserver-web"
]

#
```

```
# Assignments
#
let rds_db_instances = Resources.*[ Type == %RDS_DB_INSTANCE_TYPE ]

#
# Primary Rules
#
rule rds_storage_encrypted_kms_key_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %rds_db_instances not empty {
    check(%rds_db_instances.Properties)
        <<
        [CT.RDS.PR.30]: Require that an Amazon RDS database instance has encryption at
 rest configured to use a KMS key that you specify for supported engine types
        [FIX]: Set the KmsKeyId property to the ARN of an AWS KMS key that is
 configured to grant key usage permissions to Amazon RDS.
        >>
}

rule rds_storage_encrypted_kms_key_check when is_cfn_hook(%INPUT_DOCUMENT,
 %RDS_DB_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%RDS_DB_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.RDS.PR.30]: Require that an Amazon RDS database instance has encryption at
 rest configured to use a KMS key that you specify for supported engine types
        [FIX]: Set the KmsKeyId property to the ARN of an AWS KMS key that is
 configured to grant key usage permissions to Amazon RDS.
        >>
}

#
# Parameterized Rules
#
rule check(rds_db_instance) {
    %rds_db_instance [
        # Scenario 2
        filter_engine(this)
    ] {
        # Scenario 3
        KmsKeyId exists

        # Scenarios 4 and 5
        check_is_string_and_not_empty(KmsKeyId) or
        check_local_references(%INPUT_DOCUMENT, KmsKeyId, "AWS::KMS::Key") or
        check_local_references(%INPUT_DOCUMENT, KmsKeyId, "AWS::KMS::Alias")
```

```
    }
}

rule filter_engine(rds_db_instance) {
    %rds_db_instance {
        Engine exists
        Engine is_string
        Engine in %SUPPORTED_RDS_INSTANCE_ENGINES
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_local_references(doc, reference_properties, referenced_RESOURCE_TYPE) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_RESOURCE_TYPE)
                <<Local Stack reference was invalid>>
        }
    }
}
```

```
rule query_for_resource(doc, resource_key, referenced_RESOURCE_TYPE) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_RESOURCE_TYPE
    }
}
```

CT.RDS.PR.30 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: example-policy
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS:
              Fn::Sub: arn:${AWS::Partition}:iam::${AWS::AccountId}:root
          Action: kms:*
          Resource: '*'
      KeySpec: SYMMETRIC_DEFAULT
      EnableKeyRotation: true
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS DB instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasteruser"}'
        GenerateStringKey: password
        PasswordLength: 22
```

```
        ExcludeCharacters: '"@/\'
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: mysql
      EngineVersion: 5.7
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
      StorageEncrypted: true
      KmsKeyId:
        Ref: KMSKey
    DeletionPolicy: Delete
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  DBInstanceSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: RDS instance secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "exampleuser"}'
        GenerateStringKey: password
        PasswordLength: 22
        ExcludeCharacters: "/@\""
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: postgres
      EngineVersion: 15.4
      DBInstanceClass: db.m5.large
      StorageType: gp2
      AllocatedStorage: 5
      MasterUsername:
```

```
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${DBInstanceSecret}::password}}'
    DeletionPolicy: Delete
```

Amazon Redshift controls

Topics

- [CT.REDSHIFT.PR.1] Require an Amazon Redshift cluster to prohibit public access
- [CT.REDSHIFT.PR.2] Require an Amazon Redshift cluster to have automatic snapshots configured
- [CT.REDSHIFT.PR.3] Require an Amazon Redshift cluster to have audit logging configured
- [CT.REDSHIFT.PR.4] Require an Amazon Redshift cluster to have automatic upgrades to major versions configured
- [CT.REDSHIFT.PR.5] Require an Amazon Redshift cluster to have enhanced VPC routing
- [CT.REDSHIFT.PR.6] Require an Amazon Redshift cluster to have a unique administrator username
- [CT.REDSHIFT.PR.7] Require an Amazon Redshift cluster to have a unique database name
- [CT.REDSHIFT.PR.8] Require an Amazon Redshift cluster to be encrypted
- [CT.REDSHIFT.PR.9] Require that an Amazon Redshift cluster parameter group is configured to use Secure Sockets Layer (SSL) for encryption of data in transit

[CT.REDSHIFT.PR.1] Require an Amazon Redshift cluster to prohibit public access

This control checks whether Amazon Redshift clusters are configured to prohibit public access.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.1 example templates

Explanation

The PubliclyAccessible attribute of the Amazon Redshift cluster configuration indicates whether the cluster is publicly accessible. When the cluster is configured with the `PubliclyAccessible` parameter set to `true`, it is an internet-facing instance that has a publicly resolvable DNS name, which resolves to a public IP address.

When the cluster is not publicly accessible, it is an internal instance with a DNS name that resolves to a private IP address. Unless you intend for your cluster to be publicly accessible, the cluster should not be configured with `PubliclyAccessible` set to `true`.

Remediation for rule failure

Set `PubliclyAccessible` to `false`.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured to prohibit public access. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RedshiftClusterSecret}::username}}"
            },
            "MasterUserPassword": {
```

```
                "Fn::Sub": "{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}"
            },
            "NodeType": "ds2.xlarge",
            "PubliclyAccessible": false
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:
${RedshiftClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}'
    NodeType: ds2.xlarge
    PubliclyAccessible: false
```

CT.REDSHIFT.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_cluster_public_access_check
#
# Description:
#   This control checks whether Amazon Redshift clusters are configured to prohibit
 public access.
#
# Reports on:
#   AWS::Redshift::Cluster
```

```
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains a Redshift cluster resource
#       And: 'PubliclyAccessible' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains a Redshift cluster resource
#       And: 'PubliclyAccessible' has been specified
#       And: 'PubliclyAccessible' has been set to bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation document or CloudFormation hook
 document
#       And: The input document contains a Redshift cluster resource
#       And: 'PubliclyAccessible' has been specified
#       And: 'PubliclyAccessible' has been set to bool(false)
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]
```

```
#
# Primary Rules
#
rule redshift_cluster_public_access_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %redshift_clusters not empty  {
    check(%redshift_clusters.Properties)
         <<
        [CT.REDSHIFT.PR.1]: Require an Amazon Redshift cluster to prohibit public
 access
        [FIX]: Set 'PubliclyAccessible' to 'false'.
        >>
}

rule redshift_cluster_public_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
         <<
        [CT.REDSHIFT.PR.1]: Require an Amazon Redshift cluster to prohibit public
 access
        [FIX]: Set 'PubliclyAccessible' to 'false'.
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        PubliclyAccessible exists
        # Scenario 3 and 4
        PubliclyAccessible == false
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
```

```
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: true
```

[CT.REDSHIFT.PR.2] Require an Amazon Redshift cluster to have automatic snapshots configured

This control checks whether Amazon Redshift clusters have automated snapshots enabled, and that the clusters are set with an automated snapshot retention period greater than or equal to seven (7) days.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.2 example templates

Explanation

Backups help you to recover more quickly from a security incident. They strengthen the resilience of your systems. Amazon Redshift takes periodic snapshots by default. This control checks whether automatic snapshots are created and retained for at least seven days.

Remediation for rule failure

Set `AutomatedSnapshotRetentionPeriod` to an integer value greater than or equal to 7 days.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured with automatic snapshots active. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "ds2.xlarge",
            "AutomatedSnapshotRetentionPeriod": 7
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: ds2.xlarge
    AutomatedSnapshotRetentionPeriod: 7
```

CT.REDSHIFT.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_backup_enabled_check
#
# Description:
#   This control checks whether Amazon Redshift clusters have automated snapshots
 enabled, and that the clusters are set with an automated snapshot retention period
 greater than or equal to seven (7) days.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AutomatedSnapshotRetentionPeriod' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AutomatedSnapshotRetentionPeriod' has been specified
#       And: 'AutomatedSnapshotRetentionPeriod' has been set to '0'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AutomatedSnapshotRetentionPeriod' has been specified
#       And: 'AutomatedSnapshotRetentionPeriod' has been set to a value <7
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AutomatedSnapshotRetentionPeriod' has been specified
#       And: 'AutomatedSnapshotRetentionPeriod' has been set to a value >= 7
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]

#
# Primary Rules
#
rule redshift_backup_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                        %redshift_clusters not empty  {
    check(%redshift_clusters.Properties)
```

```
        <<
        [CT.REDSHIFT.PR.2]: Require an Amazon Redshift cluster to have automatic
 snapshots configured
        [FIX]: Set 'AutomatedSnapshotRetentionPeriod' to an integer value greater than
 or equal to 7 days.
        >>
}

rule redshift_backup_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.2]: Require an Amazon Redshift cluster to have automatic
 snapshots configured
        [FIX]: Set 'AutomatedSnapshotRetentionPeriod' to an integer value greater than
 or equal to 7 days.
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        AutomatedSnapshotRetentionPeriod exists
        # Scenario 3, 4 and 5
        AutomatedSnapshotRetentionPeriod >= 7
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      AutomatedSnapshotRetentionPeriod: 7
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
```

```
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      AutomatedSnapshotRetentionPeriod: 5
```

[CT.REDSHIFT.PR.3] Require an Amazon Redshift cluster to have audit logging configured

This control checks whether an Amazon Redshift cluster has audit logging activated.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.3 example templates

Explanation

Amazon Redshift audit logging provides additional information about connections and user activities in your cluster. This data can be stored and secured in Amazon S3, and it can be helpful for security audits and investigations.

Remediation for rule failure

Provide a `LoggingProperties` configuration and set `BucketName` to the name of an Amazon S3 bucket configured to receive Amazon Redshift audit logs.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured with audit logging enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "dc2.large",
            "PubliclyAccessible": false,
            "LoggingProperties": {
                "BucketName": {
                    "Ref": "S3Bucket"
                },
                "S3KeyPrefix": "sample-cluster-logs"
            }
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: dc2.large
    PubliclyAccessible: false
    LoggingProperties:
      BucketName: !Ref 'S3Bucket'
      S3KeyPrefix: sample-cluster-logs
```

CT.REDSHIFT.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_cluster_audit_logging_enabled_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster has audit logging activated.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'LoggingProperties' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'LoggingProperties' has been specified
#       And: 'BucketName' on 'LoggingProperties' has been specified and is an empty
 string or invalid local reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'LoggingProperties' has been specified
#       And: 'BucketName' on 'LoggingProperties' has been specified and is a non-empty
 string or valid local reference
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]

#
# Primary Rules
#
rule redshift_cluster_audit_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %redshift_clusters not empty  {
    check(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.3]: Require an Amazon Redshift cluster to have audit logging
 configured
```

```
            [FIX]: Provide a 'LoggingProperties' configuration and set 'BucketName' to
 the name of an Amazon S3 bucket configured to receive Amazon Redshift audit logs.
        >>
}

rule redshift_cluster_audit_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.3]: Require an Amazon Redshift cluster to have audit logging
 configured
            [FIX]: Provide a 'LoggingProperties' configuration and set 'BucketName' to
 the name of an Amazon S3 bucket configured to receive Amazon Redshift audit logs.
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        LoggingProperties exists
        LoggingProperties is_struct

        LoggingProperties {
            # Scenario 3 and 4
            BucketName exists
            check_is_string_and_not_empty(BucketName) or
            check_local_references(%INPUT_DOCUMENT, BucketName, "AWS::S3::Bucket")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}
```

```
rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.REDSHIFT.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
```

```
  BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: S3Bucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal:
            Service: redshift.amazonaws.com
          Action:
          - s3:GetBucketAcl
          Resource:
          - Fn::GetAtt:
            - S3Bucket
            - Arn
        - Effect: Allow
          Principal:
            Service: redshift.amazonaws.com
          Action:
          - s3:PutObject
          Resource:
          - Fn::Join:
            - ''
            - - Fn::GetAtt:
                - S3Bucket
                - Arn
              - /*
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
```

```
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      LoggingProperties:
        BucketName:
          Ref: S3Bucket
        S3KeyPrefix: example-cluster-logs
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

[CT.REDSHIFT.PR.4] Require an Amazon Redshift cluster to have automatic upgrades to major versions configured

This control checks whether automatic major version upgrades are enabled for your Amazon Redshift cluster.

- **Control objective:** Manage vulnerabilities
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.4 example templates

Explanation

Enabling automatic major version upgrades ensures that the latest major version updates to Amazon Redshift clusters are installed during the maintenance window. These updates might include security patches and bug fixes. Keeping up to date with patch installation is an important step in securing systems.

Remediation for rule failure

Set the `AllowVersionUpgrade` property to true or do not specify it (default).

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example One

Amazon Redshift cluster with automatic major version upgrades enabled through CloudFormation defaults. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "exampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "ds2.xlarge"
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: exampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: ds2.xlarge
```

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example Two

Amazon Redshift cluster configured with automatic major version upgrades enabled through the `AllowVersionUpgrade` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "exampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "ds2.xlarge",
            "AllowVersionUpgrade": true
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: exampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: ds2.xlarge
    AllowVersionUpgrade: true
```

CT.REDSHIFT.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_cluster_allow_version_upgrade_check
```

```
#
# Description:
#   Checks whether automatic major version upgrades are enabled for the Amazon Redshift
 cluster.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AllowVersionUpgrade' has been provided
#       And: 'AllowVersionUpgrade' has been set to bool(false)
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AllowVersionUpgrade' has not been provided
#      Then: PASS
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'AllowVersionUpgrade' has been provided
#       And: 'AllowVersionUpgrade' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]

#
# Primary Rules
#
rule redshift_cluster_allow_version_upgrade_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %redshift_clusters not empty  {
    check(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.4]: Require an Amazon Redshift cluster to have automatic
 upgrades to major versions configured
        [FIX]: Set the 'AllowVersionUpgrade' property to true or do not specify it
 (default).
        >>
}

rule redshift_cluster_allow_version_upgrade_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.4]: Require an Amazon Redshift cluster to have automatic
 upgrades to major versions configured
        [FIX]: Set the 'AllowVersionUpgrade' property to true or do not specify it
 (default).
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 3
        AllowVersionUpgrade not exists or
        # Scenario 2 and 4
        AllowVersionUpgrade == true
    }
}

#
# Utility Rules
#
```

```
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      AllowVersionUpgrade: false
```

[CT.REDSHIFT.PR.5] Require an Amazon Redshift cluster to have enhanced VPC routing

This control checks whether an Amazon Redshift cluster has enhanced VPC routing configured.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.5 example templates

Explanation

Enhanced VPC routing forces all `copy` and `unload` traffic between the cluster and the data repositories to go through your VPC. With enhanced routing active, you can use VPC features, such as security groups and network access control lists, to secure network traffic. You can also use VPC Flow Logs to monitor network traffic.

Remediation for rule failure

Set `EnhancedVpcRouting` to `true`.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured with enhanced VPC routing. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "ds2.xlarge",
            "EnhancedVpcRouting": true
```

```
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: ds2.xlarge
    EnhancedVpcRouting: true
```

CT.REDSHIFT.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_enhanced_vpc_routing_enabled_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster has enhanced VPC routing
 configured.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'EnhancedVpcRouting' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'EnhancedVpcRouting' has been specified
#       And: 'EnhancedVpcRouting' has been set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'EnhancedVpcRouting' has been specified
#       And: 'EnhancedVpcRouting' has been set to bool(true)
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]

#
# Primary Rules
#
rule redshift_enhanced_vpc_routing_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                                      %redshift_clusters not empty {
    check(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.5]: Require an Amazon Redshift cluster to have enhanced VPC
 routing
        [FIX]: Set 'EnhancedVpcRouting' to 'true'.
```

```
        >>
}

rule redshift_enhanced_vpc_routing_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.5]: Require an Amazon Redshift cluster to have enhanced VPC
 routing
        [FIX]: Set 'EnhancedVpcRouting' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        EnhancedVpcRouting exists
        # Scenario 3 and 4
        EnhancedVpcRouting == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      EnhancedVpcRouting: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
```

```
      SecretStringTemplate: '{"username": "examplemasterusername"}'
      GenerateStringKey: password
      PasswordLength: 16
      ExcludeCharacters: "\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      EnhancedVpcRouting: false
```

[CT.REDSHIFT.PR.6] Require an Amazon Redshift cluster to have a unique administrator username

This control checks whether an Amazon Redshift cluster has changed the administrator username from its default value.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.6 example templates

Explanation

When creating an Amazon Redshift cluster, you should change the default administrator username to a unique value. Default usernames are public knowledge, so they should be changed upon configuration. Changing the default username reduces the risk of unintended access.

Remediation for rule failure

Set `MasterUsername` to a value other than `awsuser`.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured with an administrator username different from the default value of `awsuser`. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": "samplemasterusername",
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}"
            },
            "NodeType": "ds2.xlarge"
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
```

```
    MasterUsername: samplemasterusername
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}'
    NodeType: ds2.xlarge
```

CT.REDSHIFT.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_default_admin_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster has changed the
 administrator username from its default value.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'MasterUsername' has not been specified
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
```

```
#       And: 'MasterUsername' has been specified
#       And: 'MasterUsername' has been set to 'awsuser'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: 'MasterUsername' has been specified
#       And: 'MasterUsername' has been set to a value not equal to 'awsuser'
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %REDSHIFT_CLUSTER_TYPE ]

#
# Primary Rules
#
rule redshift_default_admin_check when is_cfn_template(%INPUT_DOCUMENT)
                                       %redshift_clusters not empty  {
    check(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.6]: Require an Amazon Redshift cluster to have a unique
 administrator username
        [FIX]: Set 'MasterUsername' to a value other than 'awsuser'.
        >>
}

rule redshift_default_admin_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.6]: Require an Amazon Redshift cluster to have a unique
 administrator username
        [FIX]: Set 'MasterUsername' to a value other than 'awsuser'.
        >>
}
```

```
#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        MasterUsername exists
        check_is_string_and_not_empty(MasterUsername)

        # Scenario 3 and 4
        MasterUsername != "awsuser"
    }
}

#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
      MasterUsername: examplemasterusername
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "awsuser"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      DBName: exampledb
```

```
      MasterUsername: awsuser
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

[CT.REDSHIFT.PR.7] Require an Amazon Redshift cluster to have a unique database name

This control checks whether an Amazon Redshift cluster has changed its database name from the default value.

- **Control objective:** Protect configurations
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.7 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.7 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.REDSHIFT.PR.7 example templates

Explanation

When creating a Redshift cluster, you should change the default database name to a unique value. Default names are public knowledge, so they should be changed upon configuration. For example, a well-known name can lead to inadvertent access, if included in IAM policy conditions.

Remediation for rule failure

Set `DBName` to a database name that is different from the default value of `dev`.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

Amazon Redshift cluster configured with a database name different from the default value of dev. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RedshiftClusterSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}"
            },
            "NodeType": "dc2.large",
            "PubliclyAccessible": false
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:
${RedshiftClusterSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:
${RedshiftClusterSecret}::password}}'
    NodeType: dc2.large
    PubliclyAccessible: false
```

CT.REDSHIFT.PR.7 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_default_db_name_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster has changed its database
 name from the default value.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a Redshift cluster resource
#       And: The 'DBName' property has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Redshift cluster resource
#       And: The 'DBName' property has been provided with a value of 'dev' or an empty
 string
#      Then: FAIL
#   Scenario: 4
```

```
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains Redshift cluster resource
#       And: The 'DBName' property has been provided with a non-empty string that is
 not equal to 'dev'
#      Then: PASS

#
# Constants
#
let RESOURCE_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this
let INVALID_DB_NAME_STRING = "dev"

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %RESOURCE_TYPE ]

#
# Primary Rules
#
rule redshift_default_db_name_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %redshift_clusters not empty {
    check_db_name(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.7]: Require an Amazon Redshift cluster to have a unique
 database name
        [FIX]: Set 'DBName' to a database name that is different from the default value
 of 'dev'.
        >>
}

rule redshift_default_db_name_check when is_cfn_hook(%INPUT_DOCUMENT, %RESOURCE_TYPE) {
    check_db_name(%INPUT_DOCUMENT.%RESOURCE_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.7]: Require an Amazon Redshift cluster to have a unique
 database name
        [FIX]: Set 'DBName' to a database name that is different from the default value
 of 'dev'.
        >>
}

#
# Parameterized Rules
```

```
#
rule check_db_name(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        DBName exists

        # Scenario 3 and 4
        check_is_string_and_not_empty(DBName)
        DBName != %INVALID_DB_NAME_STRING
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}
```

CT.REDSHIFT.PR.7 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: "single-node"
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      DBName: exampledb
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: "single-node"
      MasterUsername:
```

```
      Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword:
      Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: dc2.large
    PubliclyAccessible: false
    DBName: dev
```

[CT.REDSHIFT.PR.8] Require an Amazon Redshift cluster to be encrypted

This control checks whether an Amazon Redshift cluster is encrypted.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::Cluster`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.REDSHIFT.PR.8 example templates

Explanation

In Amazon Redshift, you can enable database encryption for your clusters, which helps protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots.

Remediation for rule failure

Set the value of the `Encrypted` property to true.

The examples that follow show how to implement this remediation.

Amazon Redshift Cluster - Example

An Amazon Redshift cluster configured with encryption enabled. The example is shown in JSON and in YAML.

JSON example

```
{
    "RedshiftCluster": {
        "Type": "AWS::Redshift::Cluster",
        "Properties": {
            "ClusterType": "single-node",
            "DBName": "sampledb",
            "MasterUsername": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::username}}"
            },
            "MasterUserPassword": {
                "Fn::Sub": "{{resolve:secretsmanager:${RedshiftSecret}::password}}"
            },
            "NodeType": "dc2.large",
            "PubliclyAccessible": false,
            "Encrypted": true
        }
    }
}
```

YAML example

```
RedshiftCluster:
  Type: AWS::Redshift::Cluster
  Properties:
    ClusterType: single-node
    DBName: sampledb
    MasterUsername: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
    MasterUserPassword: !Sub '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
    NodeType: dc2.large
    PubliclyAccessible: false
    Encrypted: true
```

CT.REDSHIFT.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_cluster_encrypted_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster is encrypted.
#
# Reports on:
#   AWS::Redshift::Cluster
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon Redshift cluster resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster resource
#       And: 'Encrypted' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster resource
#       And: 'Encrypted' has been provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster resource
#       And: 'Encrypted' has been provided and set to bool(true)
#      Then: PASS

#
```

```
# Constants
#
let RESOURCE_TYPE = "AWS::Redshift::Cluster"
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_clusters = Resources.*[ Type == %RESOURCE_TYPE ]

#
# Primary Rules
#
rule redshift_cluster_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                           %redshift_clusters not empty {
    check(%redshift_clusters.Properties)
        <<
        [CT.REDSHIFT.PR.8]: Require an Amazon Redshift cluster to be encrypted
        [FIX]: Set the value of the 'Encrypted' property to true.
        >>
}

rule redshift_cluster_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT, %RESOURCE_TYPE)
 {
    check(%INPUT_DOCUMENT.%RESOURCE_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.8]: Require an Amazon Redshift cluster to be encrypted
        [FIX]: Set the value of the 'Encrypted' property to true.
        >>
}

#
# Parameterized Rules
#
rule check(redshift_cluster) {
    %redshift_cluster {
        # Scenario 2
        Encrypted exists

        # Scenario 3 and 4
        Encrypted == true
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
```

```
      DBName: exampledb
      Encrypted: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  RedshiftSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: Redshift cluster secret
      GenerateSecretString:
        SecretStringTemplate: '{"username": "examplemasterusername"}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: "'\"@/\\"
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    Properties:
      ClusterType: single-node
      MasterUsername:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::username}}'
      MasterUserPassword:
        Fn::Sub: '{{resolve:secretsmanager:${RedshiftSecret}::password}}'
      NodeType: dc2.large
      PubliclyAccessible: false
      DBName: exampledb
      Encrypted: false
```

[CT.REDSHIFT.PR.9] Require that an Amazon Redshift cluster parameter group is configured to use Secure Sockets Layer (SSL) for encryption of data in transit

This control checks whether an Amazon Redshift cluster parameter group is configured to require encryption by means of Secure Sockets Layer (SSL), for data in transit.

- **Control objective:** Encrypt data in transit
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::Redshift::ClusterParameterGroup`
- **CloudFormation guard rule:** CT.REDSHIFT.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.REDSHIFT.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.REDSHIFT.PR.9 example templates

Explanation

In Amazon Redshift, you can enable encryption of data in transit between an Amazon Redshift cluster and SQL clients over JDBC/ODBC. To support SSL connections, Amazon Redshift creates and installs certificates on each cluster, which are issued by AWS Certificate Manager (ACM).

Remediation for rule failure

Set an entry in `Parameters` with a `ParameterName` of `require_ssl` and a `ParameterValue` of true.

The examples that follow show how to implement this remediation.

Amazon Redshift cluster parameter group - Example

An Amazon Redshift cluster parameter group, configured to require encryption of data in transit. The example is shown in JSON and in YAML.

JSON example

```
{
    "ClusterParameterGroup": {
        "Type": "AWS::Redshift::ClusterParameterGroup",
        "Properties": {
            "Description": "Example parameter group",
            "ParameterGroupFamily": "redshift-1.0",
            "Parameters": [
                {
                    "ParameterName": "require_ssl",
```

```
                    "ParameterValue": true
                }
            ]
        }
    }
}
```

YAML example

```
ClusterParameterGroup:
  Type: AWS::Redshift::ClusterParameterGroup
  Properties:
    Description: Example parameter group
    ParameterGroupFamily: redshift-1.0
    Parameters:
      - ParameterName: require_ssl
        ParameterValue: true
```

CT.REDSHIFT.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   redshift_parameter_group_require_tls_ssl_check
#
# Description:
#   This control checks whether an Amazon Redshift cluster parameter group is
 configured to require
#   encryption by means of Secure Sockets Layer (SSL), for data in transit.
#
# Reports on:
#   AWS::Redshift::ClusterParameterGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
```

```
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon Redshift cluster parameter
 group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster parameter group
 resource
#       And: 'Parameters' has not been provided or has been provided as an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster parameter group
 resource
#       And: 'Parameters' has been provided as a non-empty list that does not contain
 an entry with
#            'ParameterName' set to 'require_ssl' and 'ParameterValue' set to
 bool(true) or a supported
#            boolean string value ('true', 'True', 'TRUE', 'on', 'On' or 'ON')
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster parameter group
 resource
#       And: Any entry in 'Parameters' with a 'ParameterName' is set to 'require_ssl'
 has a corresponding
#            'ParameterValue' set to a value other than bool(true) or a supported
#            boolean string value ('true', 'True', 'TRUE', 'on', 'On' or 'ON')
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon Redshift cluster parameter group
 resource
#       And: 'Parameters' has been provided as a non-empty list
#       And: 'Parameters' contains an entry with 'ParameterName' set to 'require_ssl'
 and 'ParameterValue'
#            set to bool(true) or a supported boolean string value ('true', 'True',
 'TRUE', 'on', 'On' or 'ON')
```

```
#       And: All entries in 'Parameters' with a 'ParameterName' set to 'require_ssl'
 have a corresponding
#            'ParameterValue' set to bool(true) or a supported boolean string
#            value ('true', 'True', 'TRUE', 'on', 'On' or 'ON')
#      Then: PASS

#
# Constants
#
let REDSHIFT_CLUSTER_PARAMETER_GROUP_TYPE = "AWS::Redshift::ClusterParameterGroup"
let REDSHIFT_SUPPORTED_TRUE_VALUES = [ true, /^(true|True|TRUE|on|On|ON)$/ ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let redshift_cluster_parameter_groups = Resources.*[ Type ==
 %REDSHIFT_CLUSTER_PARAMETER_GROUP_TYPE ]

#
# Primary Rules
#
rule redshift_parameter_group_require_tls_ssl_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %redshift_cluster_parameter_groups not empty {
    check(%redshift_cluster_parameter_groups.Properties)
        <<
        [CT.REDSHIFT.PR.9]: Require that an Amazon Redshift cluster parameter group is
 configured to use Secure Sockets Layer (SSL) for encryption of data in transit
        [FIX]: Set an entry in 'Parameters' with a 'ParameterName' of 'require_ssl' and
 a 'ParameterValue' of true.
        >>
}

rule redshift_parameter_group_require_tls_ssl_check when is_cfn_hook(%INPUT_DOCUMENT,
 %REDSHIFT_CLUSTER_PARAMETER_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%REDSHIFT_CLUSTER_PARAMETER_GROUP_TYPE.resourceProperties)
        <<
        [CT.REDSHIFT.PR.9]: Require that an Amazon Redshift cluster parameter group is
 configured to use Secure Sockets Layer (SSL) for encryption of data in transit
        [FIX]: Set an entry in 'Parameters' with a 'ParameterName' of 'require_ssl' and
 a 'ParameterValue' of true.
        >>
```

```
}

#
# Parameterized Rules
#
rule check(redshift_cluster_parameter_groups) {
    %redshift_cluster_parameter_groups {
        # Scenario 2
        Parameters exists
        Parameters is_list
        Parameters not empty

        # Scenarios 3, 4 and 5
        some Parameters[*] {
            ParameterName exists
            ParameterValue exists

            ParameterName == "require_ssl"
            ParameterValue in %REDSHIFT_SUPPORTED_TRUE_VALUES
        }
        Parameters [
            ParameterName exists
            ParameterName == "require_ssl"
        ] {
            ParameterValue exists
            ParameterValue in %REDSHIFT_SUPPORTED_TRUE_VALUES
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.REDSHIFT.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  ClusterParameterGroup:
    Type: AWS::Redshift::ClusterParameterGroup
    Properties:
      Description: Example parameter group
      ParameterGroupFamily: redshift-1.0
      Parameters:
      - ParameterName: require_ssl
        ParameterValue: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ClusterParameterGroup:
    Type: AWS::Redshift::ClusterParameterGroup
    Properties:
      Description: Example parameter group
      ParameterGroupFamily: redshift-1.0
```

Amazon Simple Storage Service (Amazon S3) controls

Topics

- [CT.S3.PR.1] Require an Amazon S3 bucket to have block public access settings configured
- [CT.S3.PR.2] Require an Amazon S3 bucket to have server access logging configured

- [CT.S3.PR.3] Require an Amazon S3 buckets to have versioning configured and a lifecycle policy
- [CT.S3.PR.4] Require an Amazon S3 bucket to have event notifications configured
- [CT.S3.PR.5] Require that an Amazon S3 bucket does not manage user access with an access control list (ACL)
- [CT.S3.PR.6] Require an Amazon S3 bucket to have lifecycle policies configured
- [CT.S3.PR.8] Require that Amazon S3 bucket requests use Secure Socket Layer
- [CT.S3.PR.9] Require that an Amazon S3 bucket has S3 Object Lock activated
- [CT.S3.PR.10] Require an Amazon S3 bucket to have server-side encryption configured using an AWS KMS key
- [CT.S3.PR.11] Require an Amazon S3 bucket to have versioning enabled
- [CT.S3.PR.12] Require an Amazon S3 access point to have a Block Public Access (BPA) configuration with all options set to true

[CT.S3.PR.1] Require an Amazon S3 bucket to have block public access settings configured

This control checks whether your Amazon Simple Storage Service (Amazon S3) bucket has a bucket-level Block Public Access (BPA) configuration.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.1 example templates

Explanation

Block Public Access at the Amazon S3 bucket level provides controls to ensure that objects never have public access. Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.

Unless you intend to have your S3 buckets publicly accessible, you should configure the bucket level Amazon S3 Block Public Access feature.

Usage considerations

- This control is incompatible with Amazon S3 buckets that require a public access configuration.

Remediation for rule failure

The parameters `BlockPublicAcls`, `BlockPublicPolicy`, `IgnorePublicAcls`, `RestrictPublicBuckets` must be set to true under the bucket-level `PublicAccessBlockConfiguration`.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example

Amazon S3 bucket with a bucket level Block Public Access configuration that ensures objects never have public access. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "PublicAccessBlockConfiguration": {
                "BlockPublicAcls": true,
                "BlockPublicPolicy": true,
                "IgnorePublicAcls": true,
                "RestrictPublicBuckets": true
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
```

CT.S3.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_level_public_access_prohibited_check
#
# Description:
#   Checks whether Amazon Simple Storage Service (Amazon S3) buckets have a bucket-
level Block Public Access (BPA)
#   configuration.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any Amazon S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'PublicAccessBlockConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'PublicAccessBlockConfiguration' has been provided
#       And: 'BlockPublicAcls' or 'BlockPublicPolicy' or 'IgnorePublicAcls' or
 'RestrictPublicBuckets'
#            have not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket Resource
#       And: 'PublicAccessBlockConfiguration' has been provided
#       And: Any of 'BlockPublicAcls' or 'BlockPublicPolicy' or 'IgnorePublicAcls' or
 'RestrictPublicBuckets'
#            have been set to a value other than bool(true) (e.g. bool(false),
 str(false), other)
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket Resource
#       And: 'PublicAccessBlockConfiguration' has been provided
#       And: 'BlockPublicAcls' or 'BlockPublicPolicy' or 'IgnorePublicAcls' or
 'RestrictPublicBuckets'
#            have all been set to bool(true)
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]
```

```
#
# Primary Rules
#
rule s3_bucket_level_public_access_prohibited_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.1]: Require an Amazon S3 bucket to have block public access settings
 configured
        [FIX]: The parameters 'BlockPublicAcls', 'BlockPublicPolicy',
 'IgnorePublicAcls', 'RestrictPublicBuckets' must be set to true under the bucket-level
 'PublicAccessBlockConfiguration'.
        >>
}

rule s3_bucket_level_public_access_prohibited_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.1]: Require an Amazon S3 bucket to have block public access settings
 configured
        [FIX]: The parameters 'BlockPublicAcls', 'BlockPublicPolicy',
 'IgnorePublicAcls', 'RestrictPublicBuckets' must be set to true under the bucket-level
 'PublicAccessBlockConfiguration'.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
        # Scenario 2
        PublicAccessBlockConfiguration exists
        PublicAccessBlockConfiguration is_struct

        PublicAccessBlockConfiguration {
            # Scenario 3
            BlockPublicAcls exists
            BlockPublicPolicy exists
            IgnorePublicAcls exists
            RestrictPublicBuckets exists
```

```
            # Scenarios 4 and 5
            BlockPublicAcls == true
            BlockPublicPolicy == true
            IgnorePublicAcls == true
            RestrictPublicBuckets == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
```

[CT.S3.PR.2] Require an Amazon S3 bucket to have server access logging configured

This control checks whether server access logging is enabled for your Amazon S3 bucket.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.2 example templates

Explanation

Server access logging provides detailed records of requests made to a bucket. Server access logs can assist in security and access audits.

Remediation for rule failure

Set a `LoggingConfiguration` on the S3 bucket and optionally set `DestinationBucketName` to an S3 bucket configured to receive S3 Access Logs.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example

Amazon S3 bucket with a server access logging configuration. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "LoggingConfiguration": {
                "DestinationBucketName": {
                    "Ref": "LoggingBucket"
                }
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    LoggingConfiguration:
      DestinationBucketName: !Ref 'LoggingBucket'
```

CT.S3.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_logging_enabled_check
#
# Description:
#   This control checks whether server access logging is enabled for Amazon S3 buckets.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'LoggingConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'LoggingConfiguration' has been provided
#       And: 'LoggingConfiguration.DestinationbucketName' has been provided with an
 empty string or non-valid local
#            reference
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'LoggingConfiguration' has been provided
```

```
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'LoggingConfiguration' has been provided
#       And: 'LoggingConfiguration.DestinationBucketName' has been provided with a non-
empty string or valid local
#            reference
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_bucket_logging_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.2]: Require an Amazon S3 bucket to have server access logging
 configured
        [FIX]: Set a 'LoggingConfiguration' on the S3 Bucket and optionally set
 'DestinationBucketName' to an S3 bucket configured to receive S3 Access Logs.
        >>
}

rule s3_bucket_logging_enabled_check when is_cfn_hook(%INPUT_DOCUMENT, %S3_BUCKET_TYPE)
 {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.2]: Require an Amazon S3 bucket to have server access logging
 configured
        [FIX]: Set a 'LoggingConfiguration' on the S3 bucket and optionally set
 'DestinationBucketName' to an S3 bucket configured to receive S3 Access Logs.
        >>
```

```
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
        # Scenario 2 and 4
        LoggingConfiguration exists
        LoggingConfiguration is_struct

        LoggingConfiguration {
            when DestinationBucketName exists {
                # Scenario 3, 4 and 5
                check_is_string_and_not_empty(DestinationBucketName) or
                check_local_references(%INPUT_DOCUMENT, DestinationBucketName,
 %S3_BUCKET_TYPE)
            }
        }
    }
}


#
# Utility Rules
#
rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

```
rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.S3.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      LoggingConfiguration: {}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
```

```
  Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
```

[CT.S3.PR.3] Require an Amazon S3 buckets to have versioning configured and a lifecycle policy

This control checks whether your Amazon Simple Storage Service (Amazon S3) version-enabled bucket has a lifecycle policy configured.

- **Control objective:** Optimize costs
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.3 example templates

Explanation

We recommend that you configure lifecycle rules on your Amazon S3 bucket, because these rules help you define actions that you want Amazon S3 to take during an object's lifetime.

Usage considerations

- This control applies only to Amazon S3 buckets with versioning enabled.

Remediation for rule failure

Configure versioning-enabled buckets with at least one active lifecycle rule.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example One

Amazon S3 bucket with versioning enabled and an active lifecycle rule. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "VersioningConfiguration": {
                "Status": "Enabled"
            },
            "LifecycleConfiguration": {
                "Rules": [
                    {
                        "Status": "Enabled",
                        "ExpirationInDays": 1,
                        "Id": "FirstRule"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    VersioningConfiguration:
      Status: Enabled
    LifecycleConfiguration:
      Rules:
        - Status: Enabled
          ExpirationInDays: 1
```

```
          Id: FirstRule
```

CT.S3.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_version_lifecycle_policy_check
#
# Description:
#   Checks whether Amazon Simple Storage Service (Amazon S3) version-enabled buckets
 have lifecycle policy configured.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket Resource
#       And: The S3 bucket does not have versioning enabled (VersioningConfiguration is
 missing or
#            VersioningConfiguration.Status is set to Suspended)
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket Resource
```

```
#       And: The S3 bucket has versioning enabled (VersioningConfiguration.Status is
 set to 'Enabled')
#       And: 'LifecycleConfiguration' has been been provided and there are no 'Rules'
 with 'Status' set to 'Enabled'
#            present in the 'LifecycleConfiguration'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket Resource
#       And: The S3 bucket has versioning enabled (VersioningConfiguration.Status is
 set to 'Enabled')
#       And: 'LifecycleConfiguration' has been been provided and there is at least one
 'Rule' with 'Status' set to
#            'Enabled' in the 'LifecycleConfiguration'
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_version_lifecycle_policy_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.3]: Require an Amazon S3 buckets to have versioning configured and a
 lifecycle policy
        [FIX]: Configure versioning-enabled buckets with at least one active lifecycle
 rule.
        >>
}

rule s3_version_lifecycle_policy_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
```

```
        <<
        [CT.S3.PR.3]: Require an Amazon S3 buckets to have versioning configured and a
 lifecycle policy
        [FIX]: Configure versioning-enabled buckets with at least one active lifecycle
 rule.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket [
        filter_s3_buckets_with_versioning_enabled(this)
    ] {
        # Scenario 2
        LifecycleConfiguration exists
        LifecycleConfiguration is_struct

        LifecycleConfiguration {
            # Scenario 3 and 4
            Rules exists
            Rules is_list
            Rules not empty

            some Rules[*] {
                Status exists
                Status == "Enabled"
            }
        }
    }
}

rule filter_s3_buckets_with_versioning_enabled(s3_bucket) {
    %s3_bucket {
        VersioningConfiguration exists
        VersioningConfiguration is_struct

        VersioningConfiguration {
            Status exists
            Status == "Enabled"
        }
    }
}
```

```
#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      LifecycleConfiguration:
        Rules:
        - Status: Enabled
          ExpirationInDays: 1
          Id: FirstRule
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      LifecycleConfiguration:
        Rules:
        - Status: Disabled
          ExpirationInDays: 1
          Id: FirstRule
```

[CT.S3.PR.4] Require an Amazon S3 bucket to have event notifications configured

This control checks whether Amazon S3 events notifications are enabled on your Amazon S3 bucket.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.4 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.4 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.4 example templates

Explanation

By enabling event notifications, you receive alerts on your Amazon S3 buckets when specific events occur. For example, you can be notified of object creation, object removal, and object restoration. These notifications can alert relevant teams to accidental or intentional modifications that may lead to unauthorized data access.

Remediation for rule failure

Set a `NotificationConfiguration` parameter on your bucket with one of `EventBridgeConfiguration`, `LambdaConfigurations`, `QueueConfigurations` or `TopicConfigurations.`

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example One

Amazon S3 bucket with Amazon EventBridge notifications configured. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "NotificationConfiguration": {
                "EventBridgeConfiguration": {
                    "EventBridgeEnabled": true
                }
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    NotificationConfiguration:
      EventBridgeConfiguration:
        EventBridgeEnabled: true
```

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example Two

Amazon S3 bucket with SNS topic notifications configured. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "NotificationConfiguration": {
                "TopicConfigurations": [
                    {
                        "Topic": {
                            "Ref": "SnsTopic"
                        },
                        "Event": "s3:ReducedRedundancyLostObject"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    NotificationConfiguration:
      TopicConfigurations:
        - Topic: !Ref 'SnsTopic'
          Event: s3:ReducedRedundancyLostObject
```

CT.S3.PR.4 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_event_notifications_enabled_check
#
# Description:
#   This control checks whether Amazon S3 event notifications are enabled on an S3
 bucket.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'NotificationConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'NotificationConfiguration' has been provided
#       And:  At least one of 'EventBridgeConfiguration.EventBridgeEnabled',
 'LambdaConfigurations',
#             'QueueConfigurations', or 'TopicConfigurations' have not been provided or
 provided as empty lists.
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'NotificationConfiguration' has been provided
```

```
#       And: 'EventBridgeConfiguration.EventBridgeEnabled' is set to bool(true) or
 'LambdaConfigurations',
#            'QueueConfigurations', or 'TopicConfigurations' have been provided with at
 least one configuration
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_event_notifications_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                               %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.4]: Require an Amazon S3 bucket to have event notifications
 configured
        [FIX]: Set a 'NotificationConfiguration' parameter on your bucket with one
 of 'EventBridgeConfiguration', 'LambdaConfigurations', 'QueueConfigurations' or
 'TopicConfigurations.'
        >>
}

rule s3_event_notifications_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.4]: Require an Amazon S3 bucket to have event notifications
 configured
        [FIX]: Set a 'NotificationConfiguration' parameter on your bucket with one
 of 'EventBridgeConfiguration', 'LambdaConfigurations', 'QueueConfigurations' or
 'TopicConfigurations.'
        >>
}
```

```
#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
       # Scenario 2
       NotificationConfiguration exists
       NotificationConfiguration is_struct
       NotificationConfiguration {
            # Scenario 3 and 4
            EventBridgeConfiguration exists or
            LambdaConfigurations exists or
            QueueConfigurations exists or
            TopicConfigurations exists

            check_event_bridge_notifications(EventBridgeConfiguration) or
            check_other_notifications(LambdaConfigurations) or
            check_other_notifications(QueueConfigurations) or
            check_other_notifications(TopicConfigurations)
       }
    }
}

rule check_event_bridge_notifications(configuration) {
    %configuration {
        this is_struct
        EventBridgeEnabled exists
        EventBridgeEnabled == true
    }
}

rule check_other_notifications(configuration) {
    %configuration {
        this is_list
        this not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
```

```
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.4 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration: {}
```

[CT.S3.PR.5] Require that an Amazon S3 bucket does not manage user access with an access control list (ACL)

This control checks whether your Amazon Simple Storage Service (Amazon S3) bucket allows user permissions through access control lists.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.5 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.5 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.5 example templates

Explanation

ACLs are legacy access control mechanisms that predate IAM. Instead of ACLs, we recommend using IAM policies or Amazon S3 bucket policies to more easily manage access to your S3 buckets.

Remediation for rule failure

Manage access to Amazon S3 buckets with bucket resource policies and IAM identity policies instead.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example

Amazon S3 bucket that does not allow user permissions through access control lists by omitting the `AccessControl` property. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {}
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties: {}
```

CT.S3.PR.5 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_acl_prohibited_check
#
# Description:
#   Checks whether Amazon Simple Storage Service (Amazon S3) buckets allow user
 permissions through access control lists.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any Amazon S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'AccessControl' has been provided on the S3 bucket resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 bucket resource
#       And: 'AccessControl' has not been provided on the S3 bucket resource
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_bucket_acl_prohibited_check when is_cfn_template(%INPUT_DOCUMENT)
                                         %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.5]: Require that an Amazon S3 bucket does not manage user access with
 an access control list (ACL)
        [FIX]: Manage access to Amazon S3 buckets with bucket resource policies and IAM
 identity policies instead.
        >>
}

rule s3_bucket_acl_prohibited_check when is_cfn_hook(%INPUT_DOCUMENT, %S3_BUCKET_TYPE)
 {
    check(this.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.5]: Require that an Amazon S3 bucket does not manage user access with
 an access control list (ACL)
```

```
        [FIX]: Manage access to Amazon S3 buckets with bucket resource policies and IAM
 identity policies instead.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
       # Scenario 2 and 3
       AccessControl not exists
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.5 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: Private
```

[CT.S3.PR.6] Require an Amazon S3 bucket to have lifecycle policies configured

This control checks whether a lifecycle rule is configured for Amazon S3 buckets.

- **Control objective:** Optimize costs, Improve availability
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.6 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.6 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.6 example templates

Explanation

Configuring lifecycle rules on your Amazon S3 bucket defines actions that you want S3 to take during an object's lifetime. For example, you can transition objects to another storage class, archive them, or delete them after a specified period of time.

Remediation for rule failure

Configure at least one active lifecycle rule in `LifecycleConfiguration.Rules` by setting `Status` on a rule to `Enabled`.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket - Example

Amazon S3 bucket configured with an active lifecycle rule. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "LifecycleConfiguration": {
                "Rules": [
                    {
                        "Status": "Enabled",
                        "ExpirationInDays": 1,
                        "Id": "FirstRule"
                    }
                ]
            }
        }
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    LifecycleConfiguration:
      Rules:
        - Status: Enabled
          ExpirationInDays: 1
```

```
          Id: FirstRule
```

CT.S3.PR.6 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_lifecycle_policy_check
#
# Description:
#   This control checks whether a lifecycle rule is configured for Amazon S3 buckets.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'LifecycleConfiguration.Rules' has not been been provided or has been
 provided where 'Rules' is an
#             empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: The S3 bucket has versioning enabled (VersioningConfiguration.Status is
 set to 'Enabled')
```

```
#       And: 'LifecycleConfiguration.Rules' has been been provided as a non-empty list
#       And: There are no 'Rules' with 'Status' set to 'Enabled' present in the
 'LifecycleConfiguration'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: The S3 bucket has versioning enabled (VersioningConfiguration.Status is
 set to 'Enabled')
#       And: 'LifecycleConfiguration.Rules' has been been provided as a non-empty list
#       And: There is at least one entry in 'LifecycleConfiguration.Rules' with
 'Status' set to 'Enabled'
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_lifecycle_policy_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.6]: Require an Amazon S3 bucket to have lifecycle policies configured
        [FIX]: Configure at least one active lifecycle rule in
 'LifecycleConfiguration.Rules' by setting 'Status' on a rule to 'Enabled'.
        >>
}

rule s3_lifecycle_policy_check when is_cfn_hook(%INPUT_DOCUMENT, %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.6]: Require an Amazon S3 bucket to have lifecycle policies configured
        [FIX]: Configure at least one active lifecycle rule in
 'LifecycleConfiguration.Rules' by setting 'Status' on a rule to 'Enabled'.
```

```
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
        # Scenario 2
        LifecycleConfiguration exists
        LifecycleConfiguration is_struct

        LifecycleConfiguration {
            # Scenario 3 and 4
            Rules exists
            Rules is_list
            Rules not empty

            some Rules[*] {
                Status exists
                Status == "Enabled"
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.6 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      LifecycleConfiguration:
        Rules:
        - Status: Enabled
          ExpirationInDays: 1
          Id: FirstRule
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      LifecycleConfiguration:
        Rules:
        - Status: Disabled
          ExpirationInDays: 1
          Id: FirstRule
```

[CT.S3.PR.8] Require that Amazon S3 bucket requests use Secure Socket Layer

This control checks whether Amazon S3 bucket policies require requests to use Secure Socket Layer (SSL).

- **Control objective:** Encrypt data in transit

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::BucketPolicy`
- **CloudFormation guard rule:** CT.S3.PR.8 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.8 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.8 example templates

Explanation

Amazon S3 buckets should have policies that require all requests (`Action: S3:*`) to accept transmission of data over HTTPS in the S3 resource policy only, as indicated by the condition key `aws:SecureTransport`.

Remediation for rule failure

Configure an Amazon S3 bucket policy statement that denies access to all principals and actions for the S3 bucket and bucket objects when a secure transport protocol is not in use.

The examples that follow show how to implement this remediation.

Amazon S3 Bucket Policy - Example

Amazon S3 bucket policy configured to deny all access to the bucket and bucket objects when transmission of data is not over HTTPS. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3BucketPolicy": {
        "Type": "AWS::S3::BucketPolicy",
        "Properties": {
            "Bucket": {
                "Ref": "S3Bucket"
            },
```

```
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Deny",
                        "Action": "s3:*",
                        "Resource": [
                            {
                                "Fn::GetAtt": [
                                    "S3Bucket",
                                    "Arn"
                                ]
                            },
                            {
                                "Fn::Join": [
                                    "",
                                    [
                                        {
                                            "Fn::GetAtt": [
                                                "S3Bucket",
                                                "Arn"
                                            ]
                                        },
                                        "/*"
                                    ]
                                ]
                            }
                        ],
                        "Principal": "*",
                        "Condition": {
                            "Bool": {
                                "aws:SecureTransport": "false"
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
S3BucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    Bucket: !Ref 'S3Bucket'
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Deny
          Action: s3:*
          Resource:
            - !GetAtt 'S3Bucket.Arn'
            - !Join
              - ''
              - - !GetAtt 'S3Bucket.Arn'
                - /*
          Principal: '*'
          Condition:
            Bool:
              aws:SecureTransport: 'false'
```

CT.S3.PR.8 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_policy_ssl_requests_only_check
#
# Description:
#   This control checks whether Amazon S3 bucket policies require requests to use
 Secure Socket Layer (SSL).
#
# Reports on:
#   AWS::S3::BucketPolicy
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
```

```
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any S3 bucket policies
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket policy
#       And: 'Policydocument' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket policy
#       And: 'Policydocument' does not include a statement that denies Principal  ('*',
 AWS: '*')
#            all Actions ('s3:*', '*') over resource ('*' or bucketArn,
 bucketObjectArn) when the condition
#            "aws:SecureTransport" is "false"
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket policy
#       And: 'Policydocument' includes a statement that denies Principal  ('*', AWS:
 '*')
#            all Actions ('s3:*', '*') over resource ('*' or bucketArn,
 bucketObjectArn) when the condition
#            "aws:SecureTransport" is "false"
#      Then: PASS

#
# Constants
#
let S3_BUCKET_POLICY_TYPE = "AWS::S3::BucketPolicy"
let INPUT_DOCUMENT = this

let S3_BUCKET_ARN_PATTERN = /^arn:aws[a-z0-9\-]*:s3:::[a-z0-9][a-z0-9\.-]*[a-z0-9]$/
let S3_BUCKET_OBJECT_ARN_PATTERN = /^arn:aws[a-z0-9\-]*:s3:::[a-z0-9][a-z0-9\.-]*[a-
z0-9]\/\*$/

#
# Assignments
```

```
#
let s3_bucket_policies = Resources.*[ Type == %S3_BUCKET_POLICY_TYPE ]

#
# Primary Rules
#
rule s3_bucket_policy_ssl_requests_only_check when is_cfn_template(%INPUT_DOCUMENT)
                                                    %s3_bucket_policies not empty {
    check(%s3_bucket_policies.Properties)
        <<
        [CT.S3.PR.8]: Require that Amazon S3 buckets request to use Secure Socket Layer
            [FIX]: Configure an Amazon S3 bucket policy statement that denies access
 to all principals and actions for the S3 bucket and bucket objects when a secure
 transport protocol is not in use.
        >>
}

rule s3_bucket_policy_ssl_requests_only_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_POLICY_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_POLICY_TYPE.resourceProperties)
        <<
        [CT.S3.PR.8]: Require that Amazon S3 buckets request to use Secure Socket Layer
            [FIX]: Configure an Amazon S3 bucket policy statement that denies access
 to all principals and actions for the S3 bucket and bucket objects when a secure
 transport protocol is not in use.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket_policy) {
    %s3_bucket_policy {
        # Scenario 2
        PolicyDocument exists
        PolicyDocument is_struct

        PolicyDocument {
            Statement exists
            Statement is_list or
            Statement is_struct

            #Scenario 3 and 4
            some Statement[*] {
```

```
                check_statement_ssl_requests_only(this)
            }
        }
    }
}

rule check_statement_ssl_requests_only(statement) {
    %statement{
        check_all_required_statement_properties(this)

        Effect == "Deny"
        Action[*] in ["s3:*", "*"]

        Principal == "*" or
        Principal {
            AWS exists
            AWS == "*"
        }

        Resource[*] == "*" or
        check_resource_for_bucket_arns(Resource) or
        check_resource_for_bucket_arn_refs(Resource)

        Condition is_struct
        Condition == {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        }

    }
}

rule check_all_required_statement_properties(statement) {
    %statement {
        Effect exists
        Action exists
        Principal exists
        Condition exists
        Resource exists
    }
}

rule check_resource_for_bucket_arns(resource) {
```

```
    %resource {
        this is_list
        this not empty
        some this[*] == %S3_BUCKET_ARN_PATTERN
        some this[*] == %S3_BUCKET_OBJECT_ARN_PATTERN
    }
}

rule check_resource_for_bucket_arn_refs(resource) {
    %resource {
        this is_list
        this not empty
        some this[*] {
            check_local_bucket_arn_reference(%INPUT_DOCUMENT, this, "AWS::S3::Bucket")
        }
        some this[*] {
            check_local_bucket_object_arn_reference(%INPUT_DOCUMENT, this,
 "AWS::S3::Bucket")
        }
    }
}

rule check_local_bucket_arn_reference(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            check_get_att_bucket_arn(this)
        }
    }
}

rule check_local_bucket_object_arn_reference(doc, reference_properties,
 referenced_resource_type) {
    %reference_properties {
        'Fn::Join' {
            this is_list
            this not empty
            this[1][0] {
                'Fn::GetAtt' {
                    check_get_att_bucket_arn(this)
                }
            }
            this[1][1] == "/*"
        }
```

```
    }
}

rule check_get_att_bucket_arn(get_att){
    %get_att {
        this is_list
        this not empty
        this[1] == "Arn"
        query_for_resource(%doc, this[0], %referenced_resource_type)
            <<Local Stack reference was invalid>>
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.S3.PR.8 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
  S3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: S3Bucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Deny
          Action: s3:*
          Resource:
          - Fn::GetAtt:
            - S3Bucket
            - Arn
          - Fn::Join:
            - ''
            - - Fn::GetAtt:
                - S3Bucket
                - Arn
              - /*
          Principal: '*'
          Condition:
            Bool:
              aws:SecureTransport: 'false'
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  S3Bucket:
```

```
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
  S3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: S3Bucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Action: s3:*
          Resource:
          - Fn::GetAtt:
            - S3Bucket
            - Arn
          - Fn::Join:
            - ''
            - - Fn::GetAtt:
                - S3Bucket
                - Arn
              - /*
          Principal:
            AWS:
            - Ref: AWS::AccountId
          Condition:
            Bool:
              aws:SecureTransport: 'false'
```

[CT.S3.PR.9] Require that an Amazon S3 bucket has S3 Object Lock activated

This control checks whether an Amazon Simple Storage Service (Amazon S3) bucket has been configured to use S3 Object Lock.

- **Control objective:** Protect data integrity
- **Implementation:** CloudFormation guard rule

- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.9 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.9 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.S3.PR.9 example templates

Explanation

S3 Object Lock allows you to store objects using a write-once-read-many (WORM) model. Object Lock can help prevent objects from being deleted or overwritten for a fixed amount of time, or indefinitely. You can use S3 Object Lock to meet regulatory requirements that require WORM storage, or to add an extra layer of protection against object changes and deletion.

Usage considerations

- When you create an Amazon S3 bucket with object lock activated, S3 automatically enables versioning for the bucket.

Remediation for rule failure

Set `ObjectLockEnabled` to `true`.

The examples that follow show how to implement this remediation.

S3 bucket - Example

An Amazon S3 bucket configured with S3 object lock enabled. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "ObjectLockEnabled": true
        }
    }
}
```

YAML example

```
Bucket:
  Type: AWS::S3::Bucket
  Properties:
    ObjectLockEnabled: true
```

CT.S3.PR.9 rule specification

```
# ####################################
##       Rule Specification        ##
######################################
#
# Rule Identifier:
#   s3_bucket_object_lock_enabled_check
#
# Description:
#   This control checks whether an Amazon Simple Storage Service (Amazon S3) bucket has
 been configured to use S3 Object Lock.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
```

```
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ObjectLockEnabled' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ObjectLockEnabled' has been provided and set to a value other than
 bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ObjectLockEnabled' has been provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_bucket_object_lock_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.9]: Require that an Amazon S3 bucket has S3 Object Lock activated
        [FIX]: Set 'ObjectLockEnabled' to 'true'.
        >>
}
```

```
rule s3_bucket_object_lock_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(this.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.9]: Require that an Amazon S3 bucket has S3 Object Lock activated
        [FIX]: Set 'ObjectLockEnabled' to 'true'.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
       # Scenario 2
       ObjectLockEnabled exists
       # Scenarios 3 and 4
       ObjectLockEnabled == true
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.9 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      ObjectLockEnabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      ObjectLockEnabled: false
```

[CT.S3.PR.10] Require an Amazon S3 bucket to have server-side encryption configured using an AWS KMS key

This control checks whether default server-side encryption is enabled on an Amazon S3 bucket using AWS KMS.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`
- **CloudFormation guard rule:** CT.S3.PR.10 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.10 rule specification

- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.S3.PR.10 example templates

Explanation

Server-side encryption (SSE) is the encryption of data at its destination by the application or service that receives the data. Unless you specify otherwise, Amazon S3 buckets use SSE-S3 by default to encrypt objects. However, for added control, you can choose to configure buckets to use server-side encryption with AWS KMS keys (SSE-KMS) instead. Amazon S3 encrypts your data at the object level as it writes data to disks in AWS data centers, and then decrypts the data for you, when you require access to it.

Remediation for rule failure

Set an encryption rule in `BucketEncryption.ServerSideEncryptionConfiguration` with a `ServerSideEncryptionByDefault.SSEAlgorithm` configuration of `aws:kms` or `aws:kms:dsse`

The examples that follow show how to implement this remediation.

S3 Bucket - Example

An Amazon S3 bucket configured with AWS Key Management Service (AWS KMS) (SSE-KMS) default server-side encryption. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "ServerSideEncryptionByDefault": {
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            }
        }
```

```
    }
}
```

YAML example

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketEncryption:
      ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            SSEAlgorithm: aws:kms
```

CT.S3.PR.10 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_default_encryption_kms_check
#
# Description:
#   This control checks whether default server-side encryption is enabled on an Amazon
 S3 bucket using AWS KMS.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' has not been
 provided
#            or provided as an empty list
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' has been
 provided as
#            a non empty list
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' does not contain
 an
#            encryption rule with a 'ServerSideEncryptionByDefault' configuration
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' has been
 provided as
#            a non empty list
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' contains an
 encryption
#            rule with a 'ServerSideEncryptionByDefault' configuration
#       And: For an encryption rule, 'SSEAlgorithm' in 'ServerSideEncryptionByDefault'
 is not
#            not provided or has been provided and set to an SSE Algorithm other than
 'aws:kms'
#            or 'aws:kms:dsse'
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' has been
 provided as
#            a non empty list
#       And: 'ServerSideEncryptionConfiguration' in 'BucketEncryption' contains an
 encryption rule with
#            a 'ServerSideEncryptionByDefault' configuration
```

```
#       And: For all encryption rules, 'SSEAlgorithm' in
 'ServerSideEncryptionByDefault' is provided
#            and set to 'aws:kms' or 'aws:kms:dsse'
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let AUTHORIZED_SSE_ALGORITHMS = [ "aws:kms", "aws:kms:dsse" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_bucket_default_encryption_kms_check when is_cfn_template(%INPUT_DOCUMENT)
                                                %s3_buckets not empty {
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.10]: Require an Amazon S3 bucket to have server-side encryption
 configured using an AWS KMS key
        [FIX]: Set an encryption rule in
 'BucketEncryption.ServerSideEncryptionConfiguration' with a
 'ServerSideEncryptionByDefault.SSEAlgorithm' configuration of 'aws:kms' or
 'aws:kms:dsse'
        >>
}

rule s3_bucket_default_encryption_kms_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.10]: Require an Amazon S3 bucket to have server-side encryption
 configured using an AWS KMS key
        [FIX]: Set an encryption rule in
 'BucketEncryption.ServerSideEncryptionConfiguration' with a
 'ServerSideEncryptionByDefault.SSEAlgorithm' configuration of 'aws:kms' or
 'aws:kms:dsse'
        >>
```

```
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
        # Scenario 2
        BucketEncryption exists
        BucketEncryption is_struct

        BucketEncryption {
            ServerSideEncryptionConfiguration exists
            ServerSideEncryptionConfiguration is_list
            ServerSideEncryptionConfiguration not empty

            # Scenario 3, 4 and 5
            ServerSideEncryptionConfiguration[*] {
                ServerSideEncryptionByDefault exists
                ServerSideEncryptionByDefault is_struct

                ServerSideEncryptionByDefault {
                    SSEAlgorithm exists
                    SSEAlgorithm in %AUTHORIZED_SSE_ALGORITHMS
                }
            }
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.10 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketEncryption:
        ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            SSEAlgorithm: aws:kms
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
```

[CT.S3.PR.11] Require an Amazon S3 bucket to have versioning enabled

This control checks whether an Amazon Simple Storage Service (Amazon S3) bucket has versioning enabled.

- **Control objective:** Improve availability
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::Bucket`

- **CloudFormation guard rule:** CT.S3.PR.11 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.11 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.S3.PR.11 example templates

Explanation

Versioning keeps multiple variants of an object in the same Amazon S3 bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your S3 bucket. With versioning, you can recover more easily from unintended user actions and application failures.

Usage considerations

- If you have an unversioned bucket with an object expiration lifecycle configuration, and if you want to maintain the same permanent delete behavior when you enable versioning, you must set an expiration configuration for noncurrent objects. The noncurrent expiration configuration lifecycle manages deletion of noncurrent object versions in the version-enabled bucket. (A version-enabled bucket maintains one current, and zero or more noncurrent, object versions.)

Remediation for rule failure

Set the `Status` in `VersioningConfiguration` to Enabled.

The examples that follow show how to implement this remediation.

S3 Bucket - Example One

An Amazon S3 bucket with versioning enabled. The example is shown in JSON and in YAML.

JSON example

```
{
```

```
    "Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    }
}
```

YAML example

```
Bucket:
  Type: AWS::S3::Bucket
  Properties:
    VersioningConfiguration:
      Status: Enabled
```

CT.S3.PR.11 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   s3_bucket_versioning_enabled_check
#
# Description:
#   This control checks whether an Amazon Simple Storage Service (Amazon S3) bucket has
 versioning enabled.
#
# Reports on:
#   AWS::S3::Bucket
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
```

```
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any S3 bucket resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'VersioningConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'VersioningConfiguration' has been provided
#       And: 'Status' in 'VersioningConfiguration' has not been provided or has been
 provided
#            and set to a value other than 'Enabled'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an S3 bucket resource
#       And: 'VersioningConfiguration' has been provided
#       And: 'Status' in 'VersioningConfiguration' has been provided and set to
 'Enabled'
#      Then: PASS

#
# Constants
#
let S3_BUCKET_TYPE = "AWS::S3::Bucket"
let INPUT_DOCUMENT = this

#
# Assignments
#
let s3_buckets = Resources.*[ Type == %S3_BUCKET_TYPE ]

#
# Primary Rules
#
rule s3_bucket_versioning_enabled_check when is_cfn_template(%INPUT_DOCUMENT)
                                             %s3_buckets not empty {
```

```
    check(%s3_buckets.Properties)
        <<
        [CT.S3.PR.11]: Require an Amazon S3 bucket to have versioning enabled
        [FIX]: Set the 'Status' in 'VersioningConfiguration' to Enabled.
        >>
}

rule s3_bucket_versioning_enabled_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_BUCKET_TYPE) {
    check(%INPUT_DOCUMENT.%S3_BUCKET_TYPE.resourceProperties)
        <<
        [CT.S3.PR.11]: Require an Amazon S3 bucket to have versioning enabled
        [FIX]: Set the 'Status' in 'VersioningConfiguration' to Enabled.
        >>
}

#
# Parameterized Rules
#
rule check(s3_bucket) {
    %s3_bucket {
        # Scenario 2
        VersioningConfiguration exists

        # Scenarios 3 and 4
        VersioningConfiguration is_struct
        VersioningConfiguration {
            Status exists
            Status == "Enabled"
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
```

```
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.11 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
```

[CT.S3.PR.12] Require an Amazon S3 access point to have a Block Public Access (BPA) configuration with all options set to true

This control checks whether an Amazon S3 access point has been configured with a Block Public Access (BPA) configuration that has all options set to true.

- **Control objective:** Enforce least privilege

- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::S3::AccessPoint`
- **CloudFormation guard rule:** CT.S3.PR.12 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.S3.PR.12 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.S3.PR.12 example templates

Explanation

Amazon S3 access points support independent access settings that allow each access point to block public access. When you create an access point, you can specify block public access settings that apply to that access point. For any request made through an access point, Amazon S3 evaluates the block public access settings for that access point, the underlying bucket, and the bucket owner's account. If any of these settings indicate that the request should be blocked, Amazon S3 rejects the request.

Usage considerations

- This control is incompatible with Amazon S3 access points that require a public access configuration.
- Amazon S3 currently doesnt `support changing an access points` block public access settings after the access point is created.
- Adding an Amazon S3 access point to a bucket doesn't change the bucket's behavior when you access the bucket directly through the bucket's name or Amazon Resource Name (ARN). See Configuring IAM policies for using access points in the *Amazon S3 User Guide* for informaiton on options to configure the underlying bucket's policy for use with Amazon S3 access points.

Remediation for rule failure

In the PublicAccessBlockConfiguration field, set the values of BlockPublicAcls, BlockPublicPolicy, IgnorePublicAcls, and RestrictPublicBuckets to true, or omit the PublicAccessBlockConfiguration field to adopt the default value of true for these properties.

The examples that follow show how to implement this remediation.

Amazon S3 Access Point - Example

An Amazon S3 access point with a Block Public Access configuration that ensures public access requests by means of the access point are rejected. The example is shown in JSON and in YAML.

JSON example

```
{
    "S3AccessPoint": {
        "Type": "AWS::S3::AccessPoint",
        "Properties": {
            "Bucket": "sample-bucket",
            "Name": "sample-access-point",
            "Policy": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Action": [
                            "s3:GetObject",
                            "s3:PutObject"
                        ],
                        "Effect": "Allow",
                        "Resource": [
                            {
                                "Fn::Sub": "arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/sample-access-point/object/*"
                            }
                        ],
                        "Principal": {
                            "AWS": "arn:aws:iam::123456789012:role/SampleRole"
                        }
                    }
                ]
            },
            "PublicAccessBlockConfiguration": {
```

```
                "BlockPublicAcls": true,
                "BlockPublicPolicy": true,
                "IgnorePublicAcls": true,
                "RestrictPublicBuckets": true
            }
        }
    }
}
```

YAML example

```
S3AccessPoint:
  Type: AWS::S3::AccessPoint
  Properties:
    Bucket: sample-bucket
    Name: sample-access-point
    Policy:
      Version: '2012-10-17       '
      Statement:
        - Action:
            - s3:GetObject
            - s3:PutObject
          Effect: Allow
          Resource:
            - !Sub 'arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/sample-access-point/object/*'
          Principal:
            AWS: arn:aws:iam::123456789012:role/SampleRole
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
```

CT.S3.PR.12 rule specification

```
# ####################################
##       Rule Specification        ##
```

```
####################################
#
# Rule Identifier:
#   s3_access_point_public_access_prohibited_check
#
# Description:
#   This control checks whether an S3Amazon S3access point has been configured with a
 Block Public Access (BPA) configuration that has all options set to true.
#
# Reports on:
#   AWS::S3::AccessPoint
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any Amazon S3 access point resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 access point resource
#       And: 'PublicAccessBlockConfiguration' has been provided as an empty struct
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 access point resource
#       And: 'PublicAccessBlockConfiguration' has been provided
#       And: In 'PublicAccessBlockConfiguration', one or more of 'BlockPublicAcls',
#            'BlockPublicPolicy', 'IgnorePublicAcls' or 'RestrictPublicBuckets' have
 been
#            provided and set to a value other than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an Amazon S3 access point resource
#       And: 'PublicAccessBlockConfiguration' has not been provided
#      Then: PASS
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an Amazon S3 access point resource
#       And: 'PublicAccessBlockConfiguration' has been provided
#       And: In 'PublicAccessBlockConfiguration', 'BlockPublicAcls',
 'BlockPublicPolicy',
#            'IgnorePublicAcls' or 'RestrictPublicBuckets' have all been provided and
#            set to bool(true)
#      Then: PASS

#
# Constants
#
let INPUT_DOCUMENT = this
let S3_ACCESS_POINT_TYPE = "AWS::S3::AccessPoint"

#
# Assignments
#
let s3_access_points = Resources.*[ Type == %S3_ACCESS_POINT_TYPE ]

#
# Primary Rules
#
rule s3_access_point_public_access_prohibited_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %s3_access_points not empty {
    check(%s3_access_points.Properties)
        <<
        [CT.S3.PR.12]: Require an Amazon S3 access point to have a Block Public Access
 (BPA) configuration with all options set to true
        [FIX]: In the PublicAccessBlockConfiguration field, set the values of
 BlockPublicAcls, BlockPublicPolicy, IgnorePublicAcls, and RestrictPublicBuckets to
 true, or
        omit the PublicAccessBlockConfiguration field to adopt the default value of
 true for these properties.
        >>
}

rule s3_access_point_public_access_prohibited_check when is_cfn_hook(%INPUT_DOCUMENT,
 %S3_ACCESS_POINT_TYPE) {
    check(%INPUT_DOCUMENT.%S3_ACCESS_POINT_TYPE.resourceProperties)
        <<
        [CT.S3.PR.12]: Require an Amazon S3 access point to have a Block Public Access
 (BPA) configuration with all options set to true
```

```
        [FIX]: In the PublicAccessBlockConfiguration field, set the values of
 BlockPublicAcls, BlockPublicPolicy, IgnorePublicAcls, and RestrictPublicBuckets to
 true, or omit the PublicAccessBlockConfiguration field to adopt the default value of
 true for these properties.
        >>
}

#
# Parameterized Rules
#
rule check(s3_access_point) {
    %s3_access_point {
        # Scenarios 2 and 4
        PublicAccessBlockConfiguration not exists or
        # Scenarios 3 and 5
        check_bpa_configuration(this)

    }
}

rule check_bpa_configuration(s3_access_point) {
    %s3_access_point {
        PublicAccessBlockConfiguration is_struct

        PublicAccessBlockConfiguration {
            BlockPublicAcls exists
            BlockPublicPolicy exists
            IgnorePublicAcls exists
            RestrictPublicBuckets exists

            BlockPublicAcls == true
            BlockPublicPolicy == true
            IgnorePublicAcls == true
            RestrictPublicBuckets == true
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
```

```
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.S3.PR.12 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
  S3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: S3Bucket
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: '*'
          Effect: Allow
          Resource:
          - Fn::GetAtt:
            - S3Bucket
            - Arn
          - Fn::Join:
            - ''
            - - Fn::GetAtt:
                - S3Bucket
                - Arn
              - /*
          Principal:
            AWS: '*'
          Condition:
```

```
            StringEquals:
              s3:DataAccessPointAccount:
                Ref: AWS::AccountId
  AccessPointRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: GetObjectPermissions
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - s3:GetObject
            Resource:
            - Fn::Sub: arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/example-access-point/object/*
  S3AccessPoint:
    Type: AWS::S3::AccessPoint
    Properties:
      Bucket:
        Ref: S3Bucket
      Name: example-access-point
      Policy:
        Version: '2012-10-17       '
        Statement:
        - Action:
          - s3:GetObject
          - s3:PutObject
          Effect: Allow
          Resource:
          - Fn::Sub: arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/example-access-point/object/*
          Principal:
            AWS:
```

```
              Fn::GetAtt:
              - AccessPointRole
              - Arn
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
  S3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: S3Bucket
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: '*'
          Effect: Allow
          Resource:
          - Fn::GetAtt:
            - S3Bucket
            - Arn
          - Fn::Join:
            - ''
            - - Fn::GetAtt:
                - S3Bucket
                - Arn
              - /*
          Principal:
            AWS: '*'
          Condition:
            StringEquals:
              s3:DataAccessPointAccount:
```

```
                Ref: AWS::AccountId
  AccessPointRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            AWS:
              Ref: AWS::AccountId
          Action: sts:AssumeRole
      Path: /
      Policies:
      - PolicyName: GetObjectPermissions
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - s3:GetObject
            Resource:
            - Fn::Sub: arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/example-access-point/object/*
  S3AccessPoint:
    Type: AWS::S3::AccessPoint
    Properties:
      Bucket:
        Ref: S3Bucket
      Name: example-access-point
      Policy:
        Version: '2012-10-17       '
        Statement:
        - Action:
          - s3:GetObject
          - s3:PutObject
          Effect: Allow
          Resource:
          - Fn::Sub: arn:${AWS::Partition}:s3:${AWS::Region}:
${AWS::AccountId}:accesspoint/example-access-point/object/*
          Principal:
            AWS:
              Fn::GetAtt:
              - AccessPointRole
```

```
            - Arn
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
```

Amazon SageMaker AI controls

Topics

- [CT.SAGEMAKER.PR.1] Require an Amazon SageMaker AI notebook instance to prevent direct internet access
- [CT.SAGEMAKER.PR.2] Require Amazon SageMaker AI notebook instances to be deployed within a custom Amazon VPC
- [CT.SAGEMAKER.PR.3] Require Amazon SageMaker AI notebook instances to have root access disallowed

[CT.SAGEMAKER.PR.1] Require an Amazon SageMaker AI notebook instance to prevent direct internet access

This control checks that direct internet access is not allowed for an Amazon SageMaker AI notebook instance.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::SageMaker::NotebookInstance`
- **CloudFormation guard rule:** CT.SAGEMAKER.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.SAGEMAKER.PR.1 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.SAGEMAKER.PR.1 example templates

Explanation

When you configure your SageMaker AI notebook instance without a VPC, direct internet access is allowed for your instance, by default. Instead, you should configure your instance with a VPC and change the default setting to `Disable - Access the internet through a VPC`.

To train or host models from a notebook, you require internet access. To set up internet access, make sure that your VPC has a NAT gateway, and that your security group allows outbound connections.

Ensure that access to your SageMaker configuration is limited to authorized users. Restrict users' IAM permissions for modifying SageMaker settings and resources.

Usage considerations

- To set up outbound internet access for Amazon SageMaker AI notebook instances when this control is activated - First, associate the notebook instance with a private subnet that has access to the internet, through a default route to a NAT gateway instance. Also, be sure that the security groups assigned to the notebook instance, and the network access control list (NACL) of the private subnet, allow outbound traffic to the internet.

Remediation for rule failure

Set `DirectInternetAccess` to `Disabled` and provide a `SubnetId` and one or more `SecurityGroupIds`.

The examples that follow show how to implement this remediation.

Amazon SageMaker AI Notebook Instance - Example

Amazon SageMaker AI notebook instance configured with direct internet access deactivated. The example is shown in JSON and in YAML.

JSON example

```
{
    "SageMakerNoteBookInstance": {
        "Type": "AWS::SageMaker::NotebookInstance",
        "Properties": {
            "InstanceType": "ml.t2.medium",
            "RoleArn": {
                "Fn::GetAtt": [
                    "ExecutionRole",
                    "Arn"
                ]
            },
            "DirectInternetAccess": "Disabled",
            "SubnetId": {
                "Ref": "Subnet"
            },
            "SecurityGroupIds": [
                {
                    "Fn::GetAtt": [
                        "SecurityGroup",
                        "GroupId"
                    ]
                }
            ]
        }
    }
}
```

YAML example

```
SageMakerNoteBookInstance:
  Type: AWS::SageMaker::NotebookInstance
  Properties:
    InstanceType: ml.t2.medium
    RoleArn: !GetAtt 'ExecutionRole.Arn'
    DirectInternetAccess: Disabled
    SubnetId: !Ref 'Subnet'
    SecurityGroupIds:
      - !GetAtt 'SecurityGroup.GroupId'
```

CT.SAGEMAKER.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   sagemaker_notebook_no_direct_internet_access_check
#
# Description:
#   This control checks that direct internet access is not allowed for an Amazon
 SageMaker AI notebook instance.
#
# Reports on:
#   AWS::SageMaker::NotebookInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any SageMaker AI notebook instance
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SageMaker AI notebook instance resource
#       And: 'DirectInternetAccess' has not been provided on the SageMaker AI notebook
 instance resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SageMaker AI notebook instance resource
#       And: 'DirectInternetAccess' has been provided on the SageMaker AI notebook
 instance resource
#       And: 'DirectInternetAccess' is set to 'Enabled'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document contains an SageMaker AI notebook instance resource
#       And: 'DirectInternetAccess' has been provided on the SageMaker AI notebook
 instance resource
#       And: 'DirectInternetAccess' is set to 'Disabled'
#       And: 'SecurityGroupIds' have been provided as a non-empty list with non-empty
 strings or valid local references
#       And: 'SubnetId' has been provided as an empty string or non-valid local
 reference
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SageMaker AI notebook instance resource
#       And: 'DirectInternetAccess' has been provided on the SageMaker AI notebook
 instance resource
#       And: 'DirectInternetAccess' is set to 'Disabled'
#       And: 'SubnetId' has been provided as a non-empty string or valid local
 reference
#       And: 'SecurityGroupIds' have been provided as an empty list or a list that
 contains empty string values or
#            non-valid local references
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SageMaker AI notebook instance resource
#       And: 'DirectInternetAccess' has been provided on the SageMaker AI notebook
 instance resource
#       And: 'DirectInternetAccess' is set to 'Disabled'
#       And: 'SecurityGroupIds' have been provided as a non-empty list with non-empty
 strings or valid local references
#       And: 'SubnetId' has been provided as a non-empty string or valid local
 reference
#      Then: PASS

#
# Constants
#
let SAGEMAKER_NOTEBOOK_INSTANCE_TYPE = "AWS::SageMaker::NotebookInstance"
let INPUT_DOCUMENT = this

#
# Assignments
#
let sagemaker_notebook_instances = Resources.*[ Type ==
 %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE ]
```

```
#
# Primary Rules
#
rule sagemaker_notebook_no_direct_internet_access_check when is_cfn_template(this)

 %sagemaker_notebook_instances not empty {
    check(%sagemaker_notebook_instances.Properties)
        <<
        [CT.SAGEMAKER.PR.1]: Require an Amazon SageMaker AI notebook instance to
 prevent direct internet access
        [FIX]: Set 'DirectInternetAccess' to 'Disabled' and provide a 'SubnetId' and
 one or more 'SecurityGroupIds'.
        >>
}

rule sagemaker_notebook_no_direct_internet_access_check when
 is_cfn_hook(%INPUT_DOCUMENT, %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%SAGEMAKER_NOTEBOOK_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.SAGEMAKER.PR.1]: Require an Amazon SageMaker AI notebook instance to
 prevent direct internet access
        [FIX]: Set 'DirectInternetAccess' to 'Disabled' and provide a 'SubnetId' and
 one or more 'SecurityGroupIds'.
        >>
}

#
# Parameterized Rules
#
rule check(sagemaker_notebook_instances) {
    %sagemaker_notebook_instances {
        # Scenario 2
        DirectInternetAccess exists

        # Scenario 3
        DirectInternetAccess is_string
        DirectInternetAccess == "Disabled"

        # Scenario 4,5 and 6
        check_is_string_and_not_empty(SubnetId) or
        check_local_references(%INPUT_DOCUMENT, SubnetId, "AWS::EC2::Subnet")

        SecurityGroupIds exists
```

```
        SecurityGroupIds is_list
        SecurityGroupIds not empty

        SecurityGroupIds[*] {
            check_is_string_and_not_empty(this) or
            check_local_references(%INPUT_DOCUMENT, this, "AWS::EC2::SecurityGroup")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, SAGEMAKER_NOTEBOOK_INSTANCE_TYPE) {
    %doc.%SAGEMAKER_NOTEBOOK_INSTANCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
```

```
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.SAGEMAKER.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
      AvailabilityZone:
        Fn::Select:
        - 0
        - Fn::GetAZs: ''
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Notebook SG1
      VpcId:
        Ref: VPC
  ExecutionRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - sagemaker.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - Fn::Sub: arn:${AWS::Partition}:iam::aws:policy/AmazonSageMakerFullAccess
  SageMakerNoteBookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.medium
      RoleArn:
        Fn::GetAtt:
        - ExecutionRole
        - Arn
      DirectInternetAccess: Disabled
      SubnetId:
        Fn::GetAtt:
        - Subnet
        - SubnetId
      SecurityGroupIds:
      - Fn::GetAtt:
        - SecurityGroup
        - GroupId
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  ExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
```

```
        - Effect: Allow
          Principal:
            Service:
            - sagemaker.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      - Fn::Sub: arn:${AWS::Partition}:iam::aws:policy/AmazonSageMakerFullAccess
  SageMakerNoteBookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.medium
      RoleArn:
        Fn::GetAtt:
        - ExecutionRole
        - Arn
      DirectInternetAccess: Enabled
```

[CT.SAGEMAKER.PR.2] Require Amazon SageMaker AI notebook instances to be deployed within a custom Amazon VPC

This control checks whether an Amazon SageMaker AI notebook instance is configured to launch within a custom Amazon VPC.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::SageMaker::NotebookInstance`
- **CloudFormation guard rule:** CT.SAGEMAKER.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.SAGEMAKER.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.SAGEMAKER.PR.2 example templates

Explanation

As a best practice, we recommend that you keep your resources contained inside a VPC whenever possible, to ensure the secure network protection of your infrastructure.

Remediation for rule failure

Set `SubnetId` to the identifier of an Amazon EC2 subnet and set `SecurityGroupIds` to a list containing one or more EC2 security group identifiers.

The examples that follow show how to implement this remediation.

Amazon SageMaker AI Notebook Instance - Example

Amazon SageMaker AI notebook instance configured with Amazon VPC connectivity. The example is shown in JSON and in YAML.

JSON example

```
{
    "NotebookInstance": {
        "Type": "AWS::SageMaker::NotebookInstance",
        "Properties": {
            "InstanceType": "ml.t2.large",
            "RoleArn": {
                "Fn::GetAtt": [
                    "ExecutionRole",
                    "Arn"
                ]
            },
            "SubnetId": {
                "Fn::GetAtt": [
                    "Subnet",
                    "SubnetId"
                ]
            },
            "SecurityGroupIds": [
                {
                    "Fn::GetAtt": [
                        "SecurityGroup",
                        "GroupId"
                    ]
                }
```

```
            ]
        }
    }
}
```

YAML example

```
NotebookInstance:
  Type: AWS::SageMaker::NotebookInstance
  Properties:
    InstanceType: ml.t2.large
    RoleArn: !GetAtt 'ExecutionRole.Arn'
    SubnetId: !GetAtt 'Subnet.SubnetId'
    SecurityGroupIds:
      - !GetAtt 'SecurityGroup.GroupId'
```

CT.SAGEMAKER.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   sagemaker_notebook_instance_inside_vpc_check
#
# Description:
#   This control checks whether an Amazon SageMaker AI notebook instance is configured
 to launch within a custom Amazon VPC.
#
# Reports on:
#   AWS::SageMaker::NotebookInstance
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any SageMaker AI notebook instance
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'SubnetId' has not been provided or provided as an empty string or non-
valid local reference
#       And: 'SecurityGroupIds' has not been provided or provided as an empty list or a
 list that contains empty string
#            values or non-valid local references
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'SubnetId' has been provided as a non-empty string or valid local
 reference
#       And: 'SecurityGroupIds' has not been provided or provided as an empty list or a
 list that contains empty string
#            values or non-valid local references
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'SubnetId' has not been provided or provided as an empty string or non-
valid local reference
#       And: 'SecurityGroupIds' have been provided as a non-empty list containing non-
empty string values or
#            valid local references
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'SubnetId' has been provided as a non-empty string or valid local
 reference
#       And: 'SecurityGroupIds' have been provided as a list containing one or more
 non-empty string values or
#            valid local references
#      Then: PASS

#
# Constants
#
let SAGEMAKER_NOTEBOOK_INSTANCE_TYPE = "AWS::SageMaker::NotebookInstance"
```

```
let INPUT_DOCUMENT = this

#
# Assignments
#
let sagemaker_notebook_instances = Resources.*[ Type ==
 %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE ]

#
# Primary Rules
#
rule sagemaker_notebook_instance_inside_vpc_check when is_cfn_template(%INPUT_DOCUMENT)
                                                       %sagemaker_notebook_instances
 not empty {
    check(%sagemaker_notebook_instances.Properties)
        <<
        [CT.SAGEMAKER.PR.2]: Require Amazon SageMaker AI notebook instances to be
 deployed within a custom Amazon VPC
            [FIX]: Set 'SubnetId' to the identifier of an Amazon EC2 subnet and set
 'SecurityGroupIds' to a list containing one or more EC2 security group identifiers.
        >>
}

rule sagemaker_notebook_instance_inside_vpc_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%SAGEMAKER_NOTEBOOK_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.SAGEMAKER.PR.2]: Require Amazon SageMaker AI notebook instances to be
 deployed within a custom Amazon VPC
            [FIX]: Set 'SubnetId' to the identifier of an Amazon EC2 subnet and set
 'SecurityGroupIds' to a list containing one or more EC2 security group identifiers.
        >>
}

rule check(sagemaker_notebook_instance) {
    %sagemaker_notebook_instance {
        # Scenario 2
        SubnetId exists

        # Scenario 3, 4 and 5
        check_is_string_and_not_empty(SubnetId) or
        check_local_references(%INPUT_DOCUMENT, SubnetId, "AWS::EC2::Subnet")

        SecurityGroupIds exists
```

```
        SecurityGroupIds is_list
        SecurityGroupIds not empty

        SecurityGroupIds[*] {
            check_is_string_and_not_empty(this) or
            check_local_references(%INPUT_DOCUMENT, this, "AWS::EC2::SecurityGroup")
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
```

```
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.SAGEMAKER.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NotebookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.large
      RoleArn: example-role-arn
      SubnetId: example-subnet-id
      SecurityGroupIds:
        - example-sg-id
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NotebookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.large
      RoleArn: example-role-arn
```

[CT.SAGEMAKER.PR.3] Require Amazon SageMaker AI notebook instances to have root access disallowed

This control checks whether Amazon SageMaker AI notebook instances allow root access.

- **Control objective:** Enforce least privilege
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::SageMaker::NotebookInstance`
- **CloudFormation guard rule:** CT.SAGEMAKER.PR.3 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.SAGEMAKER.PR.3 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.SAGEMAKER.PR.3 example templates

Explanation

By default, when you create a notebook instance, users that log into that notebook instance have root access. Because users with root access have administrator privileges, users have access to edit all files on a notebook instance with root access enabled. In adherence to the principle of least privilege, for security reasons, we recommend that you restrict root access to instance resources whenever possible, to avoid unintentional over-provisioning of permissions.

Usage considerations

- Lifecycle configurations associated with an Amazon SageMaker AI notebook instance always run with root access, even if you turn off root access for users.

Remediation for rule failure

Set `RootAccess` to `Disabled`.

The examples that follow show how to implement this remediation.

Amazon SageMaker AI Notebook Instance - Example

Amazon SageMaker AI notebook instance configured with root access turned off. The example is shown in JSON and in YAML.

JSON example

```
{
    "NotebookInstance": {
        "Type": "AWS::SageMaker::NotebookInstance",
        "Properties": {
            "InstanceType": "ml.t2.large",
            "RoleArn": {
                "Fn::GetAtt": [
                    "ExecutionRole",
                    "Arn"
                ]
            },
            "RootAccess": "Disabled"
        }
    }
}
```

YAML example

```
NotebookInstance:
  Type: AWS::SageMaker::NotebookInstance
  Properties:
    InstanceType: ml.t2.large
    RoleArn: !GetAtt 'ExecutionRole.Arn'
    RootAccess: Disabled
```

CT.SAGEMAKER.PR.3 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
```

```
#
# Rule Identifier:
#   sagemaker_notebook_instance_root_access_check
#
# Description:
#   This control checks whether Amazon SageMaker AI notebook instances allow root
 access.
#
# Reports on:
#   AWS::SageMaker::NotebookInstance
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any SageMaker AI notebook instance
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'RootAccess' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'RootAccess' has been provided and is set to a value other than 'Disabled'
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a SageMaker AI notebook instance resource
#       And: 'RootAccess' has been provided and is set to 'Disabled'
#      Then: PASS

#
# Constants
#
let SAGEMAKER_NOTEBOOK_INSTANCE_TYPE = "AWS::SageMaker::NotebookInstance"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let sagemaker_notebook_instances = Resources.*[ Type ==
 %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE ]

#
# Primary Rules
#
rule sagemaker_notebook_instance_root_access_check when
 is_cfn_template(%INPUT_DOCUMENT)
                                                        %sagemaker_notebook_instances
 not empty {
    check(%sagemaker_notebook_instances.Properties)
        <<
        [CT.SAGEMAKER.PR.3]: Require Amazon SageMaker AI notebook instances to have
 root access disallowed
            [FIX]: Set 'RootAccess' to 'Disabled'.
        >>
}

rule sagemaker_notebook_instance_root_access_check when is_cfn_hook(%INPUT_DOCUMENT,
 %SAGEMAKER_NOTEBOOK_INSTANCE_TYPE) {
    check(%INPUT_DOCUMENT.%SAGEMAKER_NOTEBOOK_INSTANCE_TYPE.resourceProperties)
        <<
        [CT.SAGEMAKER.PR.3]: Require Amazon SageMaker AI notebook instances to have
 root access disallowed
            [FIX]: Set 'RootAccess' to 'Disabled'.
        >>
}

rule check(sagemaker_notebook_instance) {
    %sagemaker_notebook_instance {
        # Scenarios 2, 3 and 4
        RootAccess exists
        RootAccess == "Disabled"
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
```

```
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.SAGEMAKER.PR.3 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  NotebookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.large
      RoleArn: example-role-arn
      RootAccess: Disabled
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  NotebookInstance:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t2.large
      RoleArn: example-role-arn
      RootAccess: Enabled
```

Amazon Simple Queue Service (Amazon SQS) controls

Topics

- [CT.SQS.PR.1] Require any Amazon SQS queue to have a dead-letter queue configured
- [CT.SQS.PR.2] Require any Amazon SQS queue to have encryption at rest configured

[CT.SQS.PR.1] Require any Amazon SQS queue to have a dead-letter queue configured

This control checks whether an Amazon SQS queue is configured with a dead-letter queue.

- **Control objective:** Improve resiliency
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::SQS::Queue`
- **CloudFormation guard rule:** CT.SQS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.SQS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.SQS.PR.1 example templates

Explanation

The main task of a dead-letter queue is to handle the lifecycle of unconsumed messages. A dead-letter queue lets you set aside and isolate messages that cant be processed correctly, so you can determine why their processing didnt succeed.

Usage considerations

- This control applies only to Amazon SQS queues that are not configured as a dead-letter queue with a `RedriveAllowPolicy` property.

Remediation for rule failure

Create a `RedrivePolicy` with a `deadLetterTargetArn` value thats `set to the ARN of an Amazon SQS dead-letter queue. For Amazon SQS dead-letter queues, instead provide a redrive configuration in the` RedriveAllowPolicy' property.

The examples that follow show how to implement this remediation.

Amazon SQS Queue - Example

Amazon SQS queue configured to send messages to a dead-letter queue, if the messages can't be processed (consumed) successfully. The example is shown in JSON and in YAML.

JSON example

```
{
    "SQSQueue": {
        "Type": "AWS::SQS::Queue",
        "Properties": {
            "RedrivePolicy": {
                "deadLetterTargetArn": {
                    "Fn::GetAtt": [
                        "DLQQueue",
                        "Arn"
                    ]
                },
                "maxReceiveCount": 3
            }
        }
    }
}
```

YAML example

```
SQSQueue:
  Type: AWS::SQS::Queue
  Properties:
    RedrivePolicy:
      deadLetterTargetArn: !GetAtt 'DLQQueue.Arn'
      maxReceiveCount: 3
```

CT.SQS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#    sqs_dlq_check
#
# Description:
#   This control checks whether an Amazon SQS queue is configured with a dead-letter
 queue.
#
# Reports on:
#    AWS::SQS::Queue
#
# Evaluates:
#    CloudFormation, CloudFormation hook
#
# Rule Parameters:
#     None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any SQS queue resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'RedriveAllowPolicy' has been provided on the SQS queue
#       And: 'RedriveAllowPolicy.redrivePermission' is set to 'allowAll' or 'byQueue'
```

```
#      Then: SKIP
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'RedriveAllowPolicy' has not been provided on the SQS queue or
 'RedriveAllowPolicy.redrivePermission'
#            has been provided and is set to a value other than 'allowAll' or 'byQueue'
#       And: 'RedrivePolicy' has not been provided
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'RedriveAllowPolicy' has not been provided on the SQS queue or
 'RedriveAllowPolicy.redrivePermission'
#            has been provided and is set to a value other than 'allowAll' or 'byQueue'
#       And: 'RedrivePolicy' has been provided
#       And: 'RedrivePolicy.deadLetterTargetArn' has not been provided or has been
 provided as an empty string or
#            invalid local reference to an SQS queue
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'RedriveAllowPolicy' has not been provided on the SQS queue or
 'RedriveAllowPolicy.redrivePermission' has
#            been provided and is set to a value other than 'allowAll' or 'byQueue'
#       And: 'RedrivePolicy' has been provided
#       And: 'RedrivePolicy.deadLetterTargetArn' has been provided as a non-empty
 string or valid local reference to
#             an SQS queue
#      Then: PASS

#
# Constants
#
let SQS_QUEUE_TYPE = "AWS::SQS::Queue"
let INPUT_DOCUMENT = this
let DLQ_REDRIVE_PERMISSION = ["allowAll", "byQueue"]

#
# Assignments
#
let sqs_queues = Resources.*[ Type == %SQS_QUEUE_TYPE ]
```

```
#
# Primary Rules
#
rule sqs_dlq_check when is_cfn_template(%INPUT_DOCUMENT)
                        %sqs_queues not empty {
    check(%sqs_queues.Properties)
        <<
        [CT.SQS.PR.1]: Require any Amazon SQS queue to have a dead-letter queue
 configured
            [FIX]: Create a 'RedrivePolicy' with a 'deadLetterTargetArn' value that's
 set to the ARN of an Amazon SQS dead-letter queue. For Amazon SQS dead-letter queues,
 instead provide a redrive configuration in the 'RedriveAllowPolicy' property.
        >>
}

rule sqs_dlq_check when is_cfn_hook(%INPUT_DOCUMENT, %SQS_QUEUE_TYPE) {
    check(%INPUT_DOCUMENT.%SQS_QUEUE_TYPE.resourceProperties)
        <<
        [CT.SQS.PR.1]: Require any Amazon SQS queue to have a dead-letter queue
 configured
            [FIX]: Create a 'RedrivePolicy' with a 'deadLetterTargetArn' value that's
 set to the ARN of an Amazon SQS dead-letter queue. For Amazon SQS dead-letter queues,
 instead provide a redrive configuration in the 'RedriveAllowPolicy' property.
        >>
}

#
# Parameterized Rules
#
rule check(sqs_queues) {
    %sqs_queues [
        # Scenario 2
        RedriveAllowPolicy not exists or
        filter_is_not_dlq(this)
    ] {
        # Scenario 3
        RedrivePolicy exists
        RedrivePolicy is_struct

        # Scenario 4
        RedrivePolicy {
            deadLetterTargetArn exists

            check_is_string_and_not_empty(deadLetterTargetArn) or
```

```
            check_local_references(%INPUT_DOCUMENT, deadLetterTargetArn,
 %SQS_QUEUE_TYPE)
        }
    }
}

rule filter_is_not_dlq(sqs_queue) {
    RedriveAllowPolicy exists
    RedriveAllowPolicy is_struct

    RedriveAllowPolicy {
        redrivePermission exists
        redrivePermission not in %DLQ_REDRIVE_PERMISSION
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        "Fn::GetAtt" {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
```

```
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.SQS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  SQSQueue:
    Type: AWS::SQS::Queue
    Properties:
      RedrivePolicy:
        deadLetterTargetArn:
          Fn::GetAtt: [DLQQueue, Arn]
        maxReceiveCount: 3
  DLQQueue:
    Type: AWS::SQS::Queue
    Properties:
      RedriveAllowPolicy:
        redrivePermission: allowAll
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SQSQueue:
    Type: AWS::SQS::Queue
    Properties: {}
```

[CT.SQS.PR.2] Require any Amazon SQS queue to have encryption at rest configured

This control checks whether an Amazon SQS queue is encrypted at rest.

- **Control objective:** Encrypt data at rest
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::SQS::Queue`
- **CloudFormation guard rule:** CT.SQS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.SQS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.SQS.PR.2 example templates

Explanation

Server-side encryption (SSE) allows you to transmit sensitive data in encrypted queues. To protect the content of messages in queues, SSE uses KMS keys.

Remediation for rule failure

Set `SqsManagedSseEnabled` to `true` or set an AWS KMS key identifier in the `KmsMasterKeyId` property.

The examples that follow show how to implement this remediation.

Amazon SQS Queue - Example One

Amazon SQS queue configured to encrypt data at rest with server-side encryption enabled, by means of SQS managed encryption keys (SSE-SQS). The example is shown in JSON and in YAML.

JSON example

```
{
    "SQSQueue": {
        "Type": "AWS::SQS::Queue",
        "Properties": {
            "SqsManagedSseEnabled": true
        }
    }
}
```

YAML example

```
SQSQueue:
  Type: AWS::SQS::Queue
  Properties:
    SqsManagedSseEnabled: true
```

The examples that follow show how to implement this remediation.

Amazon SQS Queue - Example Two

Amazon SQS queue configured to encrypt data at rest with server-side encryption enabled, by means of AWS KMS (SSE-KMS). The example is shown in JSON and in YAML.

JSON example

```
{
    "SQSQueue": {
        "Type": "AWS::SQS::Queue",
        "Properties": {
            "KmsMasterKeyId": {
```

```
                "Ref": "KMSKey"
            }
        }
    }
}
```

YAML example

```
SQSQueue:
  Type: AWS::SQS::Queue
  Properties:
    KmsMasterKeyId: !Ref 'KMSKey'
```

CT.SQS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#    sqs_queue_encrypted_check
#
# Description:
#   This control checks whether an Amazon SQS queue is encrypted at rest.
#
# Reports on:
#    AWS::SQS::Queue
#
# Evaluates:
#    CloudFormation, CloudFormation hook
#
# Rule Parameters:
#     None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any SQS queue resources
```

```
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'KmsMasterKeyId' or 'SqsManagedSseEnabled' have not been provided on the
 SQS queue resource
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'KmsMasterKeyId' has not been provided
#       And: 'SqsManagedSseEnabled' has been provided and set to bool(false)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'KmsMasterKeyId' has been provided as an empty string or invalid local
 reference to a KMS key or alias
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'SqsManagedSseEnabled' is not provided or set to bool(false)
#       And: 'KmsMasterKeyId' is provided as a non-empty string or local reference to a
 KMS key
#      Then: PASS
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an SQS queue resource
#       And: 'KmsMasterKeyId' is not provided
#       And: 'SqsManagedSseEnabled' is provided and set to bool(true)
#      Then: PASS

#
# Constants
#
let SQS_QUEUE_TYPE = "AWS::SQS::Queue"
let INPUT_DOCUMENT = this
#
# Assignments
#
let sqs_queues = Resources.*[ Type == %SQS_QUEUE_TYPE ]

#
```

```
# Primary Rules
#
rule sqs_queue_encrypted_check when is_cfn_template(%INPUT_DOCUMENT)
                                    %sqs_queues not empty {
    check(%sqs_queues.Properties)
        <<
        [CT.SQS.PR.2]: Require any Amazon SQS queue to have encryption at rest
 configured
        [FIX]: Set 'SqsManagedSseEnabled' to 'true' or set an AWS KMS key identifier in
 the 'KmsMasterKeyId' property.
        >>
}

rule sqs_queue_encrypted_check when is_cfn_hook(%INPUT_DOCUMENT, %SQS_QUEUE_TYPE) {
    check(%INPUT_DOCUMENT.%SQS_QUEUE_TYPE.resourceProperties)
        <<
        [CT.SQS.PR.2]: Require any Amazon SQS queue to have encryption at rest
 configured
        [FIX]: Set 'SqsManagedSseEnabled' to 'true' or set an AWS KMS key identifier in
 the 'KmsMasterKeyId' property.
        >>
}

#
# Parameterized Rules
#
rule check(sqs_queue) {
    %sqs_queue{
        check_sse_enabled(this) or
        check_kms_valid(this)
    }
}

rule check_sse_enabled(sqs_queue) {
    # Scenario 2
    SqsManagedSseEnabled exists

    # Scenario 3, 6
    KmsMasterKeyId not exists
    SqsManagedSseEnabled == true
}

rule check_kms_valid(sqs_queue) {
    # Scenario 2
```

```
    KmsMasterKeyId exists

    # Scenario 4, 5
    check_is_string_and_not_empty(KmsMasterKeyId) or
    check_local_references(%INPUT_DOCUMENT, KmsMasterKeyId, "AWS::KMS::Key") or
    check_local_references(%INPUT_DOCUMENT, KmsMasterKeyId, "AWS::KMS::Alias")

    # Scenario 5
    SqsManagedSseEnabled not exists or
    SqsManagedSseEnabled == false
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        "Fn::GetAtt" {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}
```

```
rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.SQS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  KMSKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Version: 2012-10-17
        Id: key-default-1
        Statement:
        - Sid: Enable IAM User Permissions
          Effect: Allow
          Principal:
            AWS: '*'
          Action: 'kms:*'
          Resource: '*'
  SQSQueue:
    Type: AWS::SQS::Queue
    Properties:
      KmsMasterKeyId:
        Ref: KMSKey
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  SQSQueue:
    Type: AWS::SQS::Queue
    Properties: {}
```

AWS Step Functions controls

Topics

- [CT.STEPFUNCTIONS.PR.1] Require an AWS Step Functions state machine to have logging activated
- [CT.STEPFUNCTIONS.PR.2] Require an AWS Step Functions state machine to have AWS X-Ray tracing activated

[CT.STEPFUNCTIONS.PR.1] Require an AWS Step Functions state machine to have logging activated

This control checks whether an AWS Step Functions state machine has logging enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::StepFunctions::StateMachine`
- **CloudFormation guard rule:** CT.STEPFUNCTIONS.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.STEPFUNCTIONS.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.STEPFUNCTIONS.PR.1 example templates

Explanation

Defining a logging configuration for your state machines allows you to track their execution history and results. This configuration allows you to track failed events that occur on a state machine, and this insight into errors can assist you when you're troubleshooting issues.

Remediation for rule failure

In `LoggingConfiguration`, set `Level` to `ERROR` or `ALL`, and set `Destinations` to a list with one or more valid Amazon CloudWatch Logs log group ARNs.

The examples that follow show how to implement this remediation.

AWS Step Functions State Machine - Example

AWS Step Functions state machine configured to send logs to Amazon CloudWatch Logs. The example is shown in JSON and in YAML.

JSON example

```
{
    "StateMachine": {
        "Type": "AWS::StepFunctions::StateMachine",
        "Properties": {
            "StateMachineType": "STANDARD",
            "DefinitionString": "{\"StartAt\": \"Sample\",\"States\":
 {\"Sample\": {\"Type\": \"Task\", \"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:SampleFunction\", \"End\": true}}}",
            "RoleArn": {
                "Fn::GetAtt": [
                    "StepFunctionExecutionRole",
                    "Arn"
                ]
            },
            "LoggingConfiguration": {
                "Level": "ALL",
                "Destinations": [
                    {
                        "CloudWatchLogsLogGroup": {
                            "LogGroupArn": {
                                "Fn::GetAtt": [
                                    "LogGroup",
                                    "Arn"
                                ]
                            }
```

```
                        }
                    }
                ]
            }
        }
    }
}
```

YAML example

```
StateMachine:
  Type: AWS::StepFunctions::StateMachine
  Properties:
    StateMachineType: STANDARD
    DefinitionString: '{"StartAt": "Sample","States": {"Sample": {"Type": "Task",
      "Resource":"arn:aws:lambda:us-east-1:111122223333:function:SampleFunction",
      "End": true}}}'
    RoleArn: !GetAtt 'StepFunctionExecutionRole.Arn'
    LoggingConfiguration:
      Level: ALL
      Destinations:
        - CloudWatchLogsLogGroup:
            LogGroupArn: !GetAtt 'LogGroup.Arn'
```

CT.STEPFUNCTIONS.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   step_functions_state_machine_logging_enabled_check
#
# Description:
#   This control checks whether an AWS Step Functions state machine has logging
 enabled.
#
# Reports on:
```

```
#   AWS::StepFunctions::StateMachine
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any StepFunctions state machine
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'LoggingConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'LoggingConfiguration' has been provided
#       And: In 'LoggingConfiguration', 'Level' has not been provided or provided and
 set to a value other than
#            'ERROR' or 'ALL'
#       And: In 'LoggingConfiguration', 'Destinations' has not been provided or
 provided as an empty list or list
#            containing empty strings or non-valid local references
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'LoggingConfiguration' has been provided
#       And: In 'LoggingConfiguration', 'Level' has not been provided or provided and
 set to a value other than
#            'ERROR' or 'ALL'
#       And: In 'LoggingConfiguration', 'Destinations' has not been provided or
 provided as an empty list or list
#            containing empty strings or non-valid local references
#      Then: FAIL
#   Scenario: 5
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
```

```
#       And: 'LoggingConfiguration' has been provided
#       And: In 'LoggingConfiguration', 'Level' has not been provided or provided and
 set to a value other than
#            'ERROR' or 'ALL'
#       And: In 'LoggingConfiguration', 'Destinations' has been provided as a list
 containing non-empty strings or
#            valid local references
#      Then: FAIL
#   Scenario: 6
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'LoggingConfiguration' has been provided
#       And: In 'LoggingConfiguration', 'Level' has been provided and set to 'ERROR' or
 'ALL'
#       And: In 'LoggingConfiguration', 'Destinations' has been provided as a list
 containing non-empty strings or
#            valid local references
#      Then: PASS

#
# Constants
#
let STEP_FUNCTIONS_STATE_MACHINE_TYPE = "AWS::StepFunctions::StateMachine"
let ALLOWED_LOGGING_LEVELS = [ "ERROR", "ALL" ]
let INPUT_DOCUMENT = this

#
# Assignments
#
let step_functions_state_machines = Resources.*[ Type ==
 %STEP_FUNCTIONS_STATE_MACHINE_TYPE ]

#
# Primary Rules
#
rule step_functions_state_machine_logging_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %step_functions_state_machines not empty {
    check(%step_functions_state_machines.Properties)
        <<
        [CT.STEPFUNCTIONS.PR.1]: Require an AWS Step Functions state machine to have
 logging activated
```

```
        [FIX]: In 'LoggingConfiguration', set 'Level' to 'ERROR' or 'ALL', and set
 'Destinations' to a list with one or more valid Amazon CloudWatch Logs log group ARNs.
        >>
}

rule step_functions_state_machine_logging_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %STEP_FUNCTIONS_STATE_MACHINE_TYPE) {
    check(%INPUT_DOCUMENT.%STEP_FUNCTIONS_STATE_MACHINE_TYPE.resourceProperties)
        <<
        [CT.STEPFUNCTIONS.PR.1]: Require an AWS Step Functions state machine to have
 logging activated
        [FIX]: In 'LoggingConfiguration', set 'Level' to 'ERROR' or 'ALL', and set
 'Destinations' to a list with one or more valid Amazon CloudWatch Logs log group ARNs.
        >>
}

rule check(step_functions_state_machine) {
    %step_functions_state_machine {
        # Scenario 2
        LoggingConfiguration exists
        LoggingConfiguration is_struct

        LoggingConfiguration {
            # Scenarios 3, 4, 5 and 6
            Level exists
            Level in %ALLOWED_LOGGING_LEVELS

            Destinations exists
            Destinations is_list
            Destinations not empty

            Destinations[*] {
                CloudWatchLogsLogGroup exists
                CloudWatchLogsLogGroup is_struct

                CloudWatchLogsLogGroup {
                    LogGroupArn exists

                    check_is_string_and_not_empty(LogGroupArn) or
                    check_local_references(%INPUT_DOCUMENT, LogGroupArn,
 "AWS::Logs::LogGroup")
                }
            }
        }
```

```
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}

rule check_is_string_and_not_empty(value) {
    %value {
        this is_string
        this != /\A\s*\z/
    }
}

rule check_local_references(doc, reference_properties, referenced_resource_type) {
    %reference_properties {
        'Fn::GetAtt' {
            query_for_resource(%doc, this[0], %referenced_resource_type)
                <<Local Stack reference was invalid>>
        } or Ref {
            query_for_resource(%doc, this, %referenced_resource_type)
                <<Local Stack reference was invalid>>
        }
    }
}

rule query_for_resource(doc, resource_key, referenced_resource_type) {
    let referenced_resource = %doc.Resources[ keys == %resource_key ]
    %referenced_resource not empty
    %referenced_resource {
        Type == %referenced_resource_type
    }
}
```

CT.STEPFUNCTIONS.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  StepFunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - states.amazonaws.com
          Action:
          - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn:
                Fn::Sub: arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount:
                Ref: AWS::AccountId
      Path: /
      Policies:
      - PolicyName: StepFunctionLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogDelivery
            - logs:GetLogDelivery
            - logs:UpdateLogDelivery
            - logs:DeleteLogDelivery
```

```
            - logs:ListLogDeliveries
            - logs:PutLogEvents
            - logs:PutResourcePolicy
            - logs:DescribeResourcePolicies
            - logs:DescribeLogGroups
            Resource: '*'
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties: {}
  StateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      StateMachineType: STANDARD
      DefinitionString: '{"StartAt": "Example","States": {"Example": {"Type": "Task",
 "Resource":"arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
        "End": true}}}'
      RoleArn:
        Fn::GetAtt:
        - StepFunctionExecutionRole
        - Arn
      LoggingConfiguration:
        Level: ALL
        Destinations:
        - CloudWatchLogsLogGroup:
            LogGroupArn:
              Fn::GetAtt:
              - LogGroup
              - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  StepFunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
```

```
          Principal:
            Service:
            - states.amazonaws.com
          Action:
          - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn:
                Fn::Sub: arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount:
                Ref: AWS::AccountId
      Path: /
      Policies:
      - PolicyName: StepFunctionLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogDelivery
            - logs:GetLogDelivery
            - logs:UpdateLogDelivery
            - logs:DeleteLogDelivery
            - logs:ListLogDeliveries
            - logs:PutLogEvents
            - logs:PutResourcePolicy
            - logs:DescribeResourcePolicies
            - logs:DescribeLogGroups
            Resource: '*'
  StateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      StateMachineType: STANDARD
      DefinitionString: '{"StartAt": "Example","States": {"Example": {"Type": "Task",
 "Resource":"arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
        "End": true}}}'
      RoleArn:
        Fn::GetAtt:
        - StepFunctionExecutionRole
        - Arn
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  StepFunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - states.amazonaws.com
          Action:
          - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn:
                Fn::Sub: arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount:
                Ref: AWS::AccountId
      Path: /
      Policies:
      - PolicyName: StepFunctionLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogDelivery
            - logs:GetLogDelivery
            - logs:UpdateLogDelivery
            - logs:DeleteLogDelivery
            - logs:ListLogDeliveries
            - logs:PutLogEvents
            - logs:PutResourcePolicy
            - logs:DescribeResourcePolicies
```

```
            - logs:DescribeLogGroups
            Resource: '*'
  StateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      StateMachineType: STANDARD
      DefinitionString: '{"StartAt": "Example","States": {"Example": {"Type": "Task",
"Resource":"arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
        "End": true}}}'
      RoleArn:
        Fn::GetAtt:
        - StepFunctionExecutionRole
        - Arn
      LoggingConfiguration:
        Level: 'OFF'
```

[CT.STEPFUNCTIONS.PR.2] Require an AWS Step Functions state machine to have AWS X-Ray tracing activated

This control checks whether an AWS Step Functions state machine has AWS X-Ray tracing enabled.

- **Control objective:** Establish logging and monitoring
- **Implementation:** CloudFormation guard rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::StepFunctions::StateMachine`
- **CloudFormation guard rule:** CT.STEPFUNCTIONS.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.STEPFUNCTIONS.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation templates related to this control, see: CT.STEPFUNCTIONS.PR.2 example templates

Explanation

A tracing configuration allows your state machine to send tracing data to AWS X-Ray, so you can visualize the components of your state machine, identify performance bottlenecks, and troubleshoot requests that resulted in errors.

Remediation for rule failure

In the `TracingConfiguration` property, set the value of `Enabled` to true.

The examples that follow show how to implement this remediation.

AWS Step Functions State Machine - Example

An AWS Step Functions state machine configured to send trace data to AWS X-Ray. The example is shown in JSON and in YAML.

JSON example

```
{
    "StateMachine": {
        "Type": "AWS::StepFunctions::StateMachine",
        "Properties": {
            "StateMachineType": "STANDARD",
            "DefinitionString": "{\"StartAt\": \"Sample\",\"States\":
 {\"Sample\": {\"Type\": \"Task\", \"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:SampleFunction\", \"End\": true}}}",
            "RoleArn": {
                "Fn::GetAtt": [
                    "StepFunctionExecutionRole",
                    "Arn"
                ]
            },
            "LoggingConfiguration": {
                "Level": "ALL",
                "Destinations": [
                    {
                        "CloudWatchLogsLogGroup": {
                            "LogGroupArn": {
                                "Fn::GetAtt": [
                                    "LogGroup",
                                    "Arn"
                                ]
                            }
                        }
```

```
                    }
                ]
            },
            "TracingConfiguration": {
                "Enabled": true
            }
        }
    }
}
```

YAML example

```
StateMachine:
  Type: AWS::StepFunctions::StateMachine
  Properties:
    StateMachineType: STANDARD
    DefinitionString: '{"StartAt": "Sample","States": {"Sample": {"Type": "Task",
      "Resource":"arn:aws:lambda:us-east-1:111122223333:function:SampleFunction",
      "End": true}}}'
    RoleArn: !GetAtt 'StepFunctionExecutionRole.Arn'
    LoggingConfiguration:
      Level: ALL
      Destinations:
        - CloudWatchLogsLogGroup:
            LogGroupArn: !GetAtt 'LogGroup.Arn'
    TracingConfiguration:
      Enabled: true
```

CT.STEPFUNCTIONS.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   step_functions_state_machine_tracing_enabled_check
#
# Description:
```

```
#   This control checks whether an AWS Step Functions state machine has AWS X-Ray
 tracing enabled.
#
# Reports on:
#   AWS::StepFunctions::StateMachine
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any StepFunctions state machine
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'TracingConfiguration' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'TracingConfiguration' has been provided
#       And: In 'TracingConfiguration', 'Enabled' has not been provided or provided and
 set to a value other
#            than bool(true)
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a StepFunctions state machine resource
#       And: 'TracingConfiguration' has been provided
#       And: In 'TracingConfiguration', 'Enabled' has been provided and set to
 bool(true)
#      Then: PASS

#
# Constants
#
let STEP_FUNCTIONS_STATE_MACHINE_TYPE = "AWS::StepFunctions::StateMachine"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let step_functions_state_machines = Resources.*[ Type ==
 %STEP_FUNCTIONS_STATE_MACHINE_TYPE ]

#
# Primary Rules
#
rule step_functions_state_machine_tracing_enabled_check when
 is_cfn_template(%INPUT_DOCUMENT)

 %step_functions_state_machines not empty {
    check(%step_functions_state_machines.Properties)
        <<
        [CT.STEPFUNCTIONS.PR.2]: Require an AWS Step Functions state machine to have
 AWS X-Ray tracing activated
        [FIX]: In the 'TracingConfiguration' property, set the value of 'Enabled' to
 true.
        >>
}

rule step_functions_state_machine_tracing_enabled_check when
 is_cfn_hook(%INPUT_DOCUMENT, %STEP_FUNCTIONS_STATE_MACHINE_TYPE) {
    check(%INPUT_DOCUMENT.%STEP_FUNCTIONS_STATE_MACHINE_TYPE.resourceProperties)
        <<
        [CT.STEPFUNCTIONS.PR.2]: Require an AWS Step Functions state machine to have
 AWS X-Ray tracing activated
        [FIX]: In the 'TracingConfiguration' property, set the value of 'Enabled' to
 true.
        >>
}

rule check(step_functions_state_machine) {
    %step_functions_state_machine {
        # Scenario 2
        TracingConfiguration exists
        TracingConfiguration is_struct

        TracingConfiguration {
            # Scenarios 3 and 4
            Enabled exists
            Enabled == true
```

```
        }
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists  or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.STEPFUNCTIONS.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  StepFunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - states.amazonaws.com
          Action:
          - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn:
```

```
                Fn::Sub: arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount:
                Ref: AWS::AccountId
      Path: /
      Policies:
      - PolicyName: StepFunctionLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogDelivery
            - logs:GetLogDelivery
            - logs:UpdateLogDelivery
            - logs:DeleteLogDelivery
            - logs:ListLogDeliveries
            - logs:PutLogEvents
            - logs:PutResourcePolicy
            - logs:DescribeResourcePolicies
            - logs:DescribeLogGroups
            Resource: '*'
      - PolicyName: StepFunctionTracingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - xray:PutTraceSegments
            - xray:PutTelemetryRecords
            - xray:GetSamplingRules
            - xray:GetSamplingTargets
            Resource: '*'
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties: {}
  StateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      StateMachineName:
        Fn::Sub: Example-StateMachine-${AWS::StackName}
      StateMachineType: STANDARD
```

```
      DefinitionString: '{"StartAt": "Example","States": {"Example": {"Type": "Task",
 "Resource":"arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
        "End": true}}}'
      RoleArn:
        Fn::GetAtt:
        - StepFunctionExecutionRole
        - Arn
      LoggingConfiguration:
        Level: ALL
        Destinations:
        - CloudWatchLogsLogGroup:
            LogGroupArn:
              Fn::GetAtt:
              - LogGroup
              - Arn
      TracingConfiguration:
        Enabled: true
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  StepFunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - states.amazonaws.com
          Action:
          - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn:
                Fn::Sub: arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
```

```
              aws:SourceAccount:
                Ref: AWS::AccountId
      Path: /
      Policies:
      - PolicyName: StepFunctionLoggingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogDelivery
            - logs:GetLogDelivery
            - logs:UpdateLogDelivery
            - logs:DeleteLogDelivery
            - logs:ListLogDeliveries
            - logs:PutLogEvents
            - logs:PutResourcePolicy
            - logs:DescribeResourcePolicies
            - logs:DescribeLogGroups
            Resource: '*'
      - PolicyName: StepFunctionTracingPolicy
        PolicyDocument:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Action:
            - xray:PutTraceSegments
            - xray:PutTelemetryRecords
            - xray:GetSamplingRules
            - xray:GetSamplingTargets
            Resource: '*'
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties: {}
  StateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      StateMachineName:
        Fn::Sub: Example-StateMachine-${AWS::StackName}
      StateMachineType: STANDARD
      DefinitionString: '{"StartAt": "Example","States": {"Example": {"Type": "Task",
 "Resource":"arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
        "End": true}}}'
      RoleArn:
```

```
        Fn::GetAtt:
        - StepFunctionExecutionRole
        - Arn
      LoggingConfiguration:
        Level: ALL
        Destinations:
        - CloudWatchLogsLogGroup:
            LogGroupArn:
              Fn::GetAtt:
              - LogGroup
              - Arn
      TracingConfiguration:
        Enabled: false
```

AWS WAF regional controls

Topics

- [CT.WAF-REGIONAL.PR.1] Require any AWS WAF regional rule to have a condition
- [CT.WAF-REGIONAL.PR.2] Require any AWS WAF regional web access control list (ACL) to have a rule or rule group

[CT.WAF-REGIONAL.PR.1] Require any AWS WAF regional rule to have a condition

This control checks whether an AWS WAF Classic Regional rule contains any conditions.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAFRegional::Rule`
- **CloudFormation guard rule:** CT.WAF-REGIONAL.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAF-REGIONAL.PR.1 rule specification

- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAF-REGIONAL.PR.1 example templates

Explanation

An AWS WAF (web application firewall) Regional rule can contain multiple conditions. The rule count's conditions allow for traffic inspection, based on a defined action, such as `allow`, `block`, or `count`. Without any conditions, the traffic passes without inspection. An AWS WAF Regional rule with no conditions, but with a name or tag suggesting `allow`, `block`, or `count`, could lead to the inaccurate assumption that one of those actions is occurring.

Remediation for rule failure

Provide one or more AWS WAF rule conditions within the `Predicates` property.

The examples that follow show how to implement this remediation.

AWS WAF Classic Regional Rule - Example

AWS WAF Classic Regional rule configured with an IP match predicate. The example is shown in JSON and in YAML.

JSON example

```
{
    "WafRegionalRule": {
        "Type": "AWS::WAFRegional::Rule",
        "Properties": {
            "Name": "SampleRule",
            "MetricName": "SampleRuleMetric",
            "Predicates": [
                {
                    "DataId": {
                        "Ref": "IPSet"
                    },
                    "Negated": false,
                    "Type": "IPMatch"
                }
            ]
        }
    }
}
```

YAML example

```
WafRegionalRule:
  Type: AWS::WAFRegional::Rule
  Properties:
    Name: SampleRule
    MetricName: SampleRuleMetric
    Predicates:
      - DataId: !Ref 'IPSet'
        Negated: false
        Type: IPMatch
```

CT.WAF-REGIONAL.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   waf_regional_rule_not_empty_check
#
# Description:
#   This control checks whether a AWS WAF Classic Regional rule contains any
 conditions.
#
# Reports on:
#   AWS::WAFRegional::Rule
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
```

```
#       And: The input document does not contain any AWS WAF Classic Regional rule
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional resource
#       And: 'Predicates' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional rule resource
#       And: 'Predicates' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional rule resource
#       And: 'Predicates' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAF_REGIONAL_RULE_TYPE = "AWS::WAFRegional::Rule"
let INPUT_DOCUMENT = this

#
# Assignments
#
let waf_regional_rules = Resources.*[ Type == %WAF_REGIONAL_RULE_TYPE ]

#
# Primary Rules
#
rule waf_regional_rule_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %waf_regional_rules not empty {
    check(%waf_regional_rules.Properties)
        <<
        [CT.WAF-REGIONAL.PR.1]: Require any AWS WAF regional rule to have a condition
        [FIX]: Provide one or more AWS WAF rule conditions within the 'Predicates'
 property.
        >>
}
```

```
rule waf_regional_rule_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAF_REGIONAL_RULE_TYPE) {
    check(%INPUT_DOCUMENT.%WAF_REGIONAL_RULE_TYPE.resourceProperties)
        <<
        [CT.WAF-REGIONAL.PR.1]: Require any AWS WAF regional rule to have a condition
        [FIX]: Provide one or more AWS WAF rule conditions within the 'Predicates'
 property.
        >>
}

#
# Parameterized Rules
#
rule check(waf_regional_rule) {
    %waf_regional_rule {
        # Scenario 2, 3 and 4
        Predicates exists
        Predicates is_list
        Predicates not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAF-REGIONAL.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IPSetDenylist:
    Type: AWS::WAFRegional::IPSet
    Properties:
      Name: IPSet for deny listed IP addresses
      IPSetDescriptors:
      - Type: IPV4
        Value: 192.0.2.44/32
  WafRegionalRule:
    Type: AWS::WAFRegional::Rule
    Properties:
      Name: ExampleRule
      MetricName: ExampleRuleMetric
      Predicates:
      - DataId:
          Ref: IPSetDenylist
        Negated: false
        Type: "IPMatch"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WafRegionalRule:
    Type: AWS::WAFRegional::Rule
    Properties:
      Name: ExampleRule
      MetricName: ExampleRuleMetric
```

[CT.WAF-REGIONAL.PR.2] Require any AWS WAF regional web access control list (ACL) to have a rule or rule group

This control checks whether an AWS WAF Classic Regional web ACL contains any WAF rules or rule groups.

- **Control objective:** Limit network access

- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAFRegional::WebACL`
- **CloudFormation guard rule:** CT.WAF-REGIONAL.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAF-REGIONAL.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAF-REGIONAL.PR.2 example templates

Explanation

An AWS WAF Regional web access control list (ACL) can contain a collection of rules and rule groups that inspect and control web requests. If a web ACL is empty, the web traffic can pass without being detected or acted upon by WAF, depending on the default action.

Remediation for rule failure

Provide one or more AWS WAF rules within the `Rules` property.

The examples that follow show how to implement this remediation.

AWS WAF Classic Regional web ACL - Example

AWS WAF Classic Regional web ACL configured with a rule to block requests based on an IP set match. The example is shown in JSON and in YAML.

JSON example

```
{
    "WafRegionalWebAcl": {
        "Type": "AWS::WAFRegional::WebACL",
        "Properties": {
            "Name": "SampleWebACL",
            "DefaultAction": {
                "Type": "ALLOW"
            },
```

```
            "MetricName": "SampleWebACLMetric",
            "Rules": [
                {
                    "Action": {
                        "Type": "BLOCK"
                    },
                    "Priority": 1,
                    "RuleId": {
                        "Ref": "IPSetRule"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
WafRegionalWebAcl:
  Type: AWS::WAFRegional::WebACL
  Properties:
    Name: SampleWebACL
    DefaultAction:
      Type: ALLOW
    MetricName: SampleWebACLMetric
    Rules:
      - Action:
          Type: BLOCK
        Priority: 1
        RuleId: !Ref 'IPSetRule'
```

CT.WAF-REGIONAL.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
```

```
#   waf_regional_webacl_not_empty_check
#
# Description:
#   This control checks whether an AWS WAF Classic Regional web ACL contains any WAF
 rules or rule groups.
#
# Reports on:
#   AWS::WAFRegional::WebACL
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any WAF Classic Regional web ACL
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional web ACL resource
#       And: 'Rules' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional web ACL resource
#       And: 'Rules' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic Regional web ACL resource
#       And: 'Rules' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAF_REGIONAL_WEB_ACL_TYPE = "AWS::WAFRegional::WebACL"
let INPUT_DOCUMENT = this

#
```

```
# Assignments
#
let waf_regional_web_acls = Resources.*[ Type == %WAF_REGIONAL_WEB_ACL_TYPE ]

#
# Primary Rules
#
rule waf_regional_webacl_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                              %waf_regional_web_acls not empty {
    check(%waf_regional_web_acls.Properties)
        <<
        [CT.WAF-REGIONAL.PR.2]: Require any AWS WAF regional web access control list
 (ACL) to have a rule or rule group
        [FIX]: Provide one or more AWS WAF rules within the 'Rules' property.
        >>
}

rule waf_regional_webacl_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAF_REGIONAL_WEB_ACL_TYPE) {
    check(%INPUT_DOCUMENT.%WAF_REGIONAL_WEB_ACL_TYPE.resourceProperties)
        <<
        [CT.WAF-REGIONAL.PR.2]: Require any AWS WAF regional web access control list
 (ACL) to have a rule or rule group
        [FIX]: Provide one or more AWS WAF rules within the 'Rules' property.
        >>
}

#
# Parameterized Rules
#
rule check(waf_regional_web_acl) {
    %waf_regional_web_acl {
        # Scenario 2, 3 and 4
        Rules exists
        Rules is_list
        Rules not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
```

```
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAF-REGIONAL.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IPSetDenylist:
    Type: "AWS::WAFRegional::IPSet"
    Properties:
      Name: "IPSet for deny listed IP addresses"
      IPSetDescriptors:
      - Type: "IPV4"
        Value: "192.0.2.44/32"
  IPSetRule:
    Type: AWS::WAFRegional::Rule
    Properties:
      Name: ExampleIPSetRule
      MetricName: ExampleIPSetRuleMetric
      Predicates:
      - DataId:
          Ref: IPSetDenylist
        Negated: false
        Type: IPMatch
  WafRegionalWebAcl:
    Type: AWS::WAFRegional::WebACL
    Properties:
      Name: ExampleWebACL
      DefaultAction:
        Type: ALLOW
      MetricName: ExampleWebACLMetric
```

```
      Rules:
      - Action:
          Type: BLOCK
        Priority: 1
        RuleId:
          Ref: IPSetRule
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WafRegionalWebAcl:
    Type: AWS::WAFRegional::WebACL
    Properties:
      Name: ExampleWebACL
      DefaultAction:
        Type: ALLOW
      MetricName: ExampleWebACLMetric
```

AWS WAF controls

Topics

- [CT.WAF.PR.1] Require any AWS WAF global rule to have a condition
- [CT.WAF.PR.2] Require any AWS WAF global web ACL to have a rule or rule group

[CT.WAF.PR.1] Require any AWS WAF global rule to have a condition

This control checks whether an AWS WAF Classic global rule contains any conditions.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAF::Rule`
- **CloudFormation guard rule:** CT.WAF.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAF.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAF.PR.1 example templates

Explanation

A WAF global rule can contain multiple conditions. A rule's conditions allow for traffic inspection and take a defined action (allow, block, or count). Without any conditions, the traffic passes without inspection. A WAF global rule with no conditions, but with a name or tag suggesting allow, block, or count, could lead to the wrong assumption that one of those actions is occurring.

Remediation for rule failure

Provide one or more AWS WAF rule conditions within the `Predicates` property.

The examples that follow show how to implement this remediation.

AWS WAF Classic Global Rule - Example

AWS WAF Classic global rule configured with an IP match predicate. The example is shown in JSON and in YAML.

JSON example

```
{
    "WAFRule": {
        "Type": "AWS::WAF::Rule",
        "Properties": {
            "Name": "SampleWAFRule",
            "MetricName": "SampleWAFRuleMetric",
            "Predicates": [
                {
                    "DataId": {
                        "Ref": "IPSet"
                    },
                    "Negated": false,
                    "Type": "IPMatch"
                }
```

```
            ]
        }
    }
}
```

YAML example

```
WAFRule:
  Type: AWS::WAF::Rule
  Properties:
    Name: SampleWAFRule
    MetricName: SampleWAFRuleMetric
    Predicates:
      - DataId: !Ref 'IPSet'
        Negated: false
        Type: IPMatch
```

CT.WAF.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   waf_global_rule_not_empty_check
#
# Description:
#   This control checks whether an AWS WAF Classic global rule contains any conditions.
#
# Reports on:
#   AWS::WAF::Rule
#
# Evaluates:
#   CloudFormation, CloudFormation Hook
#
# Rule Parameters:
#   None
#
```

```
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any WAF Classic global rule resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a WAF Classic global rule resource
#       And: 'Predicates' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a WAF Classic global rule resource
#       And: 'Predicates' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains a WAF Classic global rule resource
#       And: 'Predicates' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAF_GLOBAL_RULE_TYPE = "AWS::WAF::Rule"
let INPUT_DOCUMENT = this

#
# Assignments
#
let waf_global_rules = Resources.*[ Type == %WAF_GLOBAL_RULE_TYPE ]

#
# Primary Rules
#
rule waf_global_rule_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %waf_global_rules not empty {
    check(%waf_global_rules.Properties)
        <<
        [CT.WAF.PR.1]: Require any AWS WAF global rule to have a condition
        [FIX]: Provide one or more AWS WAF rule conditions within the 'Predicates'
 property.
        >>
}
```

```
rule waf_global_rule_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAF_GLOBAL_RULE_TYPE) {
    check(%INPUT_DOCUMENT.%WAF_GLOBAL_RULE_TYPE.resourceProperties)
        <<
        [CT.WAF.PR.1]: Require any AWS WAF global rule to have a condition
        [FIX]: Provide one or more AWS WAF rule conditions within the 'Predicates'
 property.
        >>
}

#
# Parameterized Rules
#
rule check(waf_global_rule) {
    %waf_global_rule {
        # Scenario 2, 3 and 4
        Predicates exists
        Predicates is_list
        Predicates not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAF.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IPSetDenylist:
    Type: AWS::WAF::IPSet
    Properties:
      Name: IPSet for deny listed IP addresses
      IPSetDescriptors:
      - Type: IPV4
        Value: 192.0.2.44/32
  WafGlobalRule:
    Type: AWS::WAF::Rule
    Properties:
      Name: ExampleWAFRule
      MetricName: ExampleWAFRuleMetric
      Predicates:
      - DataId:
          Ref: IPSetDenylist
        Negated: false
        Type: "IPMatch"
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WafGlobalRule:
    Type: AWS::WAF::Rule
    Properties:
      Name: ExampleWAFRule
      MetricName: ExampleWAFRuleMetric
```

[CT.WAF.PR.2] Require any AWS WAF global web ACL to have a rule or rule group

This control checks whether an AWS WAF Classic global web ACL contains any WAF rules or rule groups.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAF::WebACL`
- **CloudFormation guard rule:** CT.WAF.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAF.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAF.PR.2 example templates

Explanation

A WAF global web ACL can contain a collection of rules and rule groups that inspect and control web requests. If a web ACL is empty, the web traffic can pass without being detected or acted upon by WAF depending on the default action.

Remediation for rule failure

Provide one or more AWS WAF rules within the `Rules` property.

The examples that follow show how to implement this remediation.

AWS WAF Classic Global Web ACL - Example

AWS WAF Classic global web ACL configured with a rule to block requests based on an IP set match. The example is shown in JSON and in YAML.

JSON example

```
{
    "WafGlobalWebAcl": {
        "Type": "AWS::WAF::WebACL",
        "Properties": {
            "Name": "SampleWebACL",
            "DefaultAction": {
                "Type": "ALLOW"
```

```
            },
            "MetricName": "SampleWebACLMetric",
            "Rules": [
                {
                    "Action": {
                        "Type": "BLOCK"
                    },
                    "Priority": 1,
                    "RuleId": {
                        "Ref": "IPSetRule"
                    }
                }
            ]
        }
    }
}
```

YAML example

```
WafGlobalWebAcl:
  Type: AWS::WAF::WebACL
  Properties:
    Name: SampleWebACL
    DefaultAction:
      Type: ALLOW
    MetricName: SampleWebACLMetric
    Rules:
      - Action:
          Type: BLOCK
        Priority: 1
        RuleId: !Ref 'IPSetRule'
```

CT.WAF.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
```

```
# Rule Identifier:
#   waf_global_webacl_not_empty_check
#
# Description:
#   This control checks whether an AWS WAF Classic global web ACL contains any WAF
 rules or rule groups.
#
# Reports on:
#   AWS::WAF::WebACL
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS WAF Classic global web ACL
 resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic global web ACL resource
#       And: 'Rules' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic global web ACL resource
#       And: 'Rules' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAF Classic global web ACL resource
#       And: 'Rules' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAF_GLOBAL_WEB_ACL_TYPE = "AWS::WAF::WebACL"
let INPUT_DOCUMENT = this
```

```
#
# Assignments
#
let waf_global_web_acls = Resources.*[ Type == %WAF_GLOBAL_WEB_ACL_TYPE ]

#
# Primary Rules
#
rule waf_global_webacl_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                            %waf_global_web_acls not empty {
    check(%waf_global_web_acls.Properties)
        <<
        [CT.WAF.PR.2]: Require any AWS WAF global web ACL to have a rule or rule group
        [FIX]: Provide one or more AWS WAF rules within the 'Rules' property.
        >>
}

rule waf_global_webacl_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAF_GLOBAL_WEB_ACL_TYPE) {
    check(%INPUT_DOCUMENT.%WAF_GLOBAL_WEB_ACL_TYPE.resourceProperties)
        <<
        [CT.WAF.PR.2]: Require any AWS WAF global web ACL to have a rule or rule group
        [FIX]: Provide one or more AWS WAF rules within the 'Rules' property.
        >>
}

#
# Parameterized Rules
#
rule check(waf_global_web_acl) {
    %waf_global_web_acl {
        # Scenario 2, 3 and 4
        Rules exists
        Rules is_list
        Rules not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
```

```
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAF.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  IPSetDenylist:
    Type: "AWS::WAF::IPSet"
    Properties:
      Name: "IPSet for deny listed IP addresses"
      IPSetDescriptors:
      - Type: "IPV4"
        Value: "192.0.2.44/32"
  IPSetRule:
    Type: AWS::WAF::Rule
    Properties:
      Name: ExampleIPSetRule
      MetricName: ExampleIPSetRuleMetric
      Predicates:
      - DataId:
          Ref: IPSetDenylist
        Negated: false
        Type: IPMatch
  WafGlobalWebAcl:
    Type: AWS::WAF::WebACL
    Properties:
      Name: ExampleWebACL
      DefaultAction:
        Type: ALLOW
      MetricName: ExampleWebACLMetric
```

```
      Rules:
      - Action:
          Type: BLOCK
        Priority: 1
        RuleId:
          Ref: IPSetRule
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WafGlobalWebAcl:
    Type: AWS::WAF::WebACL
    Properties:
      Name: ExampleWebACL
      DefaultAction:
        Type: ALLOW
      MetricName: ExampleWebACLMetric
```

AWS WAFV2 controls

Topics

- [CT.WAFV2.PR.1] Require an AWS WAFV2 web ACL to be non-empty
- [CT.WAFV2.PR.2] Require an AWS WAFV2 rule group to be non-empty

[CT.WAFV2.PR.1] Require an AWS WAFV2 web ACL to be non-empty

This control checks whether an AWS WAFV2 web ACL contains any WAF rules or WAF rule groups.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAFv2::WebACL`
- **CloudFormation guard rule:** CT.WAFV2.PR.1 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAFV2.PR.1 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAFV2.PR.1 example templates

Explanation

A web access control list (ACL) attached to WAFv2 can contain a collection of rules and rule groups. The rules are designed to inspect and control web requests. If a web ACL is empty, the web traffic passes without being detected or acted upon by the web application firewall (WAF).

Remediation for rule failure

Provide one or more AWS WAFV2 rules within the `Rules` property.

The examples that follow show how to implement this remediation.

AWS WAFV2 web ACL - Example

AWS WAFV2 web ACL configured with a rule to block requests based on a cross-site scripting (XSS) match. statement. The example is shown in JSON and in YAML.

JSON example

```
{
    "WAFv2WebACL": {
        "Type": "AWS::WAFv2::WebACL",
        "Properties": {
            "Scope": "REGIONAL",
            "Description": "Sample WebACL",
            "DefaultAction": {
                "Allow": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "SampleWebACLMetric"
            },
            "Rules": [
```

```
                {
                    "Name": "SampleXssRule",
                    "Priority": 0,
                    "Action": {
                        "Block": {}
                    },
                    "VisibilityConfig": {
                        "SampledRequestsEnabled": true,
                        "CloudWatchMetricsEnabled": true,
                        "MetricName": "SampleXssMatchMetric"
                    },
                    "Statement": {
                        "XssMatchStatement": {
                            "FieldToMatch": {
                                "AllQueryArguments": {}
                            },
                            "TextTransformations": [
                                {
                                    "Priority": 1,
                                    "Type": "NONE"
                                }
                            ]
                        }
                    }
                }
            ]
        }
    }
}
```

YAML example

```
WAFv2WebACL:
  Type: AWS::WAFv2::WebACL
  Properties:
    Scope: REGIONAL
    Description: Sample WebACL
    DefaultAction:
      Allow: {}
    VisibilityConfig:
      SampledRequestsEnabled: true
```

```
      CloudWatchMetricsEnabled: true
      MetricName: SampleWebACLMetric
    Rules:
      - Name: SampleXssRule
        Priority: 0
        Action:
          Block: {}
        VisibilityConfig:
          SampledRequestsEnabled: true
          CloudWatchMetricsEnabled: true
          MetricName: SampleXssMatchMetric
        Statement:
          XssMatchStatement:
            FieldToMatch:
              AllQueryArguments: {}
            TextTransformations:
              - Priority: 1
                Type: NONE
```

CT.WAFV2.PR.1 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   wafv2_webacl_not_empty_check
#
# Description:
#   This control checks whether an AWS WAFV2 web ACL contains any WAF rules or WAF rule
 groups.
#
# Reports on:
#   AWS::WAFv2::WebACL
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
```

```
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any WAFv2 web ACL resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an WAFv2 web ACL resource
#       And: 'Rules' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an WAFv2 web ACL resource
#       And: 'Rules' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an WAFv2 web ACL resource
#       And: 'Rules' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAFV2_WEB_ACL_TYPE = "AWS::WAFv2::WebACL"
let INPUT_DOCUMENT = this

#
# Assignments
#
let wafv2_web_acls = Resources.*[ Type == %WAFV2_WEB_ACL_TYPE ]

#
# Primary Rules
#
rule wafv2_webacl_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                       %wafv2_web_acls not empty {
    check(%wafv2_web_acls.Properties)
        <<
        [CT.WAFV2.PR.1]: Require an AWS WAFV2 web ACL to be non-empty
        [FIX]: Provide one or more AWS WAFv2 rules within the 'Rules' property.
        >>
}
```

```
rule wafv2_webacl_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAFV2_WEB_ACL_TYPE) {
    check(%INPUT_DOCUMENT.%WAFV2_WEB_ACL_TYPE.resourceProperties)
        <<
        [CT.WAFV2.PR.1]: Require an AWS WAFV2 web ACL to be non-empty
        [FIX]: Provide one or more AWS WAFv2 rules within the 'Rules' property.
        >>
}

#
# Parameterized Rules
#
rule check(wafv2_web_acl) {
    %wafv2_web_acl {
        # Scenario 2, 3 and 4
        Rules exists
        Rules is_list
        Rules not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAFV2.PR.1 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  WAFv2WebACL:
    Type: AWS::WAFv2::WebACL
    Properties:
      Scope: REGIONAL
      Description: Example WebACL
      DefaultAction:
        Allow: {}
      VisibilityConfig:
        SampledRequestsEnabled: true
        CloudWatchMetricsEnabled: true
        MetricName: ExampleWebACLMetric
      Rules:
      - Name: ExampleXssRule
        Priority: 0
        Action:
          Block: {}
        VisibilityConfig:
          SampledRequestsEnabled: true
          CloudWatchMetricsEnabled: true
          MetricName: ExampleXssMatchMetric
        Statement:
          XssMatchStatement:
            FieldToMatch:
              AllQueryArguments: {}
            TextTransformations:
            - Priority: 1
              Type: NONE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WAFv2WebACL:
    Type: AWS::WAFv2::WebACL
    Properties:
      Scope: REGIONAL
      Description: Example WebACL
      DefaultAction:
```

```
        Allow: {}
      VisibilityConfig:
        SampledRequestsEnabled: true
        CloudWatchMetricsEnabled: true
        MetricName: ExampleWebACLMetric
```

[CT.WAFV2.PR.2] Require an AWS WAFV2 rule group to be non-empty

This control checks whether AWS WAFV2 rule groups contain rules.

- **Control objective:** Limit network access
- **Implementation:** CloudFormation Guard Rule
- **Control behavior:** Proactive
- **Resource types:** `AWS::WAFv2::RuleGroup`
- **CloudFormation guard rule:** CT.WAFV2.PR.2 rule specification

Details and examples

- For details about the PASS, FAIL, and SKIP behaviors associated with this control, see the: CT.WAFV2.PR.2 rule specification
- For examples of PASS and FAIL CloudFormation Templates related to this control, see: CT.WAFV2.PR.2 example templates

Explanation

An AWS WAFV2 rule group can contain multiple rules. The rules are designed to inspect and control web requests. When an empty AWS WAFV2 rule group is associated with a web ACL, and the web ACL is not associated with any other rules or rule groups, the web traffic passes without being detected or acted upon by the web application firewall (WAF).

Remediation for rule failure

Provide one or more AWS WAFV2 rules within the `Rules` property.

The examples that follow show how to implement this remediation.

AWS WAFV2 Rule Group - Example

AWS WAFV2 rule group configured with a rule to block requests based on a cross-site scripting (XSS) match. statement. The example is shown in JSON and in YAML.

JSON example

```
{
    "WAFv2RuleGroup": {
        "Type": "AWS::WAFv2::RuleGroup",
        "Properties": {
            "Scope": "REGIONAL",
            "Description": "Sample Rule Group",
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "SampleRuleGroupMetric"
            },
            "Capacity": 1000,
            "Rules": [
                {
                    "Name": "XssRule",
                    "Priority": 0,
                    "Action": {
                        "Block": {}
                    },
                    "VisibilityConfig": {
                        "SampledRequestsEnabled": true,
                        "CloudWatchMetricsEnabled": true,
                        "MetricName": "SampleXssMatchMetric"
                    },
                    "Statement": {
                        "XssMatchStatement": {
                            "FieldToMatch": {
                                "AllQueryArguments": {}
                            },
                            "TextTransformations": [
                                {
                                    "Priority": 1,
                                    "Type": "NONE"
                                }
                            ]
                        }
```

```
                    }
                }
            ]
        }
    }
}
```

YAML example

```
WAFv2RuleGroup:
  Type: AWS::WAFv2::RuleGroup
  Properties:
    Scope: REGIONAL
    Description: Sample Rule Group
    VisibilityConfig:
      SampledRequestsEnabled: true
      CloudWatchMetricsEnabled: true
      MetricName: SampleRuleGroupMetric
    Capacity: 1000
    Rules:
      - Name: XssRule
        Priority: 0
        Action:
          Block: {}
        VisibilityConfig:
          SampledRequestsEnabled: true
          CloudWatchMetricsEnabled: true
          MetricName: SampleXssMatchMetric
        Statement:
          XssMatchStatement:
            FieldToMatch:
              AllQueryArguments: {}
            TextTransformations:
              - Priority: 1
                Type: NONE
```

CT.WAFV2.PR.2 rule specification

```
# ####################################
##       Rule Specification        ##
####################################
#
# Rule Identifier:
#   wafv2_rulegroup_not_empty_check
#
# Description:
#   This control checks whether AWS WAFV2 rule groups contain rules.
#
# Reports on:
#   AWS::WAFv2::RuleGroup
#
# Evaluates:
#   CloudFormation, CloudFormation hook
#
# Rule Parameters:
#   None
#
# Scenarios:
#   Scenario: 1
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document does not contain any AWS WAFV2 rule group resources
#      Then: SKIP
#   Scenario: 2
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAFV2 rule group resource
#       And: 'Rules' has not been provided
#      Then: FAIL
#   Scenario: 3
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAFV2 rule group resource
#       And: 'Rules' has been provided as an empty list
#      Then: FAIL
#   Scenario: 4
#     Given: The input document is an CloudFormation or CloudFormation hook document
#       And: The input document contains an AWS WAFV2 rule group resource
#       And: 'Rules' has been provided as a non-empty list
#      Then: PASS

#
# Constants
#
let WAFV2_RULE_GROUP_TYPE = "AWS::WAFv2::RuleGroup"
```

```
let INPUT_DOCUMENT = this

#
# Assignments
#
let wafv2_rule_groups = Resources.*[ Type == %WAFV2_RULE_GROUP_TYPE ]

#
# Primary Rules
#
rule wafv2_rulegroup_not_empty_check when is_cfn_template(%INPUT_DOCUMENT)
                                          %wafv2_rule_groups not empty {
    check(%wafv2_rule_groups.Properties)
        <<
        [CT.WAFV2.PR.2]: Require an AWS WAFV2 rule group to be non-empty
        [FIX]: Provide one or more AWS WAFV2 rules within the 'Rules' property.
        >>
}

rule wafv2_rulegroup_not_empty_check when is_cfn_hook(%INPUT_DOCUMENT,
 %WAFV2_RULE_GROUP_TYPE) {
    check(%INPUT_DOCUMENT.%WAFV2_RULE_GROUP_TYPE.resourceProperties)
        <<
        [CT.WAFV2.PR.2]: Require an AWS WAFV2 rule group to be non-empty
        [FIX]: Provide one or more AWS WAFV2 rules within the 'Rules' property.
        >>
}

#
# Parameterized Rules
#
rule check(wafv2_rule_group) {
    %wafv2_rule_group {
        # Scenario 2, 3 and 4
        Rules exists
        Rules is_list
        Rules not empty
    }
}

#
# Utility Rules
#
rule is_cfn_template(doc) {
```

```
    %doc {
        AWSTemplateFormatVersion exists or
        Resources exists
    }
}

rule is_cfn_hook(doc, RESOURCE_TYPE) {
    %doc.%RESOURCE_TYPE.resourceProperties exists
}
```

CT.WAFV2.PR.2 example templates

You can view examples of the PASS and FAIL test artifacts for the AWS Control Tower proactive controls.

PASS Example - Use this template to verify a compliant resource creation.

```
Resources:
  WAFv2RuleGroup:
    Type: AWS::WAFv2::RuleGroup
    Properties:
      Scope: REGIONAL
      Description: Example Rule Group
      VisibilityConfig:
        SampledRequestsEnabled: true
        CloudWatchMetricsEnabled: true
        MetricName: SampleRuleGroupMetrics
      Capacity: 1000
      Rules:
      - Name: TestXssRule
        Priority: 0
        Action:
          Block: {}
        VisibilityConfig:
          SampledRequestsEnabled: true
          CloudWatchMetricsEnabled: true
          MetricName: ExampleXssMatchMetric
        Statement:
          XssMatchStatement:
            FieldToMatch:
```

```
              AllQueryArguments: {}
            TextTransformations:
            - Priority: 1
              Type: NONE
```

FAIL Example - Use this template to verify that the control prevents non-compliant resource creation.

```
Resources:
  WAFv2RuleGroup:
    Type: AWS::WAFv2::RuleGroup
    Properties:
      Scope: REGIONAL
      Description: Example Rule Group
      VisibilityConfig:
        SampledRequestsEnabled: true
        CloudWatchMetricsEnabled: true
        MetricName: ExampleRuleGroupMetric
      Capacity: 1000
      Rules: []
```

Preventive controls

A preventive control ensures that your accounts maintain compliance, because it disallows actions that lead to policy violations. The status of a preventive control is either **enforced** or **not enabled**. Preventive controls are supported in all AWS Regions.

- Preventive controls are implemented using service control policies (SCPs), or resource control policies (RCPs), each of which are part of AWS Organizations.
- Regarding nested OUs, preventive controls enabled on any OUs higher in the tree will apply to unregistered OUs in that tree.
- When you enable controls on an organizational unit (OU) that is registered with AWS Control Tower, preventive controls apply to all member accounts under the OU, enrolled and unenrolled.

Note

The AWS Control Tower mandatory controls have preventive behavior, except three that are detective. See Mandatory controls.

- Detect Public Read Access Setting for Log Archive
- Detect Public Write Access Setting for Log Archive
- Detect whether shared accounts under the Security organizational unit have AWS CloudTrail or CloudTrail Lake enabled

Topics

- Controls implemented with resource control policies (RCPs)
- Controls implemented with declarative policies
- Controls for AWS Backup
- Digital sovereignty controls

Controls implemented with resource control policies (RCPs)

This section provides information about AWS Control Tower controls that are implemented by resource control policies (RCPs). RCPs are a type of policy, which can enforce preventive controls on resources in your AWS Control Tower landing zone.

RCPs complement service control policies (SCPs)

- SCPs offer control over the maximum permissions for IAM roles and users in your landing zone.
- RCPs offer control over the maximum permissions on AWS resources in your landing zone.

RCPs are similar to SCPs, because they each contain explicit and implicit *allow* and *deny* capabilities, expressed in their policies. For more information, see the AWS Organizations documentation about RCPs.

Individual RCP controls apply to specific resources associated with the following AWS services:

- Amazon S3
- AWS Security Token Service (STS)

- AWS Key Management Service KMS)
- Amazon Simple Queue Service (SQS)
- AWS Secrets Manager

RCP-based controls are configurable. For more information, see Controls with parameters.

When to apply RCP controls

With RCP controls, you can establish a *data perimeter* for your landing zone.

- For example, you can limit access to resources so that only the principals in your organization can manage them, such as with control **[CT.S3.PV.4] Require that the organization's Amazon S3 resources are accessible only by IAM principals that belong to the organization or by an AWS service**.
- Similarly, you can restrict access to resources so that certain requirements must be met, such as with **[CT.S3.PV.3] Require requests to Amazon S3 resources to use a minimum TLS version of 1.3**.

AWS Control Tower RCP controls

AWS Control Tower offers multiple RCP-based controls that each focus on a single type of resource associated with a specific service, such as Amazon S3 buckets.

Topics

- [CT.KMS.PV.7] Require that the organization's AWS Key Management Service resources are accessible only by IAM principals that belong to the organization, or by an AWS service
- [CT.S3.PV.2] Require all requests to Amazon S3 resources use authentication based on an Authorization header
- [CT.S3.PV.3] Require requests to Amazon S3 resources to use a minimum TLS version of 1.3
- [CT.S3.PV.4] Require that the organization's Amazon S3 resources are accessible only by IAM principals that belong to the organization or by an AWS service
- [CT.S3.PV.5] Require encryption of data in transit for calls to Amazon S3 resources
- [CT.S3.PV.6] Require all object uploads to Amazon S3 buckets to use server-side encryption with an AWS KMS key (SSE-KMS)
- [CT.SECRETSMANAGER.PV.1] Require that the organization's AWS Secrets Manager resources are accessible only by IAM principals that belong to the organization or by an AWS service

- [CT.SQS.PV.1] Require that the organization's Amazon SQS resources are accessible only by IAM principals that belong to the organization, or by an AWS service
- [CT.STS.PV.1] Require that the organization's AWS Security Token Service resources are accessible only by IAM principals that belong to the organization, or by an AWS service

[CT.KMS.PV.7] Require that the organization's AWS Key Management Service resources are accessible only by IAM principals that belong to the organization, or by an AWS service

This control disallows AWS Key Management Service API operations for your organization's AWS Key Management Service (KMS) resources by an AWS IAM principal, when the principal is outside of the organization and is not an AWS service principal.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or through the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- When you enable this control, AWS Control Tower populates the template RCP with the ID of the organization that your AWS Control Tower landing zone governs.
- Choose the Organizational Unit (OU) to which this control will apply. If AWS Key Management Service resources in that OU must be accessible by a trusted party besides your organization or an AWS service (for instance, another organization or specific IAM role), this control causes requests by that trusted party to be denied. Consider which principals need access to the AWS Key Management Service resources in your OU before you enable this control on that OU.

- This control does not provide protection for cross-service confused deputy scenarios. Consider enabling the related control for AWS Key Management Service, which applies an RCP to govern direct AWS service access to the organization's AWS Key Management Service resources. For more information about cross-service confused deputy prevention, see Cross-service confused deputy prevention in the *AWS Identity and Access Management User Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: ExemptedPrincipalArns. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV7",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "kms:*",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:PrincipalIsAWSService": "false"
                },
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": {{OrganizationIds}}
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.S3.PV.2] Require all requests to Amazon S3 resources use authentication based on an Authorization header

This control disallows requests to your Amazon S3 resources that use an authentication method other than HTTP `Authorization` header-based authentication (presigned URL or HTTP POST requests).

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- This control disallows authenticated operations on your S3 resources where authentication information has been provided in a location other than the HTTP `Authorization` header, which means that the `s3:authType` field in the request context is set to a value other than `REST-HEADER`. This approach prevents the use of S3-presigned URL or HTTP POST requests. For more information on available authentication methods for Amazon S3, see Authenticating Requests (AWS Signature Version 4) in the *Amazon S3 User Guide*.
- If you need to use presigned URL or HTTP POST requests with your S3 resources, do not enable this control.
- This control does not help you manage public access to Amazon S3 resources. AWS Control Tower recommends using the Amazon S3 Block Public Access field to help manage public access to your Amazon S3 resources. For more information on **S3 Block Public Access**, see Blocking public access to your Amazon S3 storage in the *Amazon S3 User Guide*.

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV2",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:authType": "REST-HEADER"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.S3.PV.3] Require requests to Amazon S3 resources to use a minimum TLS version of 1.3

This control requires connections to Amazon S3 resources in your organization use TLS version 1.3 or higher.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Encrypt data in transit
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV3",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": "1.3"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.S3.PV.4] Require that the organization's Amazon S3 resources are accessible only by IAM principals that belong to the organization or by an AWS service

This control disallows Amazon S3 API operations for your organization's Amazon S3 resources by an AWS IAM principal, when the principal is outside of the organization and is not an AWS service principal.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- When you enable this control, AWS Control Tower populates the template RCP with the ID of the organization that your AWS Control Tower landing zone governs.
- Choose the Organizational Unit (OU) to which this control will apply. If Amazon S3 resources in that OU must be accessible by a trusted party besides your organization or an AWS service (for instance, another organization or specific IAM role), this control causes requests by that trusted party to be denied. Consider which principals need access to the Amazon S3 resources in your OU before you enable this control on that OU.
- This control does not provide protection for cross-service confused deputy scenarios. Consider enabling the related control for Amazon S3, which applies an RCP to govern direct AWS service access to the organization's Amazon S3 resources. For more information about cross-service confused deputy prevention, see Cross-service confused deputy prevention in the *AWS Identity and Access Management User Guide*.

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV4",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition": {
               "BoolIfExists": {
                    "aws:PrincipalIsAWSService": "false"
                },
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": {{OrganizationIds}}
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.S3.PV.5] Require encryption of data in transit for calls to Amazon S3 resources

This control prevents unencrypted connections to Amazon S3 resources in your organization, by using the `aws:SecureTransport` condition.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Encrypt data in transit
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- If you currently make HTTP connections to Amazon S3 endpoints, be sure that you migrate to HTTPS connections before you enable this control.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV5",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition": {
```

```
                "Bool": {
                  "aws:SecureTransport": "false"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.S3.PV.6] Require all object uploads to Amazon S3 buckets to use server-side encryption with an AWS KMS key (SSE-KMS)

This control prevents object uploads to your Amazon S3 buckets if the request does not include an `x-amz-server-side-encryption-aws-kms-key-id` header, unless the bucket is configured with default SSE-KMS encryption.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Encrypt data at rest
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- During landing zone setup, AWS Control Tower creates Amazon S3 buckets. Optionally, you can configure AWS Control Tower to orchestrate other AWS services, such as AWS

CloudTrail and AWS Config, to send log entries to these buckets. When you configure your landing zone, AWS Control Tower provides the option that the objects created by those other services are encrypted with an AWS KMS key that you manage. If you have not configured your landing zone to use a KMS key with those other services,this control prevents those other services from logging to the S3 buckets that AWS Control Tower created, for any targets, such as OUs, that contain these buckets.

As a best practice, configure AWS Control Tower to use KMS keys before you enable this control. Otherwise, this control could block `PutObject` requests from services that AWS Control Tower configures, such as AWS CloudTrail and AWS Config. To learn more about the resources that AWS Control Tower creates during landing zone setup, see Resources created in the shared accounts in the *AWS Control Tower User Guide*. To learn more about how AWS Control Tower uses KMS keys with other services, see Optionally configure AWS KMS keys in the *AWS Control Tower User Guide*.

- After you enable this control, if you try to upload an S3 object without the `x-amz-server-side-encryption-aws-kms-key-id` header in the request, the upload will fail for buckets that do not have default SSE-KMS encryption configured. Before enabling this control on a target, consider whether all Amazon S3 buckets in the target environment are configured with default SSE-KMS encryption. Alternatively, if you upload objects to buckets that are not configured with default SSE-KMS encryption, check that all clients set the `x-amz-server-side-encryption-aws-kms-key-id` explicitly.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTS3PV6",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            {% if ExemptedResourceArns %}
            "NotResource": {{ExemptedResourceArns}}
            {% else %}
            "Resource": "*"
```

```
            {% endif %},
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.SECRETSMANAGER.PV.1] Require that the organization's AWS Secrets Manager resources are accessible only by IAM principals that belong to the organization or by an AWS service

This control disallows AWS Secrets Manager API operations for your organization's AWS Secrets Manager resources by an AWS IAM principal, when the principal is outside of the organization and is not an AWS service principal.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Secrets Manager

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- When you enable this control, AWS Control Tower populates the template RCP with the ID of the organization that your AWS Control Tower landing zone governs.

- Choose the Organizational Unit (OU) to which this control will apply. If AWS Secrets Manager resources in that OU must be accessible by a trusted party besides your organization or an AWS service (for instance, another organization or specific IAM role), this control causes requests by that trusted party to be denied. Consider which principals need access to the AWS Secrets Manager resources in your OU before you enable this control on that OU.
- This control does not provide protection for cross-service confused deputy scenarios. Consider enabling the related control for AWS Secrets Manager, which applies an RCP to govern direct AWS service access to the organization's AWS Secrets Manager resources. For more information about cross-service confused deputy prevention, see Cross-service confused deputy prevention in the *AWS Identity and Access Management User Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTSECRETSMANAGERPV1",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "secretsmanager:*",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:PrincipalIsAWSService": "false"
                },
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": {{OrganizationIds}}
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
```

```
        }
    ]
}
```

[CT.SQS.PV.1] Require that the organization's Amazon SQS resources are accessible only by IAM principals that belong to the organization, or by an AWS service

This control disallows Amazon SQS API operations for your organization's Amazon SQS resources by an AWS IAM principal, when the principal is outside of the organization and is not an AWS service principal.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon SQS

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- When you enable this control, AWS Control Tower populates the template RCP with the ID of the organization that your AWS Control Tower landing zone governs.
- Choose the Organizational Unit (OU) to which this control will apply. If Amazon SQS resources in that OU must be accessible by a trusted party besides your organization or an AWS service (for instance, another organization or specific IAM role), this control causes requests by that trusted party to be denied. Consider which principals need access to the Amazon SQS resources in your OU before you enable this control on that OU.

- This control does not provide protection for cross-service confused deputy scenarios. Consider enabling the related control for Amazon SQS, which applies an RCP to govern direct AWS service access to the organization's Amazon SQS resources. For more information about cross-service confused deputy prevention, see Cross-service confused deputy prevention in the *AWS Identity and Access Management User Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTSQSPV1",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "sqs:*",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:PrincipalIsAWSService": "false"
                },
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": {{OrganizationIds}}
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.STS.PV.1] Require that the organization's AWS Security Token Service resources are accessible only by IAM principals that belong to the organization, or by an AWS service

This control disallows select AWS Security Token Service (STS) API operations by an AWS IAM principal for your organization's AWS Security Token Service resources, when the principal is outside of the organization and is not an AWS service principal.

This is a preventive control with elective guidance, based on resource control policies (RCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Security Token Service

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Resource control policy (RCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `Multiple`

Usage considerations

- When you enable this control, AWS Control Tower populates the template RCP with the ID of the organization that your AWS Control Tower landing zone governs.
- Choose the Organizational Unit (OU) to which this control will apply. If AWS Security Token Service (STS) resources in that OU must be accessible by a trusted party besides your organization or an AWS service (for instance, another organization or specific IAM role), this control causes requests by that trusted party to be denied. Consider which principals need access to the AWS Security Token Service resources in your OU before you enable this control on that OU.
- This control does not include `sts:AssumeRoleWithSAML` and `sts:AssumeRoleWithWebIdentity` permissions in its scope, as the respective STS operations do not use AWS security credentials, and therefore do not include the `aws:PrincipalOrgID` condition key value in the request context. To ensure that `AssumeRoleWithSAML` and `AssumeRoleWithWebIdentity` operations are not denied

by this control, `sts:SetSourceIdentity` and `sts:TagSession` permissions are also excluded from the controls scope.

- This control does not include `sts:GetCallerIdentity` permissions in its scope. No permissions are required to perform the respective STS operation.
- This control includes only actions that have resources listed in the **Resource type** column of the *AWS Security Token Service Authorization Reference* that can be invoked from outside the organization. For more information about the behavior of IAM actions that do not have associated resource types, see RCP Effects on Permissions in the *AWS Organizations User Guide*
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following resource control policy (RCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTSTSPV1",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:PrincipalIsAWSService": "false"
                },
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": {{OrganizationIds}}
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
```

```
            }
        }
    ]
}
```

Controls implemented with declarative policies

This section provides information about AWS Control Tower controls that are implemented by *declarative policies* from AWS Organizations. These are *preventive* controls. For more information about how declarative policies work as preventive controls in AWS Control Tower, see Declarative policies in the AWS Organizations documentation.

Declarative policies help you define and enforce your required configuration for specified AWS services, across your entire organization, at the OU level. When a declarative policy is applied, the configuration is maintained continuously.

Declarative policies are enforced in each AWS service's control plane, which is an important distinction from controls implemented by service control policies (SCPs). While SCPs regulate access to APIs, declarative policies are applied directly at the service level. This approach ensures that the specified configuration is enforced, even when new features or APIs are introduced by the service.

Available controls

Topics

- [CT.EC2.PV.7] Disallow all public sharing of Amazon EBS snapshots
- [CT.EC2.PV.8] Disallow inbound and outbound internet connections to your VPCs through an internet gateway (IGW) or egress-only internet gateway (EIGW)
- [CT.EC2.PV.9] Disallow access to the EC2 serial console for all EC2 instances
- [CT.EC2.PV.11] Disallow public sharing of Amazon Machine Images (AMIs)

[CT.EC2.PV.7] Disallow all public sharing of Amazon EBS snapshots

This control blocks the public sharing of your Amazon EBS snapshots by configuring block public access for Amazon EBS snapshot settings at an account level. This setting has the effect of

preventing all public sharing of your EBS snapshots, so that snapshots that previously were publicly shared are treated as private, and are no longer publicly available.

This is a preventive control with elective guidance, based on declarative policies. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Declarative policy for EC2
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`

Usage considerations

- Enabling this control does not change the permissions for snapshots that are publicly shared already. Instead, it prevents snapshots from being publicly visible and publicly accessible. Therefore, the attributes for these snapshots still indicate that they are publicly shared, even though they are not publicly available. If you later disable this control or adopt the related control to block new sharing in place of this control, these snapshots will become publicly available again.
- Enabling this control on an AWS account means that users in the account can no longer request new public sharing of EBS snapshots.
- This control includes an AWS Organizations inheritance operator for each policy setting that applies to child policies (`@@operators_allowed_for_child_policies` with a value of `@@all`). This operator allows you to add to, negate, or override each policy setting in this control, when it is applied to child OUs and accounts, by using the AWS Organizations declarative policy syntax. For more information on policy inheritance for AWS Organizations policies, see Inheritance operators in the *AWS Organizations User Guide*.

The artifact for this control is the following declarative policy.

```
{
    "ec2_attributes": {
        "snapshot_block_public_access": {
            "state": {
                "@@assign": "block_all_sharing",
                "@@operators_allowed_for_child_policies": ["@@all"]
            }
        }
    }
}
```

[CT.EC2.PV.8] Disallow inbound and outbound internet connections to your VPCs through an internet gateway (IGW) or egress-only internet gateway (EIGW)

This control blocks direct ingress and egress traffic from the internet to your VPCs through an IGW or EIGW, by configuring block public access for VPCs (VPC BPA) at an account level. This control also disallows configuration of any VPC BPA exclusions, which means that if you enable it, you cannot exclude VPCs or subnets from the effects of this control.

This is a preventive control with elective guidance, based on declarative policies. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Declarative policy for EC2
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`

Usage considerations

- This control does not block traffic to your VPC when the traffic originates from locations other than an internet gateway or an egress-only internet gateway, such as traffic from a peered VPC or other VPC attached gateway (for example, an AWS Transit Gateway network gateway or virtual private gateway). Traffic from these locations may have a public network origin; therefore, AWS Control Tower recommends using EC2 security groups to set least-privilege network access controls, and to ensure that only the required network connections are allowed.
- This control does not affect inbound traffic from serverless services, such as API Gateway and Lambda, in your VPCs (for example, traffic that arrives by means of Elastic network interfaces or API Gateway private integration). However, VPC BPA will block traffic to or from these services, if that traffic occurs through an IGW or EIGW, in a governed VPC.
- This control governs Amazon EC2 VPC block public access settings that are configured by means of EC2 `ModifyVpcBlockPublicAccessOptions`, `CreateVpcBlockPublicAccessExclusion`, and `ModifyVpcBlockPublicAccessExclusion` operations. If you apply this control, you cannot use these operations to modify these settings within an enrolled AWS account.
- For an overview of VPC connectivity options and recommendations for creating scalable and secure network architectures, see Building a Scalable and Secure Multi-VPC AWS Network Infrastructure in the *Building a Scalable and Secure Multi-VPC AWS Network Infrastructure - AWS Whitepaper*.
- This control includes an AWS Organizations inheritance operator for each policy setting that applies to child policies (`@@operators_allowed_for_child_policies` with a value of `@@all`). This operator allows you to add to, negate, or override each policy setting in this control, when it is applied to child OUs and accounts, by using the AWS Organizations declarative policy syntax. For more information on policy inheritance for AWS Organizations policies, see Inheritance operators in the *AWS Organizations User Guide*.

The artifact for this control is the following declarative policy.

```
{
    "ec2_attributes": {
```

```
        "vpc_block_public_access": {
            "internet_gateway_block": {
                "mode": {
                    "@@assign": "block_bidirectional",
                    "@@operators_allowed_for_child_policies": ["@@all"]
                },
                "exclusions_allowed": {
                    "@@assign": "disabled",
                    "@@operators_allowed_for_child_policies": ["@@all"]
                }
            }
        }
    }
}
```

[CT.EC2.PV.9] Disallow access to the EC2 serial console for all EC2 instances

This control prevents access to the Amazon EC2 serial console of all EC2 instances for your account.

This is a preventive control with elective guidance, based on declarative policies. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Declarative policy for EC2
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`

Usage considerations

- With the EC2 serial console, you have access to your Amazon EC2 instance's serial port, which you can use to troubleshoot boot, network configuration, and other issues. If you require EC2 serial console access, do not enable this control.

- This control governs Amazon EC2 Serial Console for instance settings that are configured by means of EC2 `EnableSerialConsoleAccess` and `DisableSerialConsoleAccess` operations. If you apply this control, you cannot use these operations to modify these settings within an enrolled AWS account.
- This control includes an AWS Organizations inheritance operator for each policy setting that applies to child policies (`@@operators_allowed_for_child_policies` with a value of `@@all`). This operator allows you to add to, negate, or override each policy setting in this control, when it is applied to child OUs and accounts, by using the AWS Organizations declarative policy syntax. For more information on policy inheritance for AWS Organizations policies, see Inheritance operators in the *AWS Organizations User Guide*.

The artifact for this control is the following declarative policy.

```
{
    "ec2_attributes": {
        "serial_console_access": {
            "status": {
                "@@assign": "disabled",
                "@@operators_allowed_for_child_policies": ["@@all"]
            }
        }
    }
}
```

[CT.EC2.PV.11] Disallow public sharing of Amazon Machine Images (AMIs)

This control prevents the public sharing of your AMIs by configuring block public access for AMIs at an account level. If you already have public AMIs, they remain publicly available.

This is a preventive control with elective guidance, based on declarative policies. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Declarative policy for EC2
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`

Usage considerations

- This control disallows new public sharing of AMIs. It does not restrict access to AMIs that have been shared publicly before this control was enabled.
- This control governs Amazon EC2 AMI Block Public Access (BPA) settings that are configured by means of EC2 `EnableImageBlockPublicAccess` and `DisableImageBlockPublicAccess` operations. If you apply this control, you cannot use these operations to modify these settings within an enrolled AWS account.
- This control includes an AWS Organizations inheritance operator for each policy setting that applies to child policies (`@@operators_allowed_for_child_policies` with a value of `@@all`). This operator allows you to add to, negate, or override each policy setting in this control, when it is applied to child OUs and accounts, by using the AWS Organizations declarative policy syntax. For more information on policy inheritance for AWS Organizations policies, see Inheritance operators in the *AWS Organizations User Guide*.

The artifact for this control is the following declarative policy.

```
{
    "ec2_attributes": {
        "image_block_public_access": {
            "state": {
                "@@assign": "block_new_sharing",
                "@@operators_allowed_for_child_policies": ["@@all"]
            }
        }
    }
```

```
}
```

Controls for AWS Backup

When you enable AWS Backup in your AWS Control Tower landing zone, some preventive controls are activated in your environment. These controls protect the resources that AWS Backup needs to operate with AWS Control Tower. You cannot enable these controls if AWS Backup is not enabled for your landing zone.

Topics

- [CT.BACKUP.PV.1] Disallow modification of a tag that AWS Control Tower applies to AWS Backup resources
- [CT.BACKUP.PV.2] Disallow modification of an AWS Backup report plan that AWS Control Tower manages
- [CT.BACKUP.PV.3] Disallow modification of an AWS Backup resource that AWS Control Tower manages
- [CT.IAM.PV.1] Disallow modification of an AWS IAM role that AWS Control Tower utilizes to manage AWS Backup resources
- [CT.S3.PV.1] Disallow modification of an Amazon S3 bucket that stores AWS Backup reports for AWS Control Tower

[CT.BACKUP.PV.1] Disallow modification of a tag that AWS Control Tower applies to AWS Backup resources

This control limits changes to tags that AWS Control Tower applies to AWS Backup resources.

This is a preventive control with elective guidance. By default, this control is not enabled. Although you can see the control in the console, you can enable it only by activating AWS Backup capabilities for your landing zone.

AWS service: AWS Backup

Control metadata

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)

- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control group:**
- **Resource types:** `Multiple`

Usage considerations

- AWS Backup resources managed by AWS Control Tower should be modified using the AWS Control Tower API or console. API read actions for AWS Backup, such as `ListBackupPlans` and `GetBackupVaultAccessPolicy`, can be utilized directly.
- If you apply a tag with the key `aws-control-tower` to an AWS Backup resource created independently of AWS Control Tower, the resource becomes subject to this SCP.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Sid": "CTBACKUPPV1",
        "Effect": "Deny",
        "Action": [
          "backup:TagResource",
          "backup:UntagResource"
        ],
        "Resource": "*",
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalARN": "arn:*:iam::*:role/AWSControlTowerExecution"
          },
          "ForAnyValue:StringEquals": {
            "aws:TagKeys": "aws-control-tower"
          }
        }
```

```
      }
    ]
  }
```

[CT.BACKUP.PV.2] Disallow modification of an AWS Backup report plan that AWS Control Tower manages

This control limits changes to the AWS Backup report plan that AWS Control Tower manages.

This is a preventive control with elective guidance. By default, this control is not enabled. Although you can see the control in the console, you can enable it only by activating AWS Backup capabilities for your landing zone.

AWS service: AWS Backup

Control metadata

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control group:**
- **Resource types:** `AWS::Backup::ReportPlan`

Usage considerations

- AWS Backup resources managed by AWS Control Tower should be modified using the AWS Control Tower API or console. API read actions for AWS Backup, such as `ListBackupPlans` and `GetBackupVaultAccessPolicy`, can be utilized directly.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Sid": "CTBACKUPPV2",
        "Effect": "Deny",
        "Action": [
          "backup:CreateReportPlan",
          "backup:DeleteReportPlan",
          "backup:UpdateReportPlan"
        ],
        "Resource": "arn:*:backup:*:*:report-plan:aws_controltower_*",
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalARN": [
              "arn:*:iam::*:role/AWSControlTowerExecution"
            ]
          }
        }
      }
    ]
  }
```

[CT.BACKUP.PV.3] Disallow modification of an AWS Backup resource that AWS Control Tower manages

This control limits creation or modification of AWS Backup resources that AWS Control Tower manages.

This is a preventive control with elective guidance. By default, this control is not enabled. Although you can see the control in the console, you can enable it only by activating AWS Backup capabilities for your landing zone.

AWS service: AWS Backup

Control metadata

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control group:**
- **Resource types:** `AWS::Backup::BackupPlan`, `AWS::Backup::BackupVault`

Usage considerations

- AWS Backup resources managed by AWS Control Tower should be modified using the AWS Control Tower API or console. API read actions for AWS Backup, such as `ListBackupPlans` and `GetBackupVaultAccessPolicy`, can be utilized directly.
- If you apply a tag with the key `aws-control-tower` to an AWS Backup resource created independently of AWS Control Tower, the resource becomes subject to this SCP.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Sid": "CTBACKUPPV3",
        "Effect": "Deny",
        "Action": [
          "backup:CreateBackupPlan",
          "backup:CreateBackupSelection",
          "backup:CreateBackupVault",
          "backup:CreateLogicallyAirGappedBackupVault",
          "backup:DeleteBackupPlan",
          "backup:DeleteBackupSelection",
          "backup:DeleteBackupVault",
```

```
          "backup:DeleteBackupVaultAccessPolicy",
          "backup:DeleteBackupVaultLockConfiguration",
          "backup:DeleteBackupVaultSharingPolicy",
          "backup:PutBackupVaultAccessPolicy",
          "backup:PutBackupVaultLockConfiguration",
          "backup:PutBackupVaultSharingPolicy",
          "backup:UpdateBackupPlan"
        ],
        "Resource": [
          "arn:*:backup:*:*:backup-plan:*",
          "arn:*:backup:*:*:backup-vault:*"
        ],
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalARN": "arn:*:iam::*:role/AWSControlTowerExecution"
          },
          "Null": {
            "aws:ResourceTag/aws-control-tower": false
          }
        }
      }
    ]
  }
```

[CT.IAM.PV.1] Disallow modification of an AWS IAM role that AWS Control Tower utilizes to manage AWS Backup resources

This control limits modification of the AWS IAM role (aws-controltower-BackupRole) that AWS Control Tower utilizes for management of AWS Backup resources.

This is a preventive control with elective guidance. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Identity and Access Management (IAM)

Control metadata

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)

- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control group:**
- **Resource types:** `AWS::IAM::Role`

Usage considerations

- AWS Backup resources managed by AWS Control Tower should be modified using the AWS Control Tower API or console. API read actions for AWS Backup, such as `ListBackupPlans` and `GetBackupVaultAccessPolicy`, can be utilized directly.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Sid": "CTIAMPV1",
        "Effect": "Deny",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:DeleteRolePermissionsBoundary",
          "iam:DeleteRolePolicy",
          "iam:DetachRolePolicy",
          "iam:PutRolePermissionsBoundary",
          "iam:PutRolePolicy",
          "iam:UpdateAssumeRolePolicy",
          "iam:UpdateRole"
        ],
        "Resource": "arn:*:iam::*:role/aws-controltower-BackupRole",
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalARN": "arn:*:iam::*:role/AWSControlTowerExecution"
```

```
          }
        }
      }
    ]
  }
```

[CT.S3.PV.1] Disallow modification of an Amazon S3 bucket that stores AWS Backup reports for AWS Control Tower

This control limits modification of the Amazon S3 buckets that AWS Control Tower utilizes as a destination for storing AWS Backup reports.

This is a preventive control with elective guidance. By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control group:**
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- AWS Backup resources managed by AWS Control Tower should be modified using the AWS Control Tower API or console. API read actions for AWS Backup, such as `ListBackupPlans` and `GetBackupVaultAccessPolicy`, can be utilized directly.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Sid": "CTS3PV1",
        "Effect": "Deny",
        "Action": [
          "s3:CreateBucket",
          "s3:DeleteBucket",
          "s3:DeleteBucketPolicy",
          "s3:DeleteBucketWebsite",
          "s3:PutAccelerateConfiguration",
          "s3:PutBucketAcl",
          "s3:PutBucketCORS",
          "s3:PutBucketLogging",
          "s3:PutBucketObjectLockConfiguration",
          "s3:PutBucketOwnershipControls",
          "s3:PutBucketPolicy",
          "s3:PutBucketPublicAccessBlock",
          "s3:PutBucketVersioning",
          "s3:PutBucketWebsite",
          "s3:PutEncryptionConfiguration",
          "s3:PutLifecycleConfiguration",
          "s3:PutReplicationConfiguration"
        ],
        "Resource": "arn:*:s3:::aws-controltower-backup-reports-*",
        "Condition": {
          "ArnNotLike": {
            "aws:PrincipalARN": [
              "arn:*:iam::*:role/AWSControlTowerExecution"
            ]
          }
        }
      }
    ]
  }
```

Digital sovereignty controls

Digital sovereignty means control over digital assets. AWS Control Tower offers a group of controls that are designed to enhance your digital sovereignty governance posture. The pillars of this posture are as follows:

- *Data residency:* Control over the location of your data.

  For more information, see Controls that enhance data residency protection.
- *Granular access:* Access restrictions that limit all access to your data, unless the access is requested by you, or by a partner whom you trust.

  For more information, see Region deny control applied to the OU.
- *Encryption:* Features and controls that help you encrypt data, whether in transit, at rest, or in memory.

  For example, see the control CT.APPSYNC.PR.5: Require an AWS AppSync GraphQL API cache to have encryption at rest enabled.
- *Resiliency:* Ability to sustain operations through disruption or disconnection, which is essential in the case of events such as supply chain disruption, network interruption, and natural disaster.

  For example see the control CT.NETWORK-FIREWALL.PR.5: Require an AWS Network Firewall firewall to be deployed across multiple Availability Zones.

You can read more about digital sovereignty and AWS in the blog: AWS Digital Sovereignty Pledge: Control without compromise.

> **The Data residency subgroup**
>
> Although the digital sovereignty group is primarily a group of preventive controls, it includes *preventive* and *detective* controls in the **Data residency** subgroup.

Deny access to AWS based on the requested AWS Region

This control is commonly referred to as the Region deny control, or landing zone *Region deny control.*

This control disallows access to unlisted operations in global and regional services outside of the specified Regions. That includes all Regions where AWS Control Tower is not available, as well as

all Regions not selected for governance in the **Landing zone settings** page. Actions are allowed as usual in Regions with **Governed** status.

You may wish to review the information at Configure the Region deny control in the *AWS Control Tower User Guide* before you enable this control.

> **Note**
>
> Certain global AWS services, such as AWS Identity and Access Management (IAM) and AWS Organizations, are exempt from data residency controls. Those services are specified in the SCP example code that follows.

This is an elective control with preventive guidance. It is the top-level control associated with the **Region deny** action.

The format for this control is based on the following SCP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRREGIONDENY",
            "Effect": "Deny",
            "NotAction": [
                "a4b:*",
                "access-analyzer:*",
                "account:*",
                "acm:*",
                "activate:*",
                "artifact:*",
                "aws-marketplace-management:*",
                "aws-marketplace:*",
                "aws-portal:*",
                "billing:*",
                "billingconductor:*",
                "budgets:*",
                "ce:*",
                "chatbot:*",
```

```
                "chime:*",
                "cloudfront:*",
                "cloudtrail:LookupEvents",
                "compute-optimizer:*",
                "config:*",
                "consoleapp:*",
                "consolidatedbilling:*",
                "cur:*",
                "datapipeline:GetAccountLimits",
                "devicefarm:*",
                "directconnect:*",
                "ec2:DescribeRegions",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeVpnGateways",
                "ecr-public:*",
                "fms:*",
                "freetier:*",
                "globalaccelerator:*",
                "health:*",
                "iam:*",
                "importexport:*",
                "invoicing:*",
                "iq:*",
                "kms:*",
                "license-manager:ListReceivedLicenses",
                "lightsail:Get*",
                "mobileanalytics:*",
                "networkmanager:*",
                "notifications-contacts:*",
                "notifications:*",
                "organizations:*",
                "payments:*",
                "pricing:*",
                "quicksight:DescribeAccountSubscription",
                "resource-explorer-2:*",
                "route53-recovery-cluster:*",
                "route53-recovery-control-config:*",
                "route53-recovery-readiness:*",
                "route53:*",
                "route53domains:*",
                "s3:CreateMultiRegionAccessPoint",
                "s3:DeleteMultiRegionAccessPoint",
                "s3:DescribeMultiRegionAccessPointOperation",
                "s3:GetAccountPublicAccessBlock",
```

```
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:GetMultiRegionAccessPoint",
                "s3:GetMultiRegionAccessPointPolicy",
                "s3:GetMultiRegionAccessPointPolicyStatus",
                "s3:GetStorageLensConfiguration",
                "s3:GetStorageLensDashboard",
                "s3:ListAllMyBuckets",
                "s3:ListMultiRegionAccessPoints",
                "s3:ListStorageLensConfigurations",
                "s3:PutAccountPublicAccessBlock",
                "s3:PutMultiRegionAccessPointPolicy",
                "savingsplans:*",
                "shield:*",
                "sso:*",
                "sts:*",
                "support:*",
                "supportapp:*",
                "supportplans:*",
                "sustainability:*",
                "tag:GetResources",
                "tax:*",
                "trustedadvisor:*",
                "vendor-insights:ListEntitledSecurityProfiles",
                "waf-regional:*",
                "waf:*",
                "wafv2:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": []
                },
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:aws:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

Based on this example SCP format, AWS Control Tower adds your governed Regions into the `aws:RequestedRegion` statement. You cannot exclude your home Region. Actions not listed in the SCP are not permitted.

Limitations

The OU Region deny control is subject to limitations of the `aws:RequestedRegion` global condition key and Service Control Policies (SCPs) in general.

Region deny control applied to the OU

This control is commonly referred to as the OU Region deny control, or the configurable Region deny control.

With this control, you can restrict access to unlisted operations in global and regional AWS services outside of the specified Regions. You apply this control to an organizational unit (OU). You can apply this control to any subset of the Regions that are governed by your AWS Control Tower landing zone. You must always include your AWS Control Tower home Region.

You may wish to review the information at Configure the Region deny control before you enable this control.

Warning

If you enforce this control, the configurations for the OU can conflict with the landing zone version of this control. For more information, see the section called "Policy evaluation of SCP controls" in this chapter, and SCP evaluation in the AWS Organizations documentation.

CT.MULTISERVICE.PV.1: Deny access to AWS based on the requested AWS Region for an organizational unit

Service: Multiple AWS services

- **Control objective:** Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive

- **Control owner:** AWS Control Tower
- **Groups:** Digital sovereignty

> **Limitations**
>
> The OU Region deny control is subject to limitations of the `aws:RequestedRegion` global condition key and Service Control Policies (SCPs) in general.

Enable this control from the console

In the AWS Control Tower console, you can view the OUs on which this control is enabled, if any, by navigating to the **Control details** page for this control.

To enable this control from the Control details page

1. Select **Enable control** in the upper right
2. Select the target OU, then select **Next** to continue.
3. Select the Regions you wish to activate. You must select at least one Region.
4. You can add **NotAction** elements, IAM principals, and tags.
5. You'll be able to see a summary of your selected values before you enable the control.
6. Select **Enable control** at the lower right.

CT.MULTISERVICE.PV.1: Deny access to AWS based on the requested AWS Region for an organizational unit

The OU Region deny control, **CT.MULTISERVICE.PV.1**, is configurable. You can select specific OUs to which it applies, rather than applying it to your entire AWS Control Tower landing zone. This control accepts one or more parameters, such as **AllowedRegions**, **ExemptedPrincipalARNs**, and **ExemptedActions**, which describe operations that are allowed for accounts that are part of this OU.

- **AllowedRegions**: Specifies the Regions selected, in which the OU is allowed to operate. This parameter is **mandatory**.
- **ExemptedPrincipalARNs**: Specifies the IAM principals that are exempt from this control, so that they are allowed to operate certain AWS services globally.

- **ExemptedActions**: Specifies actions that are exempt from this control, so that the actions are allowed.

Interactions between the separate Region deny controls for the landing zone and the OU can be complicated to predict. They are predictable with the logic by which SCPs are evaluated by AWS.

Policy evaluation of SCP controls

The policy evaluation process involves checking all applicable policies, starting from the most permissive and gradually moving towards the most restrictive. Any SCP applied at the Root level will impact all accounts and OUs, unless it is overridden by a more specific policy.

Evaluation Logic: When a request is made to perform an action (for example, launching an Amazon EC2 instance), AWS evaluates policies to determine whether the action is allowed or denied. The evaluation logic follows these rules:

- *Explicit Deny Overrides All:* If any policy explicitly denies the requested action, that denial takes precedence over all other policies.
- *Explicit Allow Overrides Implicit Deny:* If a policy explicitly allows the action and no higher-level policy explicitly denies it, the action is allowed.
- *Inherited Allow and No Explicit Deny:* If there is no explicit allow or deny at the requested level, AWS looks at higher-level policies. If there is an inherited allow and no explicit deny, the action is allowed.
- *Explicit Deny at a Higher Level:* If there's an explicit deny in a higher-level policy, but no explicit allow or deny at the requested level, the action is denied.

For more information about the evaluation logic, see SCP evaluation in the AWS Organizations documentation.

> **Note**
>
> With this control, you can allow any AWS Region at the OU level, even if your landing zone does not govern that Region, by design. We recommend that you use caution when allowing Regions that your AWS Control Tower landing zone does not govern.

CLI Example

This example shows how to enable this control, with parameters, from the CLI.

```
aws controltower enable-control \
    --target-identifier arn:aws:organizations::01234567890:ou/o-EXAMPLE/ou-zzxx-
zzx0zzz2 \
    --control-identifier arn:aws:controltower:us-east-1::control/EXAMPLE_NAME \
    --parameters '[{"key":"AllowedRegions","value":["us-east-1","us-
west-2"]},{"key":"ExemptedPrincipalArns","value":["arn:aws:iam::*:role/
ReadOnly","arn:aws:sts::*:assumed-role/ReadOnly/*"]},{"key":"ExemptedActions","value":
["logs:DescribeLogGroups","logs:StartQuery","logs:GetQueryResults"]}]'
```

Validating parameters

When you enter a parameter into the OU Region deny control, AWS Control Tower validates the parameter's syntax and checks it against JSON datatypes. AWS Control Tower does not make semantic validations for domain-specific correctness. This is the same approach that is followed by AWS Organizations.

Parameters for this control are entered by means of a JSON schema.

Here is the SCP template of an example JSON schema for the OU-level Region deny control. In the AWS Control Tower console, you can view it on the **Artifacts** tab of the **Control details** page.

This short example schema shows that the **AllowedRegions**, **ExemptedActions** and **ExemptedPrincipalArns** parameters accept a list of strings. Also, you can add descriptions to the schema, or restrict allowed values to be a subset of pre-defined values, using enumerated types (enums).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTMULTISERVICEPV1",
            "Effect": "Deny",
            "NotAction": [
                {{ExemptedActions}}
                ...
                "s3:CreateMultiRegionAccessPoint",
                "s3:DeleteMultiRegionAccessPoint",
```

```
                "s3:DescribeMultiRegionAccessPointOperation",
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketLocation"
                ...
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": {{AllowedRegions}}
                },
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:aws:iam::*:role/AWSControlTowerExecution",
                        {{ExemptedPrincipalARNs}}
                    ]
                }
            }
        }
    ]
}
```

The following example shows a full SCP artifact for the control. It shows the actions and principals that are exempted by default when you apply this control to an OU. Remember that **AllowedRegions** is a mandatory parameter for this control. You can view the most recent version of this SCP in the AWS Control Tower console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTMULTISERVICEPV1",
            "Effect": "Deny",
            "NotAction": [
                {{ExemptedActions}}
                "a4b:*",
                "access-analyzer:*",
                "account:*",
                "acm:*",
                "activate:*",
                "artifact:*",
                "aws-marketplace-management:*",
                "aws-marketplace:*",
                "aws-portal:*",
```

```
                "billing:*",
                "billingconductor:*",
                "budgets:*",
                "ce:*",
                "chatbot:*",
                "chime:*",
                "cloudfront:*",
                "cloudtrail:LookupEvents",
                "compute-optimizer:*",
                "config:*",
                "consoleapp:*",
                "consolidatedbilling:*",
                "cur:*",
                "datapipeline:GetAccountLimits",
                "devicefarm:*",
                "directconnect:*",
                "ec2:DescribeRegions",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeVpnGateways",
                "ecr-public:*",
                "fms:*",
                "freetier:*",
                "globalaccelerator:*",
                "health:*",
                "iam:*",
                "importexport:*",
                "invoicing:*",
                "iq:*",
                "kms:*",
                "license-manager:ListReceivedLicenses",
                "lightsail:Get*",
                "mobileanalytics:*",
                "networkmanager:*",
                "notifications-contacts:*",
                "notifications:*",
                "organizations:*",
                "payments:*",
                "pricing:*",
                "quicksight:DescribeAccountSubscription",
                "resource-explorer-2:*",
                "route53-recovery-cluster:*",
                "route53-recovery-control-config:*",
                "route53-recovery-readiness:*",
                "route53:*",
```

```
                "route53domains:*",
                "s3:CreateMultiRegionAccessPoint",
                "s3:DeleteMultiRegionAccessPoint",
                "s3:DescribeMultiRegionAccessPointOperation",
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:GetMultiRegionAccessPoint",
                "s3:GetMultiRegionAccessPointPolicy",
                "s3:GetMultiRegionAccessPointPolicyStatus",
                "s3:GetStorageLensConfiguration",
                "s3:GetStorageLensDashboard",
                "s3:ListAllMyBuckets",
                "s3:ListMultiRegionAccessPoints",
                "s3:ListStorageLensConfigurations",
                "s3:PutAccountPublicAccessBlock",
                "s3:PutMultiRegionAccessPointPolicy",
                "savingsplans:*",
                "shield:*",
                "sso:*",
                "sts:*",
                "support:*",
                "supportapp:*",
                "supportplans:*",
                "sustainability:*",
                "tag:GetResources",
                "tax:*",
                "trustedadvisor:*",
                "vendor-insights:ListEntitledSecurityProfiles",
                "waf-regional:*",
                "waf:*",
                "wafv2:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": {{AllowedRegions}}
                },
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution",
                        "arn:*:iam::*:role/aws-controltower-ConfigRecorderRole",
```

```
                        "arn:*:iam::*:role/aws-controltower-
ForwardSnsNotificationRole",
                        "arn:*:iam::*:role/AWSControlTower_VPCFlowLogsRole"
                    ]
                }
            }
        }
    ]
}
```

Preventive controls that assist with digital sovereignty

These preventive controls are designed to assist you with your digital sovereignty governance posture.

This group of controls helps you comply with digital sovereignty regulatory requirements because they prevent actions, enforce configurations, and detect resource changes that affect data residency, granular access restriction, encryption, and resilience capabilities.

- These controls are configurable. For more information about configurable controls, see Controls with parameters.
- These are optional controls with Preventive guidance, implemented with AWS service control policies (SCPs). They are not deployed on any OU by default. You can enable them through the AWS Control Tower console, or through the AWS Control Tower APIs

In the AWS Control Tower console, you can view these controls together under the **Groups** tab on the **Categories** page.

Topics

- [CT.APPSYNC.PV.1] Require an AWS AppSync GraphQL API to be configured with private visibility
- [CT.EC2.PV.1] Require an Amazon EBS snapshot to be created from an encrypted EC2 volume
- [CT.EC2.PV.2] Require that an attached Amazon EBS volume is configured to encrypt data at rest
- [CT.EC2.PV.3] Require that an Amazon EBS snapshot cannot be publicly restorable
- [CT.EC2.PV.4] Require that Amazon EBS direct APIs are not called
- [CT.EC2.PV.5] Disallow the use of Amazon EC2 VM import and export

- [CT.EC2.PV.6] Disallow the use of deprecated Amazon EC2 RequestSpotFleet and RequestSpotInstances API actions
- [CT.KMS.PV.1] Require an AWS KMS key policy to have a statement that limits creation of AWS KMS grants to AWS services
- [CT.KMS.PV.2] Require that an AWS KMS asymmetric key with RSA key material used for encryption does not have a key length of 2048 bits
- [CT.KMS.PV.3] Require that an AWS KMS key is configured with the bypass policy lockout safety check enabled
- [CT.KMS.PV.4] Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from AWS CloudHSM
- [CT.KMS.PV.5] Require that an AWS KMS customer-managed key (CMK) is configured with imported key material
- [CT.KMS.PV.6] Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from an external key store (XKS)
- [CT.LAMBDA.PV.1] Require an AWS Lambda function URL to use AWS IAM-based authentication
- [CT.LAMBDA.PV.2] Require an AWS Lambda function or AWS Lambda function URL to be configured for access only to principals within your AWS account

[CT.APPSYNC.PV.1] Require an AWS AppSync GraphQL API to be configured with private visibility

This control disallows creation of public AWS AppSync APIs by requiring APIs to be configured with private visibility.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS AppSync

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower

- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::AppSync::GraphQLApi`

Usage considerations

- This control requires AppSync GraphQL APIs to be created with a private API configuration to ensure that the API is accessible only from a VPC.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTAPPSYNCPV1",
            "Effect": "Deny",
            "Action": "appsync:CreateGraphqlApi",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "appsync:Visibility": "PRIVATE"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.EC2.PV.1] Require an Amazon EBS snapshot to be created from an encrypted EC2 volume

This control disallows creation of new snapshots that are based on unencrypted EBS volumes.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Encrypt data at rest
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::::Account`

Usage considerations

- This control does not prevent creation of unencrypted EBS snapshots that are created by means of the `CopySnapshot` operation. AWS Control Tower recommends that you enable EBS encryption by default, so that encryption is applied to copies of unencrypted snapshots. See Encryption scenarios in the *Amazon EC2 User Guide for Linux Instances* for more information.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV1",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:CreateSnapshots"
            ],
            "Resource": "arn:*:ec2:*:*:volume/*",
            "Condition": {
                "Bool": {
                    "ec2:Encrypted": "false"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.EC2.PV.2] Require that an attached Amazon EBS volume is configured to encrypt data at rest

This control disallows attaching an unencrypted EBS volume to a running or stopped EC2 instance.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Encrypt data at rest
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower

- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::EC2::Volume`

Usage considerations

- This control does not prevent replacing an EBS-backed root volume for a running instance with an unencrypted volume, by means of the `CreateReplaceRootVolumeTask` operation.
- AWS Control Tower recommends that you enable EBS encryption by default. For information about EBS encryption by default, see Encryption by default in the *Amazon EC2 User Guide for Linux Instances*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV2",
            "Effect": "Deny",
            "Action": "ec2:AttachVolume",
            "Resource": "arn:*:ec2:*:*:volume/*",
            "Condition": {
                "Bool": {
                    "ec2:Encrypted": "false"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.EC2.PV.3] Require that an Amazon EBS snapshot cannot be publicly restorable

This control disallows sharing of an EBS snapshot with all AWS accounts.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::::Account`

Usage considerations

- This control prevents unencrypted EBS snapshots from being made public, by disallowing sharing of EBS snapshots with all AWS accounts. Encrypted snapshots and snapshots with AWS Marketplace product codes cannot be made public.
- To prevent the public sharing of snapshots, AWS Control Tower recommends enabling block public access for snapshots. For more information, see Block public access for snapshots in the *Amazon EC2 User Guide for Linux Instances*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV3",
            "Effect": "Deny",
            "Action": "ec2:ModifySnapshotAttribute",
            "Resource": "arn:*:ec2:*::snapshot/*",
            "Condition": {
                "StringEquals": {
                    "ec2:Add/group": "all"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.EC2.PV.4] Require that Amazon EBS direct APIs are not called

This control disallows usage of all EBS direct APIs.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::::Account`

Usage considerations

- Do not enable this control if you use EBS direct APIs, either directly or through an AWS Backup partner product.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV4",
            "Effect": "Deny",
            "Action": "ebs:*",
            "Resource": "*"{% if ExemptedPrincipalArns %},
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[CT.EC2.PV.5] Disallow the use of Amazon EC2 VM import and export

This control disallows use of EC2 VM Import/Export APIs that can be used to import and export EC2 instance, snapshot, image and volume data.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::::Account`

Usage considerations

- This control disallows the use of VM Import/Export APIs that can be used to import and export EC2 image, snapshot, instance and volume data. If you need to use VM Import/Export functionality, do not enable this control.
- This control does not prevent cancelling existing VM Import/Export import, export or conversion tasks.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV5",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateInstanceExportTask",
                "ec2:ExportImage",
                "ec2:ImportImage",
                "ec2:ImportSnapshot",
                "ec2:ImportInstance",
                "ec2:ImportVolume"
            ],
            "Resource": "*"{% if ExemptedPrincipalArns %},
```

```
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[CT.EC2.PV.6] Disallow the use of deprecated Amazon EC2 RequestSpotFleet and RequestSpotInstances API actions

This control disallows usage of EC2 `RequestSpotFleet` and `RequestSpotInstances` APIs, because they are legacy APIs with no planned investment.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::EC2::SpotFleet`

Usage considerations

- This control denies `ec2:RequestSpotFleet` and `ec2:RequestSpotInstances` actions for all IAM principals. If you need to use these actions, do not enable this control.
- This control does not prevent cancelling or modifying an existing spot fleet or spot instance request.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTEC2PV6",
            "Effect": "Deny",
            "Action": [
                "ec2:RequestSpotFleet",
                "ec2:RequestSpotInstances"
            ],
            "Resource": "*"{% if ExemptedPrincipalArns %},
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[CT.KMS.PV.1] Require an AWS KMS key policy to have a statement that limits creation of AWS KMS grants to AWS services

This control requires that KMS grants are issued only to AWS services that are integrated with AWS KMS, or to an AWS service principal.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive

- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control disallows the creation of AWS KMS grants for your KMS keys if the request does not originate from an AWS service that's integrated with AWS KMS, or from an AWS service principal.
- If you need to issue AWS KMS grants directly to your IAM principals for a customer-managed key, do not enable this control.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV1",
            "Effect": "Deny",
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "kms:GrantIsForAWSResource": "false",
                    "aws:PrincipalIsAWSService": "false"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
```

```
    ]
}
```

[CT.KMS.PV.2] Require that an AWS KMS asymmetric key with RSA key material used for encryption does not have a key length of 2048 bits

This control disallows the creation of KMS keys used for encryption and decryption that also have a key spec of `RSA_2048`.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Encrypt data at rest, Encrypt data in transit
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control requires that you use a `KeySpec` other than `RSA_2048` when creating asymmetric KMS keys used for encryption and decryption.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV2",
            "Effect": "Deny",
            "Action": "kms:CreateKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:KeyUsage": "ENCRYPT_DECRYPT",
                    "kms:KeySpec": "RSA_2048"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.KMS.PV.3] Require that an AWS KMS key is configured with the bypass policy lockout safety check enabled

This control disallows bypassing the KMS key policy lockout safety check when creating a KMS key or updating its key policy.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Enforce least privilege, Protect configurations
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive

- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control disallows bypassing the policy lockout safety check, because bypassing this check increases the risk that a KMS key becomes unmanageable.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV3",
            "Effect": "Deny",
            "Action": [
                "kms:CreateKey",
                "kms:PutKeyPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:BypassPolicyLockoutSafetyCheck": "true"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.KMS.PV.4] Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from AWS CloudHSM

This control disallows creation of KMS keys that do not have a key origin of `AWS_CLOUDHSM`.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Encrypt data at rest, Encrypt data in transit
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control restricts creation of AWS KMS keys to those that use a specific key material origin. It is suitable when enforcing a KMS key management strategy that requires all KMS keys to an AWS CloudHSM based custom key store.
- Before enforcing the exclusive use of keys whose key material resides in an AWS CloudHSM cluster, carefully evaluate the trade-offs documented in the AWS CloudHSM key stores section of the *AWS KMS Developer Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV4",
            "Effect": "Deny",
            "Action": "kms:CreateKey",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "kms:KeyOrigin": "AWS_CLOUDHSM"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.KMS.PV.5] Require that an AWS KMS customer-managed key (CMK) is configured with imported key material

This control disallows creation of KMS keys that do not have a key origin of `EXTERNAL`.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Encrypt data at rest, Encrypt data in transit
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower

- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control restricts creation of KMS keys to those that use a specific key material origin. It is suitable when enforcing a KMS key management strategy that requires all KMS keys to use imported key material.
- Before enforcing the exclusive use of keys with imported key material, carefully evaluate the trade-offs documented in the Importing key material for AWS KMS keys section of the *AWS KMS Developer Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV5",
            "Effect": "Deny",
            "Action": "kms:CreateKey",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "kms:KeyOrigin": "EXTERNAL"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.KMS.PV.6] Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from an external key store (XKS)

This control disallows creation of KMS keys that do not have a key origin of `EXTERNAL_KEY_STORE`.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Key Management Service (AWS KMS)

Control metadata

- **Control objective:** Encrypt data at rest, Encrypt data in transit
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::KMS::Key`

Usage considerations

- This control restricts creation of AWS KMS keys to those that use a specific key material origin. It is suitable when enforcing a KMS key management strategy that requires all KMS keys to an external key store custom key store.
- Before enforcing the exclusive use of keys whose key material resides in an external key store, carefully evaluate the trade-offs documented in the External key stores section of the *AWS KMS Developer Guide*.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTKMSPV6",
            "Effect": "Deny",
            "Action": "kms:CreateKey",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "kms:KeyOrigin": "EXTERNAL_KEY_STORE"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.LAMBDA.PV.1] Require an AWS Lambda function URL to use AWS IAM-based authentication

Require an AWS Lambda function URL to restrict access to authenticated users by using `AWS_IAM` based authentication.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Lambda

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive

- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::Lambda::Url`

Usage considerations

- This control disallows creation and update of AWS Lambda function URL configurations. It does not prevent deletion of Lambda function URL configurations.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTLAMBDAPV1",
            "Effect": "Deny",
            "Action": [
                "lambda:CreateFunctionUrlConfig",
                "lambda:UpdateFunctionUrlConfig"
            ],
            "Resource": "arn:*:lambda:*:*:function:*",
            "Condition": {
                "StringNotEquals": {
                    "lambda:FunctionUrlAuthType": "AWS_IAM"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.LAMBDA.PV.2] Require an AWS Lambda function or AWS Lambda function URL to be configured for access only to principals within your AWS account

This control requires an AWS Lambda function resource-based policy to grant access only to IAM principals that reside in your AWS account.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Lambda

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::Lambda::Url`, `AWS::Lambda::Function`

Usage considerations

- This control limits cross-account access to AWS Lambda functions by restricting the allowed IAM principals in a Lambda function resource policy to those in the same account as the Lambda function. Allow listing AWS service principals is not supported by this control.
- Permissions to AWS Lambda functions and related URL(s) are governed by this control.
- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CTLAMBDAPV2",
            "Effect": "Deny",
            "Action": "lambda:AddPermission",
            "Resource": "arn:*:lambda:*:*:function:*",
            "Condition": {
                "StringNotLike": {
                    "lambda:Principal": [
                        "arn:*:iam::${aws:PrincipalAccount}:*",
                        "${aws:PrincipalAccount}"
                    ]
                },
                "ArnNotLike": {
                    "aws:PrincipalArn": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

Controls that enhance data residency protection

These elective controls complement your enterprise's data residency posture. By applying these controls together, you can set up your multi-account environment to help detect and inhibit the purposeful or accidental creation, sharing, or copying of data, outside of your selected AWS Region or Regions.

These controls take effect at the OU level, and they apply to all member accounts within the OU.

Important

Certain global AWS services, such as AWS Identity and Access Management (IAM) and AWS Organizations, are exempt from these controls. You can identify the services that are exempt by reviewing the **Region deny SCP**, shown in the example code. Services with "*" after their identifier are exempt, because all actions are permitted when the "*" notation is given. This SCP essentially contains a list of explicitly permitted actions, and all other actions are denied. You cannot deny access to your home Region.

Video: Enable data residency controls

This video (5:58) describes how to enable data residency controls with AWS Control Tower controls. For better viewing, select the icon at the lower right corner of the video to enlarge it to full screen. Captioning is available.

Note

AWS Control Tower no longer supports searching the controls list by *Category*, as shown in this video. To easily identify the Data Residency controls, we recommend you sort the controls list by *Release Date*. Controls with a release date of November 30, 2021 are the same controls in the Data Residency category shown in the video.
This video includes the term *guardrail*, an older term AWS Control Tower used for *control*. We updated the term to better align with industry usage and other AWS services. These terms are synonymous for our purposes.

Video Walkthrough of Enabling Data Residency Controls in AWS Control Tower.

Topics

- Data residency controls with preventive behavior
- Data residency controls with detective behavior

Data residency controls with preventive behavior

The following data residency controls have preventive behavior.

Topics

- Disallow internet access for an Amazon VPC instance managed by a customer
- Disallow Amazon Virtual Private Network (VPN) connections
- Disallow cross-region networking for Amazon EC2, Amazon CloudFront, and AWS Global Accelerator

Disallow internet access for an Amazon VPC instance managed by a customer

This control disallows internet access for an Amazon Virtual Private Cloud (VPC) instance managed by a customer, rather than by an AWS service.

> **Important**
>
> If you provision Account Factory accounts with VPC internet access settings enabled, that Account Factory setting overrides this control. To avoid enabling internet access for newly provisioned accounts, you must change the setting in Account Factory. For more information, see Configure AWS Control Tower Without a VPC.

- This control does not apply to VPCs managed by AWS services.
- Existing VPCs that have internet access retain their internet access. It applies to new instances only. After this control is applied, access cannot be changed.

This is a preventive control with elective guidance. By default, this control isn't enabled on any OUs.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRDISALLOWVPCINTERNETACCESS",
      "Effect": "Deny",
      "Action": [
        "ec2:CreateInternetGateway",
```

```
        "ec2:AttachInternetGateway",
        "ec2:CreateEgressOnlyInternetGateway",
        "ec2:CreateDefaultVpc",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateCarrierGateway"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::*:role/AWSControlTowerExecution"
          ]
        }
      }
    }
  ]
}
```

Disallow Amazon Virtual Private Network (VPN) connections

This control prevents Virtual Private Network (VPN) connections (Site-to-Site VPN and Client VPN) to an Amazon Virtual Private Cloud (VPC).

> **Note**
>
> Existing VPCs that have internet access retain their internet access.

This is a preventive control with elective guidance. By default, this control isn't enabled on any OUs.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "GRDISALLOWVPNCONNECTIONS",
      "Effect": "Deny",
      "Action": [
        "ec2:CreateVPNGateway",
        "ec2:AttachVPNGateway",
        "ec2:CreateCustomerGateway",
        "ec2:CreateVpnConnection",
        "ec2:ModifyVpnConnection",
        "ec2:CreateClientVpnEndpoint",
        "ec2:ModifyClientVpnEndpoint",
        "ec2:AssociateClientVpnTargetNetwork",
        "ec2:AuthorizeClientVpnIngress"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Disallow cross-region networking for Amazon EC2, Amazon CloudFront, and AWS Global Accelerator

This control prevents configuring cross-region networking connections from Amazon EC2, Amazon CloudFront, and AWS Global Accelerator services. It prevents VPC peering and transit gateway peering.

> **Note**
>
> This control prevents Amazon EC2 VPC peering and Amazon EC2 transit gateway peering within a single Region, as well as across Regions. For this reason, this control may affect certain workloads in addition to your data residency posture.

This is a preventive control with elective guidance. By default, this control isn't enabled on any OUs.

The artifact for this control is the following service control policy (SCP).

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GRDISALLOWCROSSREGIONNETWORKING",
      "Effect": "Deny",
      "Action": [
        "ec2:CreateVpcPeeringConnection",
        "ec2:AcceptVpcPeeringConnection",
        "ec2:CreateTransitGatewayPeeringAttachment",
        "ec2:AcceptTransitGatewayPeeringAttachment",
        "cloudfront:CreateDistribution",
        "cloudfront:UpdateDistribution",
        "globalaccelerator:Create*",
        "globalaccelerator:Update*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Data residency controls with detective behavior

The following data residency controls have detective behavior.

Topics

- Detect whether public IP addresses for Amazon EC2 autoscaling are enabled through launch configurations
- Detect whether replication instances for AWS Database Migration Service are public
- Detect whether Amazon EBS snapshots are restorable by all AWS accounts
- Detect whether any Amazon EC2 instance has an associated public IPv4 address
- Detect whether Amazon S3 settings to block public access are set as true for the account
- Detects whether an Amazon EKS endpoint is blocked from public access
- Detect whether an Amazon OpenSearch Service domain is in Amazon VPC

- Detect whether any Amazon EMR cluster master nodes have public IP addresses
- Detect whether the AWS Lambda function policy attached to the Lambda resource blocks public access
- Detect whether public routes exist in the route table for an Internet Gateway (IGW)
- Detect whether Amazon Redshift clusters are blocked from public access
- Detect whether an Amazon SageMaker notebook instance allows direct internet access
- Detect whether any Amazon VPC subnets are assigned a public IP address
- Detect whether AWS Systems Manager documents owned by the account are public

Detect whether public IP addresses for Amazon EC2 autoscaling are enabled through launch configurations

This control detects whether Amazon EC2 Auto Scaling groups have public IP addresses enabled through launch configurations.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the launch configuration for an autoscaling group sets the value of the field `AssociatePublicIpAddress` set as **True**.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether public IP addresses for Amazon
 EC2 Auto Scaling are enabled through launch configurations

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  AutoscalingLaunchConfigPublicIpDisabled:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
```

```
     Description: Detects whether Amazon EC2 Auto Scaling groups have public IP
 addresses enabled through launch configurations. This rule is NON_COMPLIANT if
 the launch configuration for an Auto Scaling group has the value of the field
 AssociatePublicIpAddress set as True.
     Scope:
       ComplianceResourceTypes:
         - AWS::AutoScaling::LaunchConfiguration
     Source:
       Owner: AWS
       SourceIdentifier: AUTOSCALING_LAUNCH_CONFIG_PUBLIC_IP_DISABLED
```

Detect whether replication instances for AWS Database Migration Service are public

This control detects whether AWS Database Migration Service replication instances are public.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the value of the `PubliclyAccessible` field is set as **True**.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether replication instances for AWS
 Database Migration Service are public

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours
```

```
Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  DmsReplicationNotPublic:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether AWS Database Migration Service replication instances
 are public. The rule is NON_COMPLIANT if the value of the PubliclyAccessible field is
 set as True.
      Source:
        Owner: AWS
        SourceIdentifier: DMS_REPLICATION_NOT_PUBLIC
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether Amazon EBS snapshots are restorable by all AWS accounts

This control detects whether all AWS accounts have access to restore Amazon EBS snapshots.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if any snapshots have the `RestorableByUserIds` field set to the value **All**. In that case, the Amazon EBS snapshots are public.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether Amazon EBS snapshots are
 restorable by all AWS accounts
```

```
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  EbsSnapshotPublicRestorableCheck:
    Type: AWS::Config::ConfigRule

    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether all AWS accounts have access to restore Amazon EBS
 snapshots. The rule is NON_COMPLIANT if any snapshots have the RestorableByUserIds
 field set to the value All. In that case, the Amazon EBS snapshots are public.
      Source:
        Owner: AWS
        SourceIdentifier: EBS_SNAPSHOT_PUBLIC_RESTORABLE_CHECK
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether any Amazon EC2 instance has an associated public IPv4 address

This control detects whether an Amazon Elastic Compute Cloud (Amazon EC2) instance has an associated public IPv4 address. This control applies only to IPv4 addresses.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the public IP field is present in the Amazon EC2 instance configuration item.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether any Amazon EC2 instance has an
 associated public IPv4 address

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  Ec2InstanceNoPublicIp:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether an Amazon Elastic Compute Cloud (Amazon EC2)
 instance has an associated public IPv4 address. The rule is NON_COMPLIANT if the
 public IP field is present in the Amazon EC2 instance configuration item.
      Scope:
        ComplianceResourceTypes:
        - AWS::EC2::Instance
      Source:
        Owner: AWS
        SourceIdentifier: EC2_INSTANCE_NO_PUBLIC_IP
```

Detect whether Amazon S3 settings to block public access are set as true for the account

This control periodically detects whether the required Amazon S3 settings to block public access are configured as true for the account, rather than for a bucket or an access point.

In the console:

- The rule shows **Non-compliant** status if at least one of the settings is false.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to check whether Amazon S3 settings to block
 public access are set as true for the account.

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  PublicAccessBlockSetting:
    Type: 'String'
    Default: 'True'
  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  CheckForS3PublicAccessBlock:
    Type: AWS::Config::ConfigRule
    Properties:
```

```
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks the Amazon S3 settings to block public access are set as true
 for the account. The rule is non-compliant if at-least one of the settings is false.
      Source:
        Owner: AWS
        SourceIdentifier: S3_ACCOUNT_LEVEL_PUBLIC_ACCESS_BLOCKS_PERIODIC
      Scope:
        ComplianceResourceTypes:
          - AWS::S3::AccountPublicAccessBlock
      InputParameters:
        IgnorePublicAcls: !Ref PublicAccessBlockSetting
        BlockPublicPolicy: !Ref PublicAccessBlockSetting
        BlockPublicAcls: !Ref PublicAccessBlockSetting
        RestrictPublicBuckets: !Ref PublicAccessBlockSetting
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detects whether an Amazon EKS endpoint is blocked from public access

This control detects whether an Amazon Elastic Kubernetes Service (Amazon EKS) endpoint is blocked from public access.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the endpoint is publicly accessible.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether an Amazon EKS endpoint is
 blocked from public access

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
```

```
  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
 EKSEndpointNoPublicAccess:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether an Amazon Elastic Kubernetes Service (Amazon EKS)
 endpoint is publicly accessible. The rule is NON_COMPLIANT if the endpoint is publicly
 accessible.
      Source:
        Owner: AWS
        SourceIdentifier: EKS_ENDPOINT_NO_PUBLIC_ACCESS
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether an Amazon OpenSearch Service domain is in Amazon VPC

This control detects whether an Amazon OpenSearch Service domain is in Amazon VPC.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the OpenSearch Service domain endpoint is public.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether an Amazon OpenSearch Service
 domain is in Amazon VPC

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  ElasticsearchInVpcOnly:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether Amazon OpenSearch Service domains are in Amazon
 Virtual Private Cloud (Amazon VPC). The rule is NON_COMPLIANT if the OpenSearch
 Service domain endpoint is public.
```

```
      Source:
        Owner: AWS
        SourceIdentifier: ELASTICSEARCH_IN_VPC_ONLY
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether any Amazon EMR cluster master nodes have public IP addresses

This control detects whether any Amazon EMR cluster master nodes have public IP addresses.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs

In the console:

- The rule shows **Non-compliant** status if a master node has a public IP address.
- This control checks clusters that are in RUNNING or WAITING state.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether any Amazon EMR cluster master
 nodes have public IP addresses

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
```

```
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  EmrMasterNoPublicIp:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether any Amazon Elastic MapReduce (EMR) cluster master
 nodes have public IP addresses. The rule is NON_COMPLIANT if a master node has a
 public IP. This control checks clusters that are in RUNNING or WAITING state.
      Source:
        Owner: AWS
        SourceIdentifier: EMR_MASTER_NO_PUBLIC_IP
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether the AWS Lambda function policy attached to the Lambda resource blocks public access

This control detects whether the AWS Lambda function policy attached to the Lambda resource blocks public access.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the Lambda function policy allows public access.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether the AWS Lambda function policy
 attached to the Lambda resource blocks public access
```

```
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  LambdaFunctionPublicAccessProhibited:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether the AWS Lambda function policy attached to the
 Lambda resource prohibits public access. The rule is NON_COMPLIANT if the Lambda
 function policy allows public access.
      Scope:
        ComplianceResourceTypes:
        - AWS::Lambda::Function
      Source:
        Owner: AWS
        SourceIdentifier: LAMBDA_FUNCTION_PUBLIC_ACCESS_PROHIBITED
```

Detect whether public routes exist in the route table for an Internet Gateway (IGW)

This control detects whether public routes exist in the route table associated with an Internet Gateway (IGW).

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if a route has a destination CIDR block of `0.0.0.0/0` or `::/0` or if a destination CIDR block does not match the rule parameter.

> **Note**
>
> This control fails if any of the routes to an IGW has a destination CIDR block of `0.0.0.0/0` or `::/0`.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Configure AWS Config rule to detect whether public routes exist in the
 route table for an Internet Gateway (IGW)

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  NoUnrestrictedRouteToIgw:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether public routes exist in the route table associated
 with an Internet Gateway (IGW). The rule is NON_COMPLIANT if a route has a destination
 CIDR block of '0.0.0.0/0' or '::/0' or if a destination CIDR block does not match the
 rule parameter.
      Scope:
        ComplianceResourceTypes:
        - AWS::EC2::RouteTable
      Source:
        Owner: AWS
        SourceIdentifier: NO_UNRESTRICTED_ROUTE_TO_IGW
```

Detect whether Amazon Redshift clusters are blocked from public access

This control detects whether Amazon Redshift clusters are blocked from public access.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the `publiclyAccessible` field is set to **True** in the cluster configuration item.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether Amazon Redshift clusters are
 blocked from public access

Parameters:
```

```
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  RedshiftClusterPublicAccessCheck:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether Amazon Redshift clusters are blocked from public
 access. The rule is NON_COMPLIANT if the publiclyAccessible field is true in the
 cluster configuration item.
      Scope:
        ComplianceResourceTypes:
        - AWS::Redshift::Cluster
      Source:
        Owner: AWS
        SourceIdentifier: REDSHIFT_CLUSTER_PUBLIC_ACCESS_CHECK
```

Detect whether an Amazon SageMaker notebook instance allows direct internet access

This control detects whether an Amazon SageMaker notebook instance allows direct internet access.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if Amazon SageMaker notebook instances allow direct internet access.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rule to detect whether an Amazon SageMaker notebook
 instance allows direct internet access

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
```

```
  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  SagemakerNotebookNoDirectInternetAccess:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether direct internet access is allowed for an Amazon
 SageMaker notebook instance. The rule is NON_COMPLIANT if Amazon SageMaker notebook
 instances allow direct internet access.
      Source:
        Owner: AWS
        SourceIdentifier: SAGEMAKER_NOTEBOOK_NO_DIRECT_INTERNET_ACCESS
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
        - !Ref MaximumExecutionFrequency
```

Detect whether any Amazon VPC subnets are assigned a public IP address

This control detects whether Amazon Virtual Private Cloud (Amazon VPC) subnets are assigned a public IP address.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console:

- The rule shows **Non-compliant** status if the Amazon VPC has subnets that are assigned a public IP address.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Detect whether any Amazon VPC subnets are assigned a public IP address

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  SubnetAutoAssignPublicIpDisabled:
    Type: AWS::Config::ConfigRule

    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether Amazon Virtual Private Cloud (Amazon VPC) subnets
 are assigned a public IP address. The rule is NON_COMPLIANT if Amazon VPC has subnets
 that are assigned a public IP address.
      Scope:
        ComplianceResourceTypes:
        - AWS::EC2::Subnet
      Source:
        Owner: AWS
        SourceIdentifier: SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED
```

Detect whether AWS Systems Manager documents owned by the account are public

This control detects whether AWS Systems Manager documents owned by the account are public.

This is a detective control with elective guidance. By default, this control isn't enabled on any OUs.

In the console: The rule shows **Non-compliant** status if any documents with owner 'Self' are public.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Configure AWS Config rule to detect whether AWS Systems Manager documents
 owned by the account are public

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency at which AWS Config will run evaluations for the rule.
    AllowedValues:
      - 1hour
      - 3hours
      - 6hours
      - 12hours
      - 24hours

Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours

Resources:
  SsmDocumentNotPublic:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether AWS Systems Manager (SSM) documents owned by the
 account are public. This rule is NON_COMPLIANT if any documents with owner 'Self' are
 public.
      Source:
        Owner: AWS
        SourceIdentifier: SSM_DOCUMENT_NOT_PUBLIC
      MaximumExecutionFrequency:
        !FindInMap
        - Settings
        - FrequencyMap
```

```
        - !Ref MaximumExecutionFrequency
```

Detective controls

A detective control detects noncompliance of resources within your accounts, such as policy violations, and provides alerts through the dashboard. The status of a detective control is either **clear**, **in violation**, or **not enabled**. Detective controls apply only in those AWS Regions supported by AWS Control Tower.

- Detective controls are implemented using AWS Config rules. Most of the **Strongly recommended** controls, and many of the **Elective** controls, that are owned by AWS Control Tower are detective controls. The name of these controls typically begins with the word *Detect*, to denote a detective control.
- The integrated, detective Security Hub CSPM controls are implemented using AWS Config rules, similarly to all Security Hub CSPM controls. These controls are owned by the **Service-Managed Standard: AWS Control Tower**, which is part of Security Hub CSPM.
- Certain AWS Config controls are manageable directly from the AWS Control Tower console, implemented with AWS Config rules.
- When you enable controls on an organizational unit (OU) that is registered with AWS Control Tower, detective controls apply to enrolled accounts only, not to all member accounts in the OU, if some accounts are not enrolled in AWS Control Tower.

> **Note**
>
> For information about how detective controls are applied to nested OUs, in AWS Control Tower, see Nested Ous and controls.

More about detective controls

Most of the AWS Control Tower **Strongly recommended** controls are detective. By default, **Strongly recommended** controls are not enabled. For more information, see Strongly recommended controls.

Three of the AWS Control Tower **Elective** controls are detective. By default, **Elective** controls are not enabled. For more information, see Elective controls.

Detective controls with Elective guidance

- Detect Whether MFA is Enabled for AWS IAM Users
- Detect Whether MFA is Enabled for AWS IAM Users of the AWS Console
- Detect Whether Versioning for Amazon S3 Buckets is Enabled

The integrated AWS Config controls in AWS Control Tower have **Elective** guidance. For more information, see Integrated AWS Config controls available in AWS Control Tower.

Topics

- The Security Hub CSPM standard
- Integrated AWS Config controls available in AWS Control Tower

The Security Hub CSPM standard

AWS Control Tower is integrated with AWS Security Hub CSPM to provide detective controls that help you monitor your AWS environment. The integration is accomplished with a Security Hub CSPM standard, called the **Service-Managed Standard: AWS Control Tower**.

The **Service-Managed Standard: AWS Control Tower** supports a subset of controls in the **AWS Foundational Security Best Practices (FSBP)** standard. To learn more about this standard and to view the available controls, see Service-Managed Standard: AWS Control Tower. For more general information about Security Hub CSPM standards, see Security standards and controls in Security Hub CSPM, in the *AWS Security Hub User Guide*.

This standard is available only for AWS Control Tower customers who have created the standard in the AWS Control Tower console. AWS Control Tower creates the standard for you when you enable the first Security Hub CSPM control in the AWS Control Tower console. When you enable the first control, if you haven't already enabled Security Hub CSPM, AWS Control Tower also enables Security Hub CSPM for you.

After you create this standard, you can view the Security Hub CSPM detective controls alongside other AWS Control Tower controls, in the AWS Control Tower console and in Security Hub CSPM.

Control behavior

- No controls are enabled automatically when you create this standard in AWS Control Tower.

- The Security Hub CSPM controls are active at the OU level only, not for all AWS Control Tower OUs (if not enabled for all), and not for individual accounts.

Find Security Hub CSPM Controls in AWS Control Tower

To see what Security Hub CSPM controls are supported by AWS Control Tower, you can use one of the following methods:

- AWS Control Tower console where you can filter for `"Control owner = AWS Security Hub"`
- AWS Control Catalog API (call the ListControls API) with a filter for `Implementations.Types` set to `AWS::SecurityHub::SecurityControl`
- AWS CLI (run the list-controls command) with a filter for `Implementations.Types` set to `AWS::SecurityHub::SecurityControl`. Example CLI command:

```
aws controlcatalog list-controls --filter '{"Implementations":{"Types":
["AWS::SecurityHub::SecurityControl"]}}'
```

To identify a Security Hub CSPM control by control ID in AWS Control Tower, you can use the field `Implementation.Identifier`. This field maps to Security Hub CSPM control ID and can be used to filter for a specific control ID. To retrieve control metadata for a specific Security Hub CSPM control (say, "CodeBuild.1") in AWS Control Tower, you can use the ListControls API:

```
aws controlcatalog list-controls --filter '{"Implementations":{"Identifiers":
["CodeBuild.1"],"Types":["AWS::SecurityHub::SecurityControl"]}}'
```

Enable or remove controls for the Service-Managed Standard

To avoid drift, always enable and remove controls for the Service-Managed Standard by means of the AWS Control Tower service, either in the console or by calling the AWS Control Tower APIs, `EnableControl` and `DisableControl`. When you change the enablement status of a control in AWS Control Tower, the change also is reflected in Security Hub CSPM.

If you deactivate a Service-Managed Standard control by means of the Security Hub CSPM console, the AWS Control Tower member account enters a state of control drift. In this situation, AWS Control Tower is not receiving the Security Hub CSPM findings for the control that you deactivated.

You must resolve this drift before AWS Control Tower can apply the control successfully to your registered organizational units and member accounts.

You can delete this standard in the AWS Control Tower console by deactivating all controls in the standard. This deletes the standard for all managed accounts and governed Regions in AWS Control Tower. Deleting the standard does not deactivate Security Hub CSPM for your account.

Deprecated controls

The control named **[SH.S3.4] S3 buckets should have server-side encryption enabled** is deprecated, effective July 18, 2023. It was removed from the controls library on August 18, 2023. For more information, see AWS Control Tower deprecates two controls.

The control named **[SH.RDS.18] RDS instances should be deployed in a VPC** is deprecated, effective April 28, 2025, and is to be removed from the Control Catalog.

Security Hub CSPM score and findings

Based on control status, Security Hub CSPM calculates a security score for the **Service-Managed Standard: AWS Control Tower**. This security score and the control findings are available only in Security Hub CSPM. These items aren't available in AWS Control Tower.

> **Note**
>
> When you create **Service-Managed Standard: AWS Control Tower** and enable controls for it, Security Hub CSPM may take up to 18 hours to generate findings for controls that use the same underlying AWS Config service-linked rule as controls from other enabled Security Hub CSPM standards. For more information, see Schedule for running security checks in the AWS Security Hub CSPM User Guide.

Security Hub CSPM control drift reporting

When reporting drift for controls that are part of the AWS Security Hub CSPM Service-Managed Standard, AWS Control Tower receives a daily status update from Security Hub CSPM. If no update is received, AWS Control Tower verifies whether drift has occurred. If so, AWS Control Tower reports drift. If a control shows drift, AWS Control Tower sends an Amazon SNS notification to the AWS Control Tower `security-aggregate-notification` channel. You must subscribe to this SNS notification to receive information about which specific account is affected by Security Hub CSPM

control drift. For more information about Security Hub CSPM control drift in AWS Control Tower, see Security Hub control drift.

Although controls are active in every governed Region, AWS Control Tower sends the AWS Security Hub CSPM **Finding** events to the AWS Control Tower home Region only.

Remediate drift

When drift is reported, you can remediate the situation by choosing **Re-register OU** on the **Organizations** page in the AWS Control Tower console, or by deactivating and re-activating the control that's in a drifted state, either by means of the console, or through the AWS Control Tower API.

Manage controls

You can enable and manage some Security Hub CSPM controls from AWS Control Tower, with the Security Hub CSPM Service-managed Standard: AWS Control Tower.

Unsupported Regions

It is important to know that some Security Hub CSPM controls do not operate in certain AWS Regions where AWS Control Tower is available, because those Regions do not support the required underlying functionality. As a result, when you deploy an Security Hub CSPM control through AWS Control Tower, the control may not be operating in all Regions that you govern with AWS Control Tower. For more information about the Security Hub CSPM controls that cannot be deployed in certain Regions, see the Security Hub CSPM controls reference documentation.

You can view the most updated list of the Regions for each control in the AWS Control Tower console, or by calling the `GetControl` API.

Integrated AWS Config controls available in AWS Control Tower

AWS Control Tower is integrated with AWS Config to provide over 500 selected additional detective controls that help you monitor and manage your AWS environment. These AWS Config controls are available in the AWS Control Tower console and the Control Catalog APIs. The **Control owner** or **Implementation** field for these controls is displayed as AWS Config or `AWS::Config::ConfigRule`.

You can use AWS Control Tower to search and discover the AWS Config rules that you need to govern your multi-account environment; and you can enable and manage these controls directly

from the AWS Control Tower console. To search from the console, go to the Control Catalog and search for controls with the **Implementation** filter AWS Config. (Example: `Implementation = AWS Config`)

The AWS Control Tower console and AWS Config console each display the same metadata for these controls.

You can enable and disable the AWS Config controls through the AWS Control Tower console or the `EnableControl` and `DisableControl` APIs. Control details are viewable programmatically by calling the Control Catalog `GetControl` and `ListControls` APIs.

Differences

- In AWS Config, these integrated controls are listed by identifier.
- In the AWS Control Tower console and APIs, the integrated controls are shown with names that summarize their function.

> **Note**
>
> AWS Control Tower documentation does not provide a comprehensive list of integrated AWS Config controls. For more information about these controls, see List of AWS Config managed rules in the *AWS Config Developer Guide*, or view them in the AWS Control Tower console.

> **Important**
>
> AWS Control Tower doesn't support parameter configuration for detective controls. If a control relies on optional parameters, it deploys without them. This can result in more restrictive evaluation behavior. For example, the `CONFIG.EC2.DT.17` control evaluates all internet gateway attachments as `NON_COMPLIANT` when you deploy it without the `AuthorizedVpcIds` parameter. To deploy these types of controls with parameters, create the corresponding AWS Config rule directly in AWS Config.

Change in drift behavior with service-linked AWS Config rules

Before the introduction of service-linked Config rules in AWS Control Tower, you could modify AWS Config rule configurations or add remediations outside of AWS Control Tower. With the release of service-linked Config rules, this behavior has changed:

- Modifications made to Config rule settings outside of AWS Control Tower are treated as drift.
- External remediation configurations added to these Config rules are treated as drift.
- AWS Control Tower automatically removes these external modifications with the adoption of service-linked Config rules.
- To maintain consistent governance, all updates that AWS Control Tower supports for your service-linked Config rules must be managed through AWS Control Tower.

> **Important**
>
> Before you adopt service-linked Config rules, review the existing customizations, such as remediations, that you have made to Config rules outside of AWS Control Tower, because these customizations will be removed during the transition. The AWS Config APIs do not support adding remediation configurations for service-linked AWS Config rules. See `PutRemediationConfigurations`.

Controls with parameters

In AWS Control Tower, RCP-based and certain SCP-based controls support configuration. These controls contain elements that are included by AWS Control Tower conditionally, based on the configuration you select.

For example, some control policies include inline templating variables, such as the one shown in the example that follows. The example shows the **ExemptedPrincipalArns** parameter.

```
{
            "Sid": "CTEC2PV1",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:CreateSnapshots"
```

```
            ],
            "Resource": "arn:*:ec2:*:*:volume/*",
            "Condition": {
                "Bool": {
                    "ec2:Encrypted": "false"
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
```

A control may support any of the following five configuration parameters:

- **ExemptedPrincipalArns**: A list of AWS IAM principal ARNs that are exempted from this control.
  - This parameter allows you to exempt IAM Principals from this control by way of an **ArnNotLikeIfExists** condition key operator and **aws:PrincipalArn** condition key that is applied to the control policy by AWS Control Tower when you enable the control. The **ExemptedPrincipalArns** parameter allows you to use the wildcard character (*) in the IAM principal ARNs that you specify. You can use the wildcard character to exempt all IAM principals in an AWS account, or exempt a common principal across multiple AWS accounts.
  - When you use the wildcard character to exempt principals, be sure that you follow the principal of least privilege: include only those IAM principal ARNs that you require to be exempt from a control. Otherwise, if your exemptions are too broad, the control may not come into effect when you intend it to.
- **AllowedRegions**: List of AWS Regions exempted from the control.
- **ExemptedActions**: List of AWS IAM actions exempted from the control.
- **ExemptedResourceArns**: List of resource ARNs exempted from the control.
- **ExemptAssumeRoot**: A parameter that lets you choose whether to exempt requests made with `AssumeRoot` from this control, for this OU.
  - For member accounts, the `AssumeRoot` property is included in requests initiated by IAM centralized root access. This parameter applies only to the `AWS-GR_RESTRICT_ROOT_USER` control.
  - If you add the parameter when enabling the control, the `AssumeRoot` exemption is allowed. If you omit the parameter, the `AssumeRoot` exemption is not permitted. The parameter does not accept `False` as a value.

For more details about configuring controls with parameters, see `ControlParameter` in the *AWS Control Tower API Reference*.

List of parameterized controls

Control identifier	Display name
AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED	Disallow modification of an Amazon S3 bucket default encryption
AWS-GR_AUDIT_BUCKET_LOGGING_ENABLED	Disallow modification of server access logging for an Amazon S3 bucket
AWS-GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED	Disallow policy changes to an Amazon S3 bucket
AWS-GR_AUDIT_BUCKET_RETENTION_POLICY	Set a retention policy for log archive
AWS-GR_DISALLOW_CROSS_REGION_NETWORKING	Disallow cross-region networking for Amazon EC2, Amazon CloudFront, and AWS Global Accelerator
AWS-GR_DISALLOW_VPC_INTERNET_ACCESS	Disallow internet access for an Amazon VPC instance managed by a customer
AWS-GR_DISALLOW_VPN_CONNECTIONS	Disallow AWS Virtual Private Network (VPN) connections
AWS-GR_RESTRICT_ROOT_USER	Disallow actions as a root user
AWS-GR_RESTRICT_ROOT_USER_ACCESS_KEYS	Disallow creation of access keys for the root user
AWS-GR_RESTRICT_S3_CROSS_REGION_REPLICATION	Disallow cross region replication for S3 buckets
AWS-GR_RESTRICT_S3_DELETE_WITHOUT_MFA	Disallow delete actions on S3 buckets without MFA

Control identifier	Display name
CT.APPSYNC.PV.1	Require an AWS AppSync GraphQL API to be configured with private visibility
CT.EC2.PV.1	Require an Amazon EBS snapshot to be created from an encrypted EC2 volume
CT.EC2.PV.2	Require that an attached Amazon EBS volume is configured to encrypt data at rest
CT.EC2.PV.3	Require that an Amazon EBS snapshot cannot be publicly restorable
CT.EC2.PV.4	Require that Amazon EBS direct APIs are not called
CT.EC2.PV.5	Disallow the use of Amazon EC2 VM import and export
CT.EC2.PV.6	Disallow the use of deprecated Amazon EC2 RequestSpotFleet and RequestSpotInstances API actions
CT.KMS.PV.1	Require an AWS KMS key policy to have a statement that limits creation of AWS KMS grants to AWS services
CT.KMS.PV.2	Require that an AWS KMS asymmetric key with RSA key material used for encryption does not have a key length of 2048 bits
CT.KMS.PV.3	Require that an AWS KMS key is configured with the bypass policy lockout safety check enabled
CT.KMS.PV.4	Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from AWS CloudHSM

Control identifier	Display name
CT.KMS.PV.5	Require that an AWS KMS customer-managed key (CMK) is configured with imported key material
CT.KMS.PV.6	Require that an AWS KMS customer-managed key (CMK) is configured with key material originating from an external key store (XKS)
CT.LAMBDA.PV.1	Require an AWS Lambda function URL to use AWS IAM-based authentication
CT.LAMBDA.PV.2	Require an AWS Lambda function or AWS Lambda function URL to be configured for access only to principals within your AWS account
CT.KMS.PV.7	Require that the organization's AWS Key Management Service resources are accessibl e only by IAM principals that belong to the organization, or by an AWS service
CT.S3.PV.2	Require all requests to Amazon S3 resources use authentication based on an Authorization header
CT.S3.PV.3	Require requests to Amazon S3 resources to use a minimum TLS version of 1.3
CT.S3.PV.4	Require that the organization's Amazon S3 resources are accessible only by IAM principals that belong to the organization or by an AWS service
CT.S3.PV.5	Require encryption of data in transit for calls to Amazon S3 resources

Control identifier	Display name
CT.S3.PV.6	Require all object uploads to Amazon S3 buckets to use server-side encryption with an AWS KMS key (SSE-KMS
CT.SECRETSMANAGER.PV.1	Require that the organization's AWS Secrets Manager resources are accessible only by IAM principals that belong to the organization or by an AWS service
CT.SQS.PV.1	Require that the organization's Amazon SQS resources are accessible only by IAM principals that belong to the organization, or by an AWS service
CT.STS.PV.1	Require that the organization's AWS Security Token Service resources are accessible only by IAM principals that belong to the organization, or by an AWS service

Optional controls

Optional controls in AWS Control Tower are applied at the OU level. You can activate and deactivate these optional controls through the AWS Control Tower console, or by means of the control APIs.

AWS Control Tower offers several types of optional controls:

- Proactive controls, which are based on CloudFormation hooks.
- Controls implemented with resource control policies (RCPs), which are based on RCPs from AWS Organizations. For more information, see Resource control policies in the AWS Organizations documentation.
- Controls implemented with declarative policies, which are based on *declarative policies* from AWS Organizations. For more information, see Declarative policies in the AWS Organizations documentation.

- Security Hub CSPM controls, which are based on AWS Config rules – these controls are owned by Security Hub CSPM and integrated with AWS Control Tower, by means of the **Service-Managed Standard: AWS Control Tower**.
- Digital sovereignty controls, which are elective controls based on SCPs and AWS Config rules, implemented within AWS Control Tower. This group includes the data residency controls.
- Strongly recommended controls, which are based on SCPs and AWS Config rules, implemented within AWS Control Tower.
- Elective controls, which are based on SCPs and AWS Config rules, implemented within AWS Control Tower.

The strongly recommended and elective controls owned by AWS Control Tower are optional, which means that you can customize the level of enforcement for OUs in your landing zone by choosing which ones to enable. Optional controls are not enabled by default. For more information about optional controls, see the following control reference pages in the next sections.

> **Note**
>
> It is important to know that some detective controls in AWS Control Tower do not operate in certain AWS Regions where AWS Control Tower is available, because those Regions do not support the required underlying functionality. As a result, when you deploy a detective control, the control may not be operating in all Regions that you govern with AWS Control Tower. For details, see Control limitations and Security Hub controls.
> You can view the Regions for each control in the AWS Control Tower console, or by calling the `GetControl` API that is part of the Control Catalog namespace.
> For more information about the detective controls that cannot be deployed in certain Regions, see the Regional services list documentation to learn more about the Regions where AWS Config is available. If the detective control is implemented as a managed AWS Config rule, see the Security Hub CSPM controls reference documentation.

Strongly recommended controls

Strongly recommended controls are owned by AWS Control Tower. They are based on best practices for well-architected multi-account environments. These controls are not enabled by default, and they can be deactivated through the AWS Control Tower console or the control APIs.

Following, you'll find a reference for each of the strongly recommended controls available in AWS Control Tower.

Topics

- Strongly recommended controls with preventive behavior
- Strongly recommended controls with detective behavior

Strongly recommended controls with preventive behavior

The following strongly recommended controls have preventive behavior.

Topics

- [AWS-GR_RESTRICT_ROOT_USER_ACCESS_KEYS] Disallow creation of access keys for the root user
- [AWS-GR_RESTRICT_ROOT_USER] Disallow actions as a root user

[AWS-GR_RESTRICT_ROOT_USER_ACCESS_KEYS] Disallow creation of access keys for the root user

Secure your AWS accounts by disallowing creation of access keys for the root user, which will allow unrestricted access to all resources in the account. We recommend that you instead create access keys for an AWS Identity and Access Management (IAM) user for everyday interaction with your AWS account.

This is a preventive control with strongly-recommended guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Identity and Access Management (IAM)

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`, `AWS::IAM::AccessKey`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRRESTRICTROOTUSERACCESSKEYS",
            "Effect": "Deny",
            "Action": "iam:CreateAccessKey",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "aws:PrincipalArn": [
                        "arn:*:iam::*:root"
                    ]
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[AWS-GR_RESTRICT_ROOT_USER] Disallow actions as a root user

Secure your AWS accounts by disallowing account access with root user credentials, which are credentials of the account owner and allow unrestricted access to all resources in the account. We recommend that you instead create AWS Identity and Access Management (IAM) users for everyday interaction with your AWS account.

This is a preventive control with strongly-recommended guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: AWS Identity and Access Management (IAM)

Control metadata

- **Control objective:** Enforce least privilege
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::::Account`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**, **ExemptAssumeRoot**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRRESTRICTROOTUSER",
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "aws:PrincipalArn": [
                        "arn:*:iam::*:root"
                    ]
```

```
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}{% if ExemptAssumeRoot %},
                "Null": {
                    "aws:AssumedRoot": "true"
                }{% endif %}
            }
        }
    ]
}
```

Strongly recommended controls with detective behavior

The following strongly recommended controls have detective behavior.

Topics

- Detect Whether Encryption is Enabled for Amazon EBS Volumes Attached to Amazon EC2 Instances
- Detect Whether Unrestricted Incoming TCP Traffic is Allowed
- Detect Whether Unrestricted Internet Connection Through SSH is Allowed
- Detect Whether MFA for the Root User is Enabled
- Detect Whether Public Read Access to Amazon S3 Buckets is Allowed
- Detect Whether Public Write Access to Amazon S3 Buckets is Allowed
- Detect Whether Amazon EBS Volumes are Attached to Amazon EC2 Instances
- Detect Whether Amazon EBS Optimization is Enabled for Amazon EC2 Instances
- Detect Whether Public Access to Amazon RDS Database Instances is Enabled
- Detect Whether Public Access to Amazon RDS Database Snapshots is Enabled
- Detect Whether Storage Encryption is Enabled for Amazon RDS Database Instances
- Detect whether an account has AWS CloudTrail or CloudTrail Lake enabled

Detect Whether Encryption is Enabled for Amazon EBS Volumes Attached to Amazon EC2 Instances

This control detects whether the Amazon EBS volumes attached to an Amazon EC2 instance are encrypted. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control isn't enabled on any OUs.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check for encryption of all storage volumes
 attached to compute
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForEncryptedVolumes:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether EBS volumes that are in an attached state are
 encrypted.
      Source:
        Owner: AWS
        SourceIdentifier: ENCRYPTED_VOLUMES
      Scope:
        ComplianceResourceTypes:
          - AWS::EC2::Volume
```

Detect Whether Unrestricted Incoming TCP Traffic is Allowed

This control helps reduce a server's exposure to risk by detecting whether unrestricted incoming TCP traffic is allowed. It detects whether internet connections are enabled to Amazon EC2 instances through services such as Remote Desktop Protocol (RDP). This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

Note

This control fails if any of the rules in a security group allow ingress traffic from `0.0.0.0/0` or `::/0` for those ports.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether security groups that are in
 use disallow unrestricted incoming TCP traffic to the specified ports.
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  blockedPort1:
    Type: String
    Default: '20'
    Description: Blocked TCP port number.
  blockedPort2:
    Type: String
    Default: '21'
    Description: Blocked TCP port number.
  blockedPort3:
    Type: String
    Default: '3389'
    Description: Blocked TCP port number.
  blockedPort4:
    Type: String
    Default: '3306'
    Description: Blocked TCP port number.
  blockedPort5:
    Type: String
    Default: '4333'
    Description: Blocked TCP port number.
Conditions:
  blockedPort1:
    Fn::Not:
    - Fn::Equals:
      - ''
      - Ref: blockedPort1
  blockedPort2:
    Fn::Not:
```

```
    - Fn::Equals:
      - ''
      - Ref: blockedPort2
  blockedPort3:
    Fn::Not:
    - Fn::Equals:
      - ''
      - Ref: blockedPort3
  blockedPort4:
    Fn::Not:
    - Fn::Equals:
      - ''
      - Ref: blockedPort4
  blockedPort5:
    Fn::Not:
    - Fn::Equals:
      - ''
      - Ref: blockedPort5
Resources:
  CheckForRestrictedCommonPortsPolicy:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether security groups that are in use disallow unrestricted
 incoming TCP traffic to the specified ports.
      InputParameters:
        blockedPort1:
          Fn::If:
          - blockedPort1
          - Ref: blockedPort1
          - Ref: AWS::NoValue
        blockedPort2:
          Fn::If:
          - blockedPort2
          - Ref: blockedPort2
          - Ref: AWS::NoValue
        blockedPort3:
          Fn::If:
          - blockedPort3
          - Ref: blockedPort3
          - Ref: AWS::NoValue
        blockedPort4:
          Fn::If:
          - blockedPort4
```

```
          - Ref: blockedPort4
          - Ref: AWS::NoValue
        blockedPort5:
          Fn::If:
          - blockedPort5
          - Ref: blockedPort5
          - Ref: AWS::NoValue
      Scope:
        ComplianceResourceTypes:
        - AWS::EC2::SecurityGroup
      Source:
        Owner: AWS
        SourceIdentifier: RESTRICTED_INCOMING_TRAFFIC
```

Detect Whether Unrestricted Internet Connection Through SSH is Allowed

This control detects whether internet connections are allowed through remote services such as the Secure Shell (SSH) protocol. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

> **Note**
>
> This control fails if any of the rules in a security group allow ingress traffic from `0.0.0.0/0` or `::/0` for SSH traffic.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether security groups that are in
 use disallow SSH
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForRestrictedSshPolicy:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether security groups that are in use disallow unrestricted
 incoming SSH traffic.
```

```
      Scope:
        ComplianceResourceTypes:
        - AWS::EC2::SecurityGroup
      Source:
        Owner: AWS
        SourceIdentifier: INCOMING_SSH_DISABLED
```

Detect Whether MFA for the Root User is Enabled

This control detects whether multi-factor authentication (MFA) is enabled for the root user of the management account. MFA reduces vulnerability risks from weak authentication by requiring an additional authentication code after the user name and password are successful. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to require MFA for root access to accounts
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  MaximumExecutionFrequency:
    Type: String
    Default: 24hours
    Description: The frequency that you want AWS Config to run evaluations for the
 rule.
    AllowedValues:
    - 1hour
    - 3hours
    - 6hours
    - 12hours
    - 24hours
Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours
Resources:
```

```
  CheckForRootMfa:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether the root user of your AWS account requires multi-
factor authentication for console sign-in.
      Source:
        Owner: AWS
        SourceIdentifier: ROOT_ACCOUNT_MFA_ENABLED
      MaximumExecutionFrequency:
        !FindInMap
          - Settings
          - FrequencyMap
          - !Ref MaximumExecutionFrequency
```

Detect Whether Public Read Access to Amazon S3 Buckets is Allowed

This control detects whether public read access is allowed to Amazon S3 buckets. It helps you maintain secure access to data stored in the buckets. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check that your S3 buckets do not allow
 public access
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForS3PublicRead:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks that your S3 buckets do not allow public read access. If an
 S3 bucket policy or bucket ACL allows public read access, the bucket is noncompliant.
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_PUBLIC_READ_PROHIBITED
      Scope:
        ComplianceResourceTypes:
```

```
          - AWS::S3::Bucket
```

Detect Whether Public Write Access to Amazon S3 Buckets is Allowed

This control detects whether public write access is allowed to Amazon S3 buckets. It helps you maintain secure access to data stored in the buckets. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check that your S3 buckets do not allow
 public access
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForS3PublicWrite:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks that your S3 buckets do not allow public write access. If an
 S3 bucket policy or bucket ACL allows public write access, the bucket is noncompliant.
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_PUBLIC_WRITE_PROHIBITED
      Scope:
        ComplianceResourceTypes:
          - AWS::S3::Bucket
```

Detect Whether Amazon EBS Volumes are Attached to Amazon EC2 Instances

This control detects whether an Amazon EBS volume device persists independently from an Amazon EC2 instance. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Configure AWS Config rules to check whether EBS volumes are attached to
 EC2 instances
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  deleteOnTermination:
    Type: 'String'
    Default: 'None'
    Description: 'Check for Delete on termination'
Conditions:
  deleteOnTermination:
    Fn::Not:
    - Fn::Equals:
      - 'None'
      - Ref: deleteOnTermination
Resources:
  CheckForEc2VolumesInUse:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether EBS volumes are attached to EC2 instances
      InputParameters:
        deleteOnTermination:
          Fn::If:
            - deleteOnTermination
            - Ref: deleteOnTermination
            - Ref: AWS::NoValue
      Source:
        Owner: AWS
        SourceIdentifier: EC2_VOLUME_INUSE_CHECK
      Scope:
        ComplianceResourceTypes:
          - AWS::EC2::Volume
```

Detect Whether Amazon EBS Optimization is Enabled for Amazon EC2 Instances

Detects whether Amazon EC2 instances are launched without an Amazon EBS volume that is optimized for performance. Amazon EBS-optimized volumes minimize contention between Amazon EBS I/O and other traffic from your instance. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether EBS optimization is enabled
 for your EC2 instances that can be EBS-optimized
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForEbsOptimizedInstance:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether EBS optimization is enabled for your EC2 instances
 that can be EBS-optimized
      Source:
        Owner: AWS
        SourceIdentifier: EBS_OPTIMIZED_INSTANCE
      Scope:
        ComplianceResourceTypes:
          - AWS::EC2::Instance
```

Detect Whether Public Access to Amazon RDS Database Instances is Enabled

Detects whether your Amazon RDS database instances allow public access. You can secure your Amazon RDS database instances by disallowing public access. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether Amazon RDS instances are not
 publicly accessible.
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForRdsPublicAccess:
    Type: AWS::Config::ConfigRule
```

```
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether the Amazon Relational Database Service
 (RDS) instances are not publicly accessible. The rule is non-compliant if the
 publiclyAccessible field is true in the instance configuration item.
      Source:
        Owner: AWS
        SourceIdentifier: RDS_INSTANCE_PUBLIC_ACCESS_CHECK
      Scope:
        ComplianceResourceTypes:
          - AWS::RDS::DBInstance
```

Detect Whether Public Access to Amazon RDS Database Snapshots is Enabled

Detects whether your Amazon RDS database snapshots have public access enabled. You can protect your information by disabling public access. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Checks if Amazon Relational Database Service (Amazon RDS) snapshots are
 public.
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForRdsStorageEncryption:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks if Amazon Relational Database Service (Amazon RDS) snapshots
 are public. The rule is non-compliant if any existing and new Amazon RDS snapshots are
 public.
      Source:
        Owner: AWS
        SourceIdentifier: RDS_SNAPSHOTS_PUBLIC_PROHIBITED
      Scope:
        ComplianceResourceTypes:
          - AWS::RDS::DBSnapshot
```

Detect Whether Storage Encryption is Enabled for Amazon RDS Database Instances

Detects Amazon RDS database instances that are not encrypted at rest. You can secure your Amazon RDS database instances at rest by encrypting the underlying storage for database instances and their automated backups, Read Replicas, and snapshots. This control does not change the status of the account. This is a detective control with strongly recommended guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether storage encryption is enabled
 for your RDS DB instances
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForRdsStorageEncryption:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether storage encryption is enabled for your RDS DB
 instances.
      Source:
        Owner: AWS
        SourceIdentifier: RDS_STORAGE_ENCRYPTED
      Scope:
        ComplianceResourceTypes:
          - AWS::RDS::DBInstance
```

Detect whether an account has AWS CloudTrail or CloudTrail Lake enabled

This control detects whether an account has AWS CloudTrail or CloudTrail Lake enabled. The rule is NON_COMPLIANT if either CloudTrail or CloudTrail Lake is not enabled in an account. This is a detective control with strongly recommended guidance. By default, this control is not enabled on any OUs.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Configure AWS Config rules to detect whether an account has AWS CloudTrail
 or CloudTrail Lake enabled.

Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'

Resources:
  CheckForCloudtrailEnabled:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Detects whether an account has AWS CloudTrail or CloudTrail Lake
 enabled. The rule is NON_COMPLIANT if either CloudTrail or CloudTrail Lake is not
 enabled in an account.
      Source:
        Owner: AWS
        SourceIdentifier: CLOUD_TRAIL_ENABLED
```

Elective controls

Elective controls enable you to lock down or track attempts at performing commonly restricted actions in an AWS enterprise environment. These controls are not enabled by default, and can be disabled. Following, you'll find a reference for the elective controls available in AWS Control Tower. The elective controls specifically for data residency are collected into a separate section, Controls that enhance data residency protection.

The elective controls include the data residency controls, which are part of the **Digital sovereignty** group. For more information, see Digital sovereignty controls.

Topics

- Elective controls with detective behavior
- Elective controls with preventive behavior

Elective controls with detective behavior

The following elective controls have detective behavior.

Topics

- Detect Whether MFA is Enabled for AWS IAM Users
- Detect Whether MFA is Enabled for AWS IAM Users of the AWS Console
- Detect Whether Versioning for Amazon S3 Buckets is Enabled

Detect Whether MFA is Enabled for AWS IAM Users

This control detects whether MFA is enabled for AWS IAM users. You can protect your account by requiring MFA for all AWS users in the account. MFA requires an additional authentication code after the user name and password are successful. This control does not change the status of the account. This is a detective control with elective guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether the IAM users have MFA enabled
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  MaximumExecutionFrequency:
    Type: String
    Default: 1hour
    Description: The frequency that you want AWS Config to run evaluations for the
 rule.
    AllowedValues:
    - 1hour
    - 3hours
    - 6hours
    - 12hours
    - 24hours
Mappings:
  Settings:
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours
Resources:
  CheckForIAMUserMFA:
    Type: AWS::Config::ConfigRule
```

```
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether the AWS Identity and Access Management users have
 multi-factor authentication (MFA) enabled. The rule is COMPLIANT if MFA is enabled.
      Source:
        Owner: AWS
        SourceIdentifier: IAM_USER_MFA_ENABLED
      MaximumExecutionFrequency:
        !FindInMap
          - Settings
          - FrequencyMap
          - !Ref MaximumExecutionFrequency
```

Detect Whether MFA is Enabled for AWS IAM Users of the AWS Console

Protects your account by requiring MFA for all AWS IAM users in the console. MFA reduces vulnerability risks from weak authentication by requiring an additional authentication code after the user name and password are successful. This control detects whether MFA is enabled. This control does not change the status of the account. This is a detective control with elective guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether MFA is enabled for all AWS IAM
 users that use a console password.
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
  MaximumExecutionFrequency:
    Type: String
    Default: 1hour
    Description: The frequency that you want AWS Config to run evaluations for the
 rule.
    AllowedValues:
    - 1hour
    - 3hours
    - 6hours
    - 12hours
    - 24hours
Mappings:
  Settings:
```

```
    FrequencyMap:
      1hour   : One_Hour
      3hours  : Three_Hours
      6hours  : Six_Hours
      12hours : Twelve_Hours
      24hours : TwentyFour_Hours
Resources:
  CheckForIAMUserConsoleMFA:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether AWS Multi-Factor Authentication (MFA) is enabled for
 all AWS Identity and Access Management (IAM) users that use a console password. The
 rule is COMPLIANT if MFA is enabled.
      Source:
        Owner: AWS
        SourceIdentifier: MFA_ENABLED_FOR_IAM_CONSOLE_ACCESS
      MaximumExecutionFrequency:
        !FindInMap
          - Settings
          - FrequencyMap
          - !Ref MaximumExecutionFrequency
```

Detect Whether Versioning for Amazon S3 Buckets is Enabled

Detects whether your Amazon S3 buckets are enabled for versioning. Versioning allows you to recover objects from accidental deletion or overwrite. This control does not change the status of the account. This is a detective control with elective guidance. By default, this control is not enabled.

The artifact for this control is the following AWS Config rule.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Configure AWS Config rules to check whether versioning is enabled for your
 S3 buckets.
Parameters:
  ConfigRuleName:
    Type: 'String'
    Description: 'Name for the Config rule'
Resources:
  CheckForS3VersioningEnabled:
    Type: AWS::Config::ConfigRule
    Properties:
```

```
      ConfigRuleName: !Sub ${ConfigRuleName}
      Description: Checks whether versioning is enabled for your S3 buckets.
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_VERSIONING_ENABLED
      Scope:
        ComplianceResourceTypes:
          - AWS::S3::Bucket
```

Elective controls with preventive behavior

The following elective controls have preventive behavior.

The elective controls with preventive behavior are configurable. For more information about configurable controls, see Controls with parameters.

Topics

- [AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED] Disallow modification of Amazon S3 bucket encryption
- [AWS-GR_AUDIT_BUCKET_LOGGING_ENABLED] Disallow modification of server access logging for an Amazon S3 bucket
- [AWS-GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED] Disallow policy changes to an Amazon S3 bucket
- [AWS-GR_AUDIT_BUCKET_RETENTION_POLICY] Set a retention policy for log archive
- [AWS-GR_DISALLOW_CROSS_REGION_NETWORKING] Disallow cross-region networking for Amazon EC2, Amazon CloudFront, and AWS Global Accelerator
- [AWS-GR_DISALLOW_VPC_INTERNET_ACCESS] Disallow internet access for an Amazon VPC instance managed by a customer
- [AWS-GR_DISALLOW_VPN_CONNECTIONS] Disallow Amazon Virtual Private Network (VPN) connections
- [AWS-GR_RESTRICT_S3_CROSS_REGION_REPLICATION] Disallow cross region replication for Amazon S3 buckets
- [AWS-GR_RESTRICT_S3_DELETE_WITHOUT_MFA] Disallow delete actions on S3 buckets without MFA
- [CT.CLOUDFORMATION.PR.1] Disallow management of resource types, modules, and hooks within the CloudFormation registry

[AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED] Disallow modification of Amazon S3 bucket encryption

This control disallows modification of Amazon S3 bucket encryption configuration.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Encrypt data at rest
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRAUDITBUCKETENCRYPTIONENABLED",
            "Effect": "Deny",
```

```
            "Action": "s3:PutEncryptionConfiguration",
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

[AWS-GR_AUDIT_BUCKET_LOGGING_ENABLED] Disallow modification of server access logging for an Amazon S3 bucket

This control disallows modification of server access logging for an Amazon S3 bucket.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Establish logging and monitoring
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports

the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRAUDITBUCKETLOGGINGENABLED",
            "Effect": "Deny",
            "Action": "s3:PutBucketLogging",
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

[AWS-GR_AUDIT_BUCKET_POLICY_CHANGES_PROHIBITED] Disallow policy changes to an Amazon S3 bucket

This control disallows modification of an Amazon S3 bucket policy.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Protect data integrity
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRAUDITBUCKETPOLICYCHANGESPROHIBITED",
            "Effect": "Deny",
            "Action": "s3:PutBucketPolicy",
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

[AWS-GR_AUDIT_BUCKET_RETENTION_POLICY] Set a retention policy for log archive

Limit data retention in the log archive using a retention policy that defaults to 365 days.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Improve resiliency
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRAUDITBUCKETRETENTIONPOLICY",
```

```
            "Effect": "Deny",
            "Action": "s3:PutLifecycleConfiguration",
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

[AWS-GR_DISALLOW_CROSS_REGION_NETWORKING] Disallow cross-region networking for Amazon EC2, Amazon CloudFront, and AWS Global Accelerator

Disallow cross-region networking connections from Amazon EC2, Amazon CloudFront, and AWS Global Accelerator services.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon CloudFront, Amazon EC2, AWS Global Accelerator

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::CloudFront::Distribution`, `AWS::EC2::VPCPeeringConnection`, `AWS::EC2::TransitGatewayPeeringAttachment`, `AWS::GlobalAccelerator::Accelerator`, `AWS::GlobalAccelerator::EndpointGroup`, `AWS::GlobalAccelerator::Listener`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRDISALLOWCROSSREGIONNETWORKING",
            "Effect": "Deny",
            "Action": [
                "cloudfront:CreateDistribution",
                "cloudfront:UpdateDistribution",
                "ec2:AcceptTransitGatewayPeeringAttachment",
                "ec2:AcceptVpcPeeringConnection",
                "ec2:CreateTransitGatewayPeeringAttachment",
                "ec2:CreateVpcPeeringConnection",
                "globalaccelerator:Create*",
                "globalaccelerator:Update*"
            ],
            "Resource": "*"{% if ExemptedPrincipalArns %},
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[AWS-GR_DISALLOW_VPC_INTERNET_ACCESS] Disallow internet access for an Amazon VPC instance managed by a customer

Disallow internet access for an Amazon Virtual Private Cloud (VPC) instance managed by a customer, rather than by an AWS service.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Limit network access
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Control groups:** digital-sovereignty
- **Resource types:** `AWS::EC2::InternetGateway`, `AWS::EC2::EgressOnlyInternetGateway`, `AWS::EC2::VPC`, `AWS::EC2::Subnet`, `AWS::EC2::CarrierGateway`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "GRDISALLOWVPCINTERNETACCESS",
            "Effect": "Deny",
            "Action": [
                "ec2:AttachEgressOnlyInternetGateway",
                "ec2:AttachInternetGateway",
                "ec2:CreateCarrierGateway",
                "ec2:CreateDefaultSubnet",
                "ec2:CreateDefaultVpc",
                "ec2:CreateEgressOnlyInternetGateway",
                "ec2:CreateInternetGateway"
            ],
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        {{ExemptedPrincipalArns}}
                        "arn:*:iam::*:role/AWSControlTowerExecution"
                    ]
                }
            }
        }
    ]
}
```

[AWS-GR_DISALLOW_VPN_CONNECTIONS] Disallow Amazon Virtual Private Network (VPN) connections

Disallows Virtual Private Network (VPN) connections (Site-to-Site VPN and Client VPN) to an Amazon Virtual Private Cloud (VPC).

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon EC2

Control metadata

- **Control objective:** Limit network access

- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::EC2::VPNGateway`, `AWS::EC2::CustomerGateway`, `AWS::EC2::VPNConnection`, `AWS::EC2::ClientVpnEndpoint`, `AWS::EC2::ClientVpnTargetNetworkAssociation`, `AWS::EC2::ClientVpnAuthorizationRule`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRDISALLOWVPNCONNECTIONS",
            "Effect": "Deny",
            "Action": [
                "ec2:AssociateClientVpnTargetNetwork",
                "ec2:AttachVPNGateway",
                "ec2:AuthorizeClientVpnIngress",
                "ec2:CreateClientVpnEndpoint",
                "ec2:CreateCustomerGateway",
                "ec2:CreateVPNGateway",
                "ec2:CreateVpnConnection",
                "ec2:ModifyClientVpnEndpoint",
                "ec2:ModifyVpnConnection"
            ],
            "Resource": "*"{% if ExemptedPrincipalArns %},
            "Condition": {
                "ArnNotLike": {
```

```
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[AWS-GR_RESTRICT_S3_CROSS_REGION_REPLICATION] Disallow cross region replication for Amazon S3 buckets

Contain the location of your Amazon S3 data to a single region by disabling any automatic, asynchronous copying of objects across buckets to other AWS Regions.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Improve resiliency
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRRESTRICTS3CROSSREGIONREPLICATION",
            "Effect": "Deny",
            "Action": "s3:PutReplicationConfiguration",
            "Resource": "*"{% if ExemptedPrincipalArns %},
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }
            }{% endif %}
        }
    ]
}
```

[AWS-GR_RESTRICT_S3_DELETE_WITHOUT_MFA] Disallow delete actions on S3 buckets without MFA

Protect your S3 buckets by requiring multi-factor authentication (MFA) for delete actions. MFA adds an extra authentication code on top of a user name and password.

This is a preventive control with elective guidance based on service control policies (SCPs). By default, this control is not enabled. You can enable this control through the AWS Control Tower console, or though the AWS Control Tower APIs.

AWS service: Amazon S3

Control metadata

- **Control objective:** Protect data integrity
- **Implementation:** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control owner:** AWS Control Tower
- **Resource types:** `AWS::S3::Bucket`

Usage considerations

- This control supports configuration. It contains elements that are included by AWS Control Tower conditionally, based on the configuration you select. This control supports the following configuration parameters: **ExemptedPrincipalArns**. For more information, see Configure controls with parameters.

The artifact for this control is the following service control policy (SCP).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GRRESTRICTS3DELETEWITHOUTMFA",
            "Effect": "Deny",
            "Action": [
                "s3:DeleteObject",
                "s3:DeleteBucket"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": [
                        "false"
                    ]
                }{% if ExemptedPrincipalArns %},
                "ArnNotLike": {
                    "aws:PrincipalArn": {{ExemptedPrincipalArns}}
                }{% endif %}
            }
        }
    ]
}
```

[CT.CLOUDFORMATION.PR.1] Disallow management of resource types, modules, and hooks within the CloudFormation registry

This elective control disallows management of the following extension types in the CloudFormation registry: resource types, modules, and hooks. For more information about CloudFormation extensions, see Using the CloudFormationregistry.

A typical use case for this control is a situation in which you do not wish to allow your organization to register CloudFormation types. It prevents registration of types, and it prevents disabling existing CloudFormation hooks.

- **Control objective:** Protect configurations
- **Implementation** Service control policy (SCP)
- **Control behavior:** Preventive
- **Control guidance:** Elective
- **Control owner:** AWS Control Tower
- **Control ID:** CT.CLOUDFORMATION.PR.1
- **Severity:** Critical
- **AWS Service:** CloudFormation
- **Resource types:** `AWS::CloudFormation::HookDefaultVersion, AWS::CloudFormation::HookTypeConfig, AWS::CloudFormation::HookVersion, AWS::CloudFormation::ModuleDefaultVersion, AWS::CloudFormation::ModuleVersion, AWS::CloudFormation::ResourceDefaultVersion, AWS::CloudFormation::ResourceVersion`

The following example shows the SCP artifact for this control.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GRDISALLOWMODIFICATIONCFNREGISTRY",
            "Effect": "Deny",
```

```
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:DeregisterType",
                "cloudformation:SetTypeConfiguration",
                "cloudformation:SetTypeDefaultVersion",
                "cloudformation:PublishType"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": "arn:aws:iam::*:role/
AWSControlTowerExecution"
                }
            }
        }
    ]
}
```

Search for controls with AWS assistant

When you are logged into the AWS management console, the AWS assistant can help you find the controls that you need to manage your AWS Control Tower environment. You can type prompts to the AWS assistant, and find sumarized information about specific types or use cases for controls, such as these examples:

- What are the Digital Sovereignty controls?
- What controls in AWS Control Tower monitor S3?
- I have an AWS service running that uses Amazon ECS to host it, which controls from AWS Control Tower should I enable?
- What are preventive controls in AWS Control Tower?

Any time you'd like a paraphrase of information from the documentation, and a list of links for further reading, you can *ask AWS assistant*.

The AWS assistant is more proficient at summarizing information from the documentation than at giving complete lists of controls. The lists of controls you may receive are helpful starting points. The AWS assistant also provides tips for security and best practices.

Here's an example answer to the last example prompt.

What are preventive controls in AWS Control Tower?
Preventive controls in AWS Control Tower are an important part of the service's governance and compliance capabilities. Here's an overview of preventive controls in AWS Control Tower:
Purpose: Preventive controls are designed to disallow actions that could lead to policy violations or security risks. They help enforce configurations and prevent non-compliant resource changes.
Implementation: Preventive controls are implemented using Service Control Policies (SCPs), which are part of AWS Organizations.
Scope: These controls apply to entire organizational units (OUs) and affect every AWS account within the OU.
Function: They help express policy intentions and enforce governance rules across your AWS environment.
Examples of preventive controls:

- Requiring encryption at rest for certain services
- Enforcing multi-Availability Zone deployments for improved resiliency
- Restricting access to specific AWS Regions
- Digital Sovereignty: AWS Control Tower offers preventive controls that enhance digital sovereignty governance, covering areas such as data residency, granular access restrictions, encryption, and resiliency

Customization: While some controls are mandatory, others are strongly recommended or elective, allowing you to tailor your governance posture to your specific needs.
Management: You can view and manage these controls in the AWS Control Tower console, including a dedicated group for digital sovereignty-related controls
Exceptions: It's important to note that the root user and administrators in the management account have exceptions to these controls to prevent the account from becoming unusable.
When implementing preventive controls, it's crucial to follow the principle of least privilege and regularly review your control settings to ensure they align with your organization's security and compliance requirements. For the most up-to-date information on available controls and their specific functionalities, please refer to the official AWS Control Tower documentation.

Tagging EnabledControl resources in AWS Control Tower

You can add tags to EnabledControl resources by means of CloudFormation templates, through the AWS Control Tower console, and by calling the AWS Control Tower APIs.

> **Note**
>
> The AWS Control Tower GetEnabledControl, EnableControl, and ListTagsforResource API operations rely on the EnabledControl resource tagging functionality for proper drift reporting.

Required permissions

When you configure resource tags with CloudFormation, you must add the new ListTagsforResource IAM permission to the policy for the customer-managed role that you use to update your controls. If you do not add the permission, the CloudFormation template may have the tags, but CloudFormation cannot see them without the ability to call ListTagsforResource. If you already have created a role that updates your AWS Control Tower landing zone, that role probably has this permission in place already, because the same permission is required to view tags associated with the landing zone resource.

Step 1: Add the permissions

To tag a resource, update a tag, and enable proper drift reporting, three permissions are required, as shown in the example that follows.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "VisualEditor0",
   "Effect": "Allow",
   "Action": [
    "controltower:TagResource",
```

```
    "controltower:ListTagsForResource",
    "controltower:UntagResource"
   ],
   "Resource": "*"
  }
 ]
}
```

CloudFormation drift

If you do not add the proper permissions, you can experience some side-effects that we refer to as *false positive or false negative CloudFormation drift*. During CloudFormation drift, the **Detect drift** command in the CloudFormation console may not give reliable results. You also can encounter these side effects if you modify your CloudFormation resource outside the CloudFormation console.

> **We strongly recommend**
>
> If you provision a resource, including a tag, with CloudFormation, it is important to update the resource through CloudFormation only.

When you experience *false positive* CloudFormation drift, the CloudFormation console shows a **Modified** status (drifted) for a resource, although you are not aware of making any modifications. In this situation, the status means that you have not added the `ListTagsforResource` permission. When the permission is not present in the role, the `ReadHandler` cannot read the tags. CloudFormation returns an error, because it cannot tell whether the resource actually was modified. The error is surfaced as **Modified** status.

When you experience *false negative drift* the CloudFormation console shows a resource as unmodified, when in fact, it has been modified. This situation means that the AWS Control Tower `EnabledControl` resource has tags, but CloudFormation cannot retrieve those tags. In this case, two things must have occurred: the resource has been modified outside CloudFormation, which is not a recommended practice, and also the `ListTagsforResource` permission was not added to the policy.

Step 2. Add the tags to the resource

Here is an example CloudFormation resource template with tags added.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  myEnabledControlTest:
    Type: 'AWS::ControlTower::EnabledControl'
    Properties:
      ControlIdentifier: 'arn:aws:controltower:us-west-2::control/ZTCMZEXAMPLE'
      TargetIdentifier: 'arn:aws:organizations::012345678901:ou/o-exampleou/ou-xxxx-
f35g82v9'
      Tags:
        - Key: "K1"
          Value: "V1"
      Parameters:
        - Key: AllowedRegions
          Value:
            - us-west-2
            - us-west-1
            - us-east-1
```

For more information, see EnabledControl in the *AWS CloudFormation User Guide*.

Controls and compliance

Within AWS Control Tower, compliance refers to the state of a resource, when it is evaluated with respect to a deployed detective control, or a drift detection rule. Compliance in AWS Control Tower is related to drift — usually, a non-compliant resource is in a state of drift. AWS Control Tower controls embody rules of compliance. They help you identify compliant and non-compliant resources by helping identify drift.

When AWS Control Tower evaluates the compliance of resources, it reports the compliance results at the OU, account, and control levels. This section describes compliance status in detail, for controls, OUs, and accounts.

Compliance reporting is intended to let cloud administrators know when the resources associated with the accounts in their organization are compliant with established policies. When the resources are in compliance, builders can provision new AWS accounts quickly in a few clicks.

When we talk about compliance in AWS Control Tower, we do not intend the same meaning as compliance with governmental regulations, such as data privacy or health information standards. However, AWS Control Tower can assist your organization to comply with many governmental regulations, sometimes referred to as *frameworks*.

- For more information about how AWS Control Tower helps you maintain compliance with governmental regulations and industry standards, see Compliance Validation.
- For more information about how you can verify AWS Control Tower resource compliance during CloudFormation stack creation, see this blog post, How AWS Control Tower users can proactively verify compliance in AWS CloudFormationstacks.

For ongoing governance, administrators can enable pre-configured controls—clearly defined rules for security, operations, and compliance. These controls can:

- Prevent deployment of resources that don't conform to policies (by means of preventive controls, implemented with SCPs, or by means of proactive controls, implemented with CloudFormation hooks).
- Continuously monitor deployed resources for nonconformance (by means of detective controls, implemented with AWS Config rules).

Examples of compliance rules (controls) in AWS Control Tower:

- Detect Whether Public Write Access to Amazon S3 Buckets is Allowed
- Detect Whether Unrestricted Incoming TCP Traffic is Allowed

Examples of governmental compliance regulations (frameworks):

- The U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA)
- The European Union's General Data Protection Regulation of 2016 (GDPR)

How can administrators review compliance?

Compliance with detective controls is determined according to data retrieved from the AWS Config aggregator in the AWS Control Tower Audit account. You can review compliance status in the AWS Control Tower console, by subscribing to SNS topics that send email messages to the Audit account, or both.

Detective control status

To view the compliance status of detective controls in the AWS Control Tower console, select **Controls** in the left navigation, choose the control name from the controls table, and then scroll to the **Accounts** section on that control details page. Accounts may show a control compliance status of **Unknown** if any detective controls are misconfigured. For example, status **Unknown** often can appear due to account drift, such as **Moved account** drift. The **Unknown** status also can appear as a result of SCP drift.

> **Note**
>
> AWS Control Tower displays the compliance status of all AWS Config rules deployed into organizational units registered with AWS Control Tower, including rules that were activated outside of the AWS Control Tower console. To view the compliance status of all your Config rules, navigate to the **Account details** page in the AWS Control Tower console. You will see a list showing the compliance status of controls managed by AWS Control Tower and Config rules deployed outside of AWS Control Tower. You can identify any non-compliant AWS Config rule.

Preventive control status

The compliance status of preventive controls on an OU may be viewed on the **OU detail** page, by scrolling to the **Enabled controls** section. If any preventive controls are misconfigured for an OU, the **State** field for that OU may show the state of **Registration failed**, in the **Details** section near the top of the page. Preventive control misconfiguration is caused most often by SCP drift, which can occur if the control's SCP is modified or detached from the OU by means of the AWS Organizations console.

Proactive control status

The control compliance status also can be viewed on other pages:

- On the AWS Control Tower **Dashboard** page, by scrolling to the **Controls** section near the bottom of the page.
- On the **Control details** page, which you can view by selecting the name of a control on another page.

> **Note**
>
> The **State** of a control, as viewed in the AWS Control Tower console, reflects only the enabled or de-activated state of the control for a specific OU. This field does not reflect any information about the framework compliance status or the drift status of the landing zone environment. The control **State** and **Status** information is available in the console only. It is not available from the public API. To view the control status, navigate to the **Control details** page in the AWS Control Tower console.

Nested OUs and compliance

When an OU shows a status of **Noncompliant**, it means that one of the accounts directly under the OU contains noncompliant resources. The compliance status of an OU is not influenced by the compliance status of nested OUs under the OU, or the compliance status of any accounts that are not directly under the OU.

Other resources

If an account has any non-compliant resources, that account may be shown with **Noncompliant** status on the **OU** or **Account** page in the AWS Control Tower console. Details about the specific resources that have caused the non-compliant status are shown on the **Account details** page.

If an account shows **Compliant** status, that means it has no resources that are non-compliant; therefore, no resource details are shown on the **Account details** page, only an empty table.

Receive compliance status updates

To receive updates about compliance, you can subscribe to SNS topics that send notifications when resource compliance status changes. See Compliance notifications by SNS in the audit account, later in this chapter.

For more information on how AWS Control Tower collects information about resources, see the AWS Config Aggregator Documentation.

Drift changes the compliance status for OU and account resources

Drifted resources may be shown with status **Unknown** in the **Compliance** status field of the AWS Control Tower console. The **Unknown** state indicates that AWS Control Tower cannot determine the compliance status of the resource, because drift is present. Drift is not necessarily a detective control compliance violation. For more information about drift, see Detect and resolve drift in AWS Control Tower.

In another case of this type of drift, resources may be shown as compliant when they are not. If you delete an AWS Config rule, or if you turn off the Config recorder, compliance status may be reflected inaccurately in the console, because compliance no longer can be evaluated. For example, if you turn off the Config recorder, the last evaluated status continues to appear in the console. Similarly, if you delete an AWS Config rule, the resources covered by that rule always show to be compliant. In this situation, your environment could have some non-compliant resources that are not reported. Avoid deleting or turning off your AWS Config resources.

AWS Control Tower compliance status for OUs and accounts

Compliance is reported in the AWS Control Tower dashboard for accounts and OUs. This section lists the possible categories of compliance and non-compliance in AWS Control Tower, assuming that controls are enabled for an account or an OU.

- **For an account or OU:** A compliance status of **Compliant**, **Noncompliant**, or **Unknown** is possible. The compliance status refers to the status of the resources associated with a single account, or the status of all accounts in an OU that has multiple controls enabled on it. The account or OU compliance status can be found on the account or OU detail pages.

> **Note**
>
> The **State** of a control, as viewed in the AWS Control Tower console, reflects only the enabled or de-activated state of the control for a specific OU. This field does not reflect any information about the framework compliance status or the drift status of the landing zone environment. The control **State** and **Status** information is available in the AWS Control Tower console. Enabled controls can be viewed through the public API.

The following list gives more information about compliance status as reported specifically for OUs and their member accounts.

- **`Compliant`** – Compliance rules are properly in place. No violations have been detected for any resources. Controls are applied at the OU level, for all enrolled accounts in the OU, and their resources.
  - **Reported for:** Detective controls (AWS Config Rules)
  - **What it checks:**
    - Any individual detective control that's applied to the member accounts in an OU
    - Multiple detective controls that are applied to the member accounts in an OU
- **`Noncompliant`** – Compliance rules are in place. However, non-compliant resources have been detected in one or more member accounts in the OU.
  - **Reported for:** Detective controls (AWS Config Rules)
  - **What it checks:**
    - Any individual detective control that's applied to the member accounts in an OU
    - Multiple detective controls that are applied to the member accounts in an OU

The following status can be reported for any account, control, or OU.

Unknown – A compliance rule is broken or compliance cannot be guaranteed.

- **Reported for:**
  - Detective controls (AWS Config Rules)
  - Preventive controls (SCPs)

- **What it checks:**
  - Any detective control that's enabled on any accounts that are members of an OU. Controls are enabled at the OU level.
  - Any preventive control that's enabled on any accounts that are members of an OU. Controls are enabled at the OU level.
  - Basically anything with a compliance status (account, control, resource, or OU).

Drift prevention and notification

You can enable certain controls and subscribe to certain SNS notifications that help you maintain compliance in AWS Control Tower.

> **Note**
>
> AWS Control Tower will no longer be sending drift notifications to SNS topic for all customers on LZ4.0+. For customers on LZ4.0+ follow the EventBridge Notification setup.

Drift monitoring protection

AWS Control Tower provides passive and active methods of drift monitoring protection for preventive controls.

- **Passive protection:** AWS Organizations monitors and logs preventive control (SCP) drift.
- **Active protection:** The AWS Control Tower drift monitoring service actively scans the preventive control SCPs, on a regular basis.

AWS Control Tower notifies you by means of SNS messaging, if drift is detected.

Drift prevention

Some controls prevent modification of compliance reporting mechanisms.

- Disallow Changes to AWS Config Rules Set Up by AWS Control Tower (Mandatory, preventive control)
- Disallow Deletion of AWS Config Aggregation Authorizations Created by AWS Control Tower (Mandatory, preventive control)

- Disallow Changes to Tags Created by AWS Control Tower for AWS Config Resources (Mandatory, preventive control)
- Disallow Configuration Changes to AWS Config (Mandatory, preventive control)

In contrast to preventive controls, detective controls notify you of resources that violate the associated AWS Config rule.

To receive SNS notifications about drift and control compliance

For information about how to receive appropriate drift and control compliance notifications by Amazon SNS, see Compliance notifications by SNS in the audit account.

Publishers and subscribers for SNS topics

The `aws-controltower-AllConfigNotifications` topic:

- The `AWS::Config::DeliveryChannel` resource is configured to send notifications about configuration changes to this topic.
- The possible types of notifications that AWS Config can send are defined in the **Amazon SNS Topic** section of the AWS Config documentation.
- The `AWS::CloudTrail::Trail` resource is configured to send notifications of log file delivery to this topic.
- You may subscribe to this topic.

The `aws-controltower-SecurityNotifications` topic:

- The `AWS::Events::Rule` resource is configured to send notifications about AWS Config Rule compliance changes (one of the SNS notification types) to this topic.
- The `aws-controltower-NotificationForwarder` Lambda function is subscribed to this topic, and it forwards the SNS notifications to the `aws-controltower-AggregateSecurityNotifications` topic.

The `aws-controltower-AggregateSecurityNotifications` topic:

- This topic receives notifications from `aws-controltower-SecurityNotifications`, forwarded by the Lambda function.
- It also receives drift notifications in the home Region.

- When AWS Control Tower creates the topic, a subscription is added for the audit account email address, and you must confirm the subscription.

> **Note**
>
> The endpoint, such as an email address, must confirm each subscription, SNS doesn't send messages to an endpoint until the subscription is confirmed.

Compliance notifications by SNS in the audit account

> **Note**
>
> AWS Control Tower will no longer be sending drift notifications to SNS topic for all customers on LZ4.0+. For customers on LZ4.0+ follow the EventBridge Notification setup.

To receive compliance change notifications in email sent to your audit account, subscribe to this Amazon SNS topic:

`arn:aws:sns:`*`AWSRegion`*`:`*`AuditAccount`*`:aws-controltower-AggregateSecurityNotifications`

When subscribing, substitute your actual AWS Control Tower home Region and audit account information into the topic name shown. You can subscribe to SNS topics that receive notifications about each supported AWS Region in which you run AWS Control Tower.

SNS topics and notifications you can receive

- The `aws-controltower-AllConfigNotifications` topic:

  It receives notifications from AWS Config regarding compliance, noncompliance, and change. It also receives notification from AWS CloudTrail on log file delivery.
- The `aws-controltower-SecurityNotifications` topic:

  One of these topics exists for each supported AWS Region. It receives compliance, noncompliance, and change notifications from AWS Config in that Region. It forwards all incoming notifications to `aws-controltower-AggregateSecurityNotifications`

- The `aws-controltower-AggregateSecurityNotifications` topic:

  This topic exists in each supported AWS Region. It receives compliance change notifications from the region-specific `aws-controltower-SecurityNotifications` topics. Additionally, in the home Region, it also receives drift notifications.

Other considerations about SNS topics:

- All of these topics exist and receive notifications in the Audit account.
- By default, the Audit account email address is subscribed to the `aws-controltower-AggregateSecurityNotifications` SNS topic.
- SNS topics in AWS Control Tower are extremely noisy, by design. For example, AWS Config sends a notification every time AWS Config discovers a new resource.
- Administrators who wish to filter out specific types of notifications from an SNS topic can create an AWS Lambda function and subscribe it to the SNS topic. Alternatively, you can set up an EventBridge rule to filter notifications, as described in this support article, How can I be notified when an AWS resource is non-compliant using AWS Config?
- You can apply an Amazon SNS subscription filter policy to reduce the volume of notifications that a subscription receives. The filter policy also applies to the default Audit account email subscription. The filter policy controls which messages Amazon SNS delivers to the subscription from that topic. For more information about filtering messages, see Amazon SNS message filtering in the *Amazon Simple Notification Service Developer Guide*.
- AWS Config notifications contain a JSON object.
- AWS Control Tower drift notifications appear in plain text.

The AWS Config SNS topic policy

The AWS Config SNS topic policy contains the `aws:SourceOrgID` condition key. The policy is shown in the following example.

```
SNSAllConfigurationTopicPolicy:
   Type: AWS::SNS::TopicPolicy
   Properties:
     Topics:
       - !Ref SNSAllConfigurationTopic
     PolicyDocument:
       Statement:
```

```
          - Sid: AWSSNSPolicy
            Action:
              - sns:Publish
            Effect: Allow
            Resource: !Ref SNSAllConfigurationTopic
            Principal:
              Service:
                - cloudtrail.amazonaws.com
                - config.amazonaws.com
            Condition:
              StringEquals:
                aws:SourceOrgID: !Ref OrganizationId
```

AWS Region availability for AWS Control Tower controls

Warning

The data previously displayed on this page was outdated and has been removed. This page will be removed in a future release.

To retrieve current information about control Region availability and other control metadata, see View control details.

Tables of control metadata

Warning

The data previously displayed on this page was outdated and has been removed. This page will be removed in a future release.

To retrieve current control metadata, see View control details.

All global identifiers for AWS Control Tower controls

> **Warning**
>
> The data previously displayed on this page was outdated and has been removed. This page will be removed in a future release.

A *global identifier* for a control is a unique identifier, independent of the Region in which a control is deployed. Global identifiers are associated with controls that are included in AWS Control Catalog, which is a compendium of controls from several AWS services.

The global identifier fulfills the final portion of the Amazon Resource Name (ARN), which is of the following form:

```
arn:{PARTITION}:controlcatalog:::control/{CONTROL_CATALOG_OPAQUE_ID}
```

To obtain the complete global identifier programmatically, with the full ARN, you can call the `GetControl` and `ListControls` APIs. You also can view the complete global ARN in the AWS Control Tower console, on the **Control details** page.

For more information about control identifiers and ARNs, see Resource identifiers for APIs and controls.

Document history

- **Latest documentation update:** June 12, 2025

The following table describes important changes to the *AWS Control Tower Controls Reference Guide*. For notifications about documentation updates, you can subscribe to the RSS feed.

Change	Description	Date
Renamed Controls Library to Control Catalog	Renamed for consistency.	June 12, 2025
Added declarative policy controls	A new control type.	December 1, 2024
Added integrated AWS Config controls	Another control type manageable through AWS Control Tower.	November 21, 2024
Added RCP controls	A new control type.	November 15, 2024
Content navigation update	Updated the presentation of control categories.	October 16, 2024
Content organization update	Updated the guide to improve discoverability of content for AI.	September 12, 2024
New AWS Control Catalog API calls added	AWS Control Tower adds a reference to two new Control Catalog APIs for controls.	August 1, 2024
Controls Reference Guide published	AWS Control Tower adds a reference guide specifically for controls.	April 21, 2024

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.