OpenSecOps SOAR

Technical Design Specification

CONTENTS

Document Versions

Version	Date	Changes	Author
1.0	2024-02-14	First version	Peter Bengtson
1.1	2024-09-25	FTR updates	Peter Bengtson
1.2	2025-04-07	Replaced “Delegat” by “OpenSecOps” throughout	Peter Bengtson

1 Solution Overview

1.1 Background

In any system, especially in a complex enterprise-level one, there is a need to bring together many heterogeneous subsystems on many and widely different levels of abstraction to a whole so that security conditions and situations that arise in any of these subsystems can be identified, weighed, and compared against one another, then acted upon consistently whether by people or by automation.

All such subsystems handle incidents and monitoring differently, which means that integrations with human-facing processes either need to be duplicated many times, which is impractical and wasteful; or that one of them, usually a SIEM, is chosen as the integration point for those processes.

AWS recognizes its customers' need to integrate specialised external software for monitoring and security incident detection, whether on the infrastructural level or the application level. AWS has therefore provided means to integrate all security findings, regardless of source, into a single workflow upon which all kinds of security issue orchestration, automation, and response procedures can be based uniformly.

The AWS service providing this integration point is Security Hub, which can be set up to receive security information from a plethora of external software such as QRadar, ServiceNow, Jira, etc, and integrate it seamlessly with AWS-native security findings. Security Hub also provides event handling based on this data.

OpenSecOps SOAR ("Security Orchestration, Automation and Response") builds 24/7 security control management and incident handling on top of this, including email notification and ServiceNow or Jira ticketing as well as a large number of auto-remediatory actions on the infrastructural level.

1.2 Solution Description

OpenSecOps SOAR is a system built on top of AWS Security Hub, the natural integration point for all security-related state information on AWS. Using the capabilities provided by OpenSecOps SOAR, uniform processes are established around all AWS monitoring and incident handling. This functionality is also available for the application level.

2 Design Objectives

2.1 Scope

OpenSecOps SOAR handles all procedural automation around two main types of issues, *incidents* and *controls*. They are similar in many ways.

For **incidents**, such as a server going rogue, or a user failing to authenticate too many times, there is a degree of severity and perhaps a need for immediate action. OpenSecOps SOAR will create a ServiceNow or Jira ticket to SOC for each incident. If the incident is severe, OpenSecOps SOAR will also snapshot and terminate any rogue instance and create a ServiceNow or Jira ticket to SOC for further forensic investigation.

For **controls**, representing ongoing security conditions such as a security group being too open, there is severity but also *state*: controls in a FAILED state need to be remedied with various degrees of urgency. If the control can be auto-remediated, OpenSecOps SOAR performs the auto-remediation and notifies the team. If the control cannot be automatically fixed, OpenSecOps SOAR creates a ServiceNow or Jira ticket to the team responsible for creating the issue. Tickets are created with various levels of severity and SLAs.

Security Hub automatically revisits all control findings periodically, typically several times a day, to re-evaluate them for changes in compliance. Should a FAILED finding become PASSED, meaning that the underlying issue was fixed, the ServiceNow or Jira ticket will be closed automatically, relieving the teams from keeping track manually.

For each control failure issue, each autoremediation and each security and/or application incident, counts and weighted statistics are kept in several dimensions (team, project, account, environment type, etc). If the optional AI integration is activated, this data is used to create weekly AI-based reports to security management and to the individual teams. It could also be used to create dashboards of the most compliant/non-compliant accounts and teams or to determine where to best concentrate educational efforts. Actions, including auto-remediatory ones, can also be based on this data.

2.2 Design Goals

1. To fully automate the handling of **security incidents** and **security controls** vis-a-vis builder teams and SOC, including email notification to stakeholders as well as full handling of the entire ServiceNow or Jira ticketing cycle.

2. To automatically **remediate** FAILED security controls for the most common infrastructural issues.

3. To automatically **snapshot,** for forensic purposes, and **terminate** misbehaving EC2 instances involved in security incidents (e.g. Bitcoin mining, contacting Tor servers, etc).

4. To maintain security control **issue counts** and/or **severity sums** for each account, team, environment (etc), based on the type of issues encountered

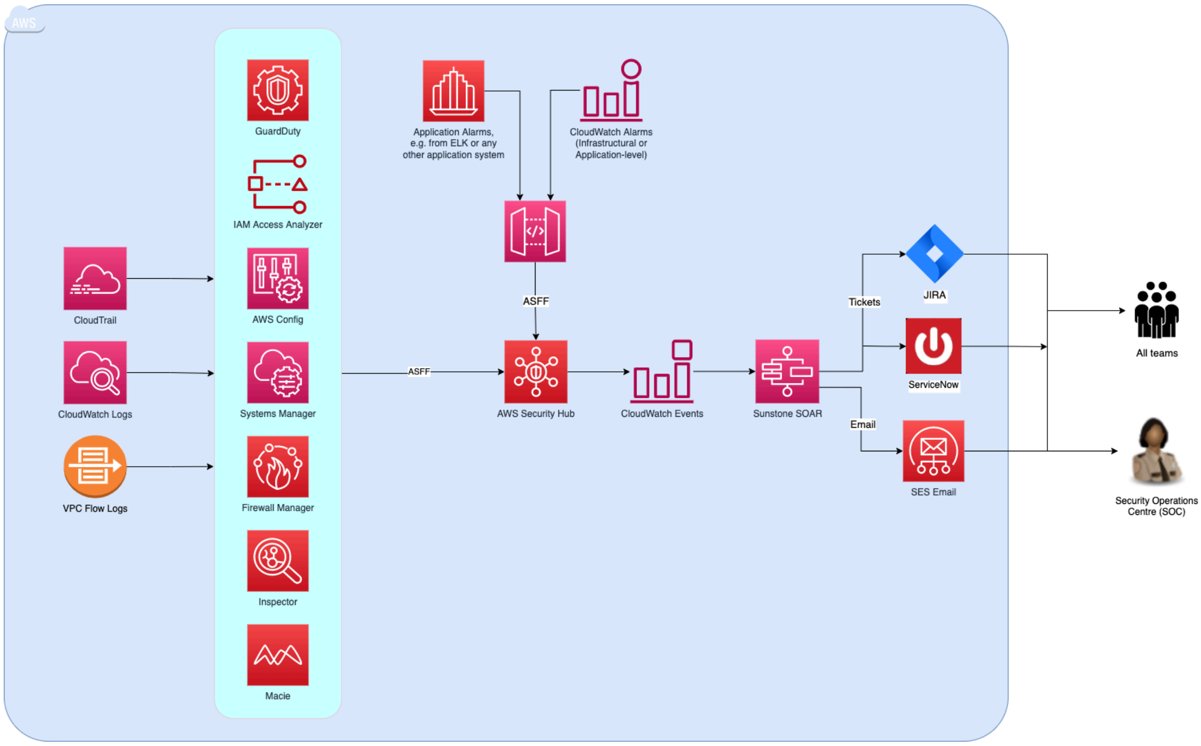
and the environment to which they apply, in order to be able to specify different courses of actions (including disciplinary ones) in each environment.

5. To allow the **application domain** to leverage the same handling and automation facilities simply by adopting the use of the API Gateway POST endpoint for creating incidents in Security Hub and thereby also in ServiceNow or Jira, and also by simple CloudWatch alarm naming conventions.

3 Design

3.1 Abstract View

The following diagram is a conceptual overview of the OpenSecOps SOAR system:



3.1.1 AWS Data

The main input of the system on the AWS side consists of logs from CloudWatch and CloudTrail, along with VPC Flow Logs and a wealth of other internal data and metadata. All these are fed internally to a range of security products which analyse data in real time using machine learning and other techniques. The results from these analyses are fed directly to AWS Security Hub using the ASFF finding format (https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-findings-format.html).

3.1.2 Application-level Data

A third analysis path, that of application-level alarms and notifications is represented by the ELK icon. As long as the application level uses the API Gateway endpoint for creating incidents and/or the predefined CloudWatch alarm naming conventions, application-level findings will be treated exactly like all other findings and will be part of the same ticketing flow.

As long as these two methods are used for all incident creation, uniformity and manageability at scale is guaranteed.

3.1.3 Security Hub CloudWatch Events

AWS Security Hub processes all data, generating CloudWatch events as each finding is processed and re-processed over time.

OpenSecOps SOAR uses only these ASFF events as its driving input and thus works in a fully decoupled fashion. The Security Hub ASFF events trigger all logic, from auto-remediation of non-compliant AWS infrastructure to ServiceNow or Jira ticketing and email notification via AWS SES to the involved teams as well as to SOC.

3.1.4 OpenSecOps SOAR's Ticketing and Notification Monopoly

As can be seen from the diagram, all interaction with humans is driven by the SOAR layer. The same thing applies to application level incidents such as from ELK: *all* ticketing is done by OpenSecOps SOAR – never by direct calls from ELK to ServiceNow or Jira.

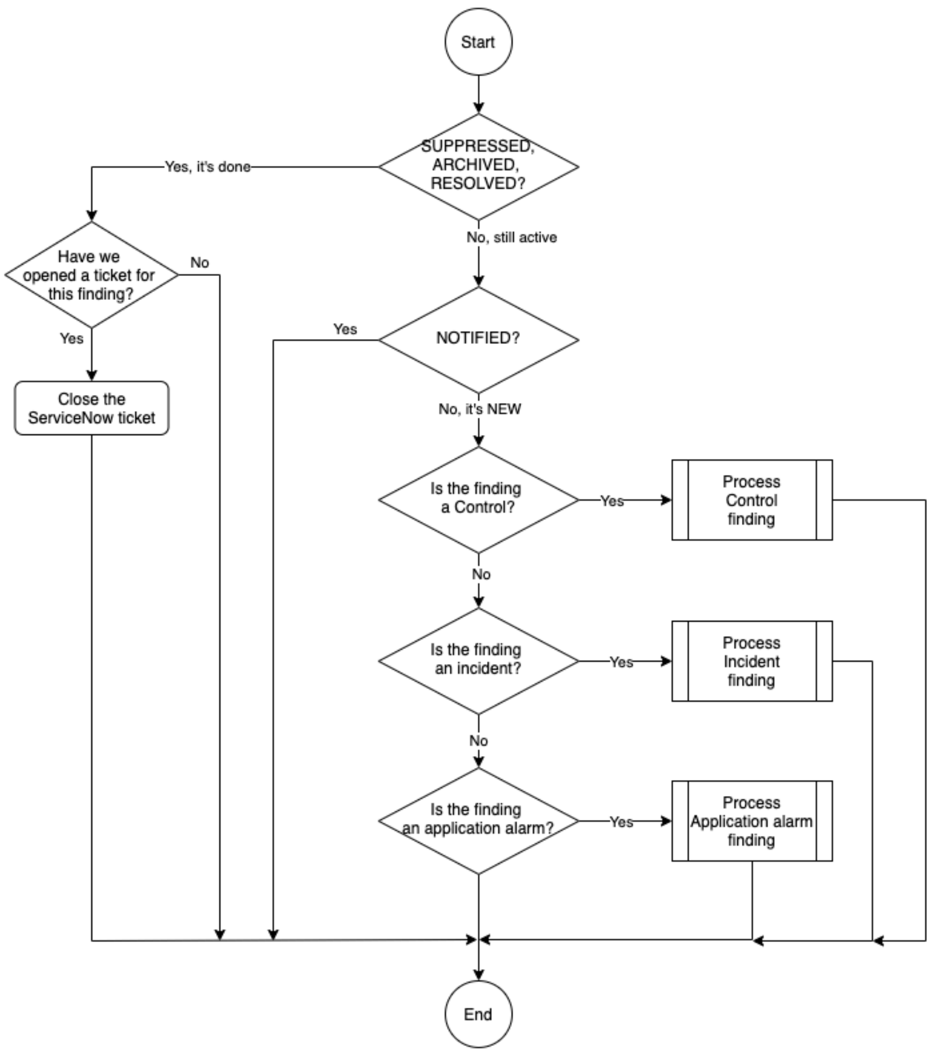
This principle applies throughout: in order to guarantee system scalability, ticketing and notification should be done from one central place and in a uniform way. OpenSecOps SOAR is designed to fulfil this role.

3.2 Detailed View

OpenSecOps SOAR is triggered by either of two CloudWatch Events whenever a Security Hub finding is generated or modified ("new finding created" and "finding updated", respectively).

The reception of either event results in an AWS Step Function being invoked with the underlying ASFF finding as input. If necessary, additional nested state machines are invoked to handle controls and incidents, respectively.

The top level of the processing done conceptually looks like this and is thus the same for all automation:



First, the workflow state of the finding is examined. A state of SUPPRESSED, ARCHIVED, or RESOLVED indicates that the finding no longer is relevant and has been closed. OpenSecOps SOAR checks whether the finding has an open ticket associated with it. If so, it closes the ticket and notifies the stakeholders involved.

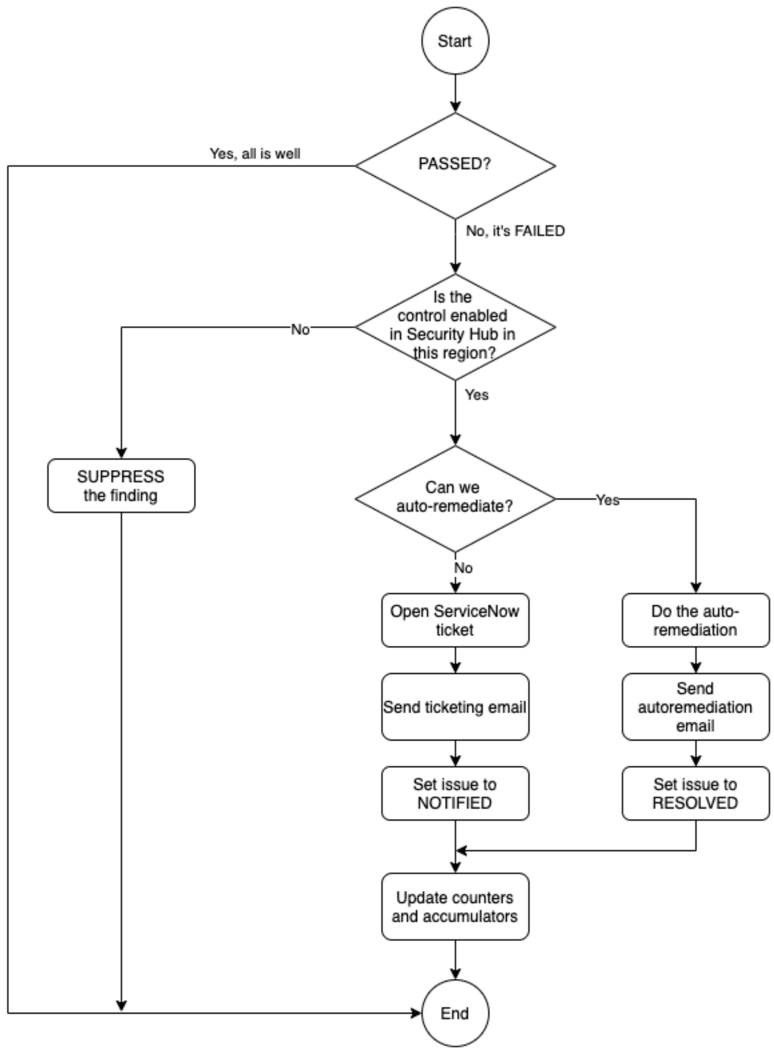
Otherwise, the finding is still active and relevant. OpenSecOps SOAR then checks to see whether it has processed the finding before to the extent that stakeholders already have been notified through email or a ticket. If the state is NOTIFIED, then notification has been done, and no further actions are necessary this time around.

If the state is none of the above, then it is NEW, indicating the issue has to be processed by OpenSecOps SOAR to determine what needs to be done with it.

After this point, the actions to take vary depending on what generated the finding.

3.2.1 Processing of Security Hub Controls

Controls have a compliance state of either PASSED or FAILED. The goal here is to make FAILED controls reach a PASSED state using auto-remediation or by triggering manual intervention by the teams themselves.



3.2.1.1 PASSED Controls

If the control is in a compliant PASSED state, nothing needs to be done with it and we simply terminate processing. Security Hub will soon set the workflow state to RESOLVED, which will trigger ticket cleanup, etc.

3.2.1.2 FAILED Controls

If the control is FAILED, however, we first check whether the control is enabled in Security Hub. If it is not, we suppress the finding as we're not interested in it. Suppressing the finding will modify it: this will trigger another round through the state machine in which the top level will see that it is SUPPRESSED and take care of ticket cleanup, etc.

If the finding is in a FAILED compliance state and the control is enabled, some kind of action is required.

3.2.1.3 Auto-Remediation of a Control

OpenSecOps SOAR first checks whether there is an auto-remediatory action available. If there is, OpenSecOps SOAR attempts to remediate the failed control.

If auto-remediation could be performed and was successful, stakeholders are notified via email and the finding is set to RESOLVED. No ticket is issued as the matter has been dealt with.

3.2.1.4 Manual Remediation of a Control

If auto-remediation wasn’t possible or didn’t succeed, OpenSecOps SOAR creates a ticket in ServiceNow or Jira and associates it with the finding, which then is set to NOTIFIED, signifying that stakeholders have been informed and that the system is waiting for the issue to be addressed. OpenSecOps SOAR also notifies all stakeholders via email.

3.2.1.5 Control Conformance Statistics

Finally, statistics are gathered for FAILED issues. This is done regardless of whether auto-remediation fixes it or not as the teams are still responsible and accountable for the infrastructural flaws in their infrastructure. Auto-remediation is not intended to replace basic knowledge of AWS best practices, at least not in the long run.

The finding is examined and the severity of the issue (INFORMATIONAL, LOW, MEDIUM, HIGH, CRITICAL) is weighted depending on the type of environment.

Counts and weighted statistics are kept in several dimensions (team, project, account, environment type, etc) for each control issue. If the optional AI integration is activated, this data is used to create weekly AI-based reports to security management and to the individual teams. It could also be used to create dashboards of the most compliant/non-compliant accounts and teams or to determine where to best concentrate educational efforts. Actions, including auto-remediatory ones, can also be based on this data.

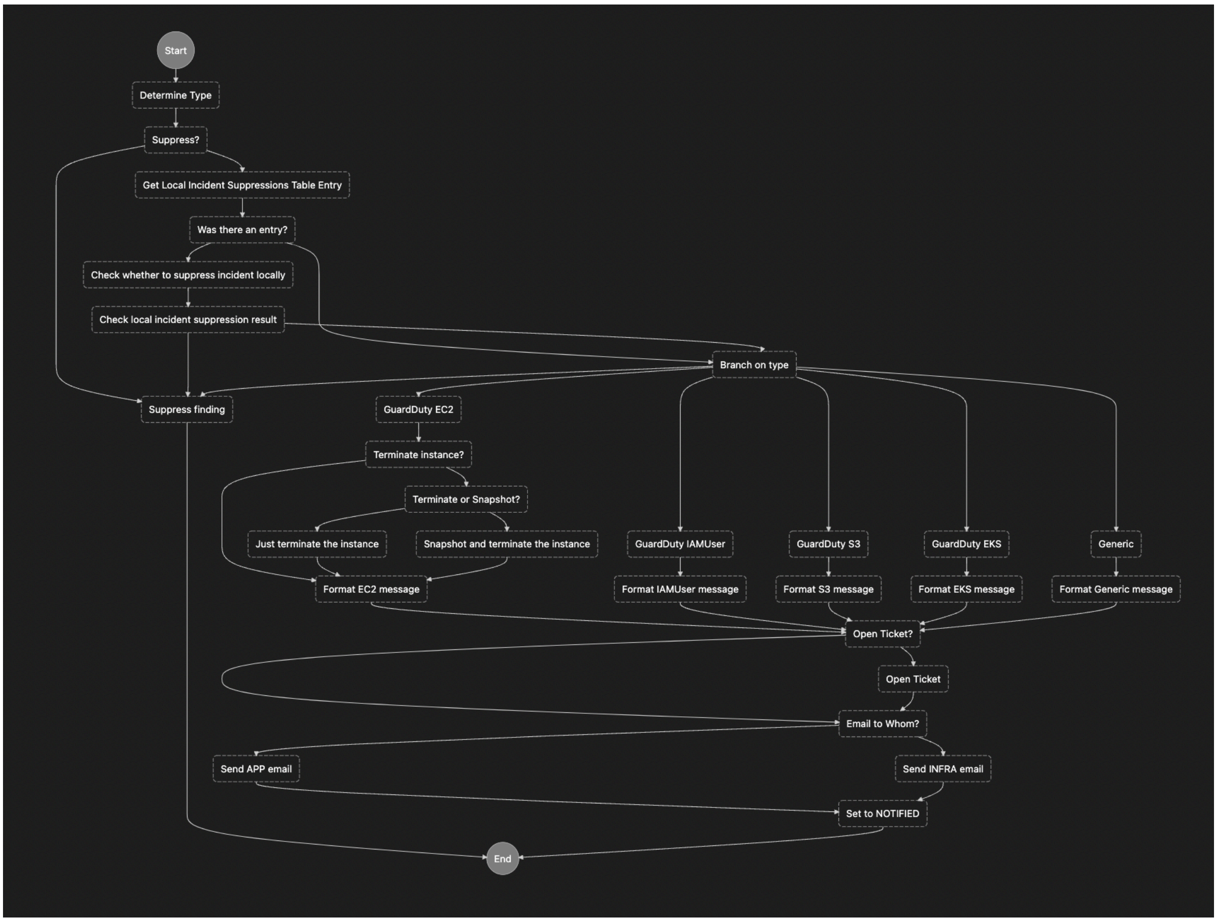
3.2.2 Processing of GuardDuty Incidents

A security incident is an event that may indicate that an organisation's systems or data have been compromised or that measures put in place to protect them have failed. Incidents have a Severity (INFORMATIONAL, LOW, MEDIUM, HIGH, CRITICAL) just like controls, but they are one-off events and thus do not have any conformance status (of FAILED or PASSED).

GuardDuty generates incidents of four different types:

1. EC2 related incidents (TTPs such as Bitcoin mining, Brute Force attacks, contacting Tor servers, etc),
2. S3 related incidents such as disabling global read protection, and,

3. IAM related incidents such as unexpected logons, password strength reduction, excessive number of log-on failures, etc.
4. Kubernetes-related incidents, similar to the EC2 incidents but for K8S pods.



Here's a list of all incidents covered by GuardDuty (NB that the list scrolls): https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html. Please refer to Chapter 8 for a complete listing of incidents detected by GuardDuty.

For each incident type we keep a parameterised list of severities on which OpenSecOps SOAR should act. For instance, we might want to notify stakeholders of all EC2 related incidents of a MEDIUM and higher severity and suppress everything on the INFORMATIONAL and LOW severity levels. Likewise for S3 and IAM related incidents: each type has a list of severities on which to act.

For each incident, counts and weighted statistics are kept in several dimensions (team, project, account, environment type, etc). If the optional AI integration is activated, this data is used to create weekly AI-based reports to security management and to the individual teams. It could also be used to create dashboards of the most compliant/non-compliant accounts and teams or to determine where to best concentrate educational efforts. Actions, including auto-remediatory ones, can also be based on this data.

3.2.2.1 EC2 Incidents

If an EC2 incident occurs of a MEDIUM, HIGH or CRITICAL severity level, then OpenSecOps SOAR will immediately snapshot the instance for forensic purposes, then terminate it. SOC will be informed and a ServiceNow or Jira ticket to SOC will be created for investigation. The team will not be issued a ticket, for security reasons. OpenSecOps SOAR processes all compromised instances in parallel, reducing the attack window to a minimum.

For an example of this exact logic at work at Goldman Sachs, see https://www.youtube.com/watch?v=CR4_a-TO_gw&list=PLNWtfvMjCtMxt34My_Ctq5SpfV1jhXfgq&index=10

If the incident is not of a MEDIUM, HIGH or CRITICAL severity and the severity doesn’t appear in the list of EC2 severities to process, the finding is suppressed. Otherwise SOC is issued a ticket to investigate.

3.2.2.2 S3 Incidents

Most S3 incidents are already covered by Security Hub standard controls, which means the severity list can be set to cover only the most critical cases, e.g. to “HIGH,CRITICAL” or simply set to blank. The default setting is “LOW,MEDIUM,HIGH,CRITICAL”.

3.2.2.3 IAM Incidents

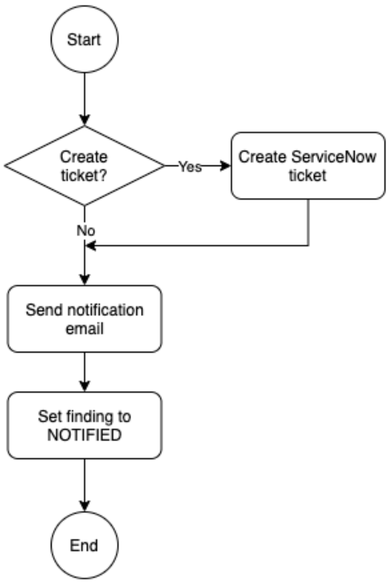
Similar to S3 incidents, most IAM incidents are already covered by Security Hub standard controls. This means that the severity list can be set to cover only the most critical cases, e.g. to “HIGH,CRITICAL” or simply set to blank. The default setting is “LOW,MEDIUM,HIGH,CRITICAL”.

3.2.3 Processing of Application Alarms

Application alarms are incidents. They are created from the application layer in two ways: (1) by calling the API Gateway ingestion endpoint for creating incidents, or (2) by following a predefined naming standard for CloudWatch alarms.

Application alarms are handled just like all other findings. Notification and ticketing is done automatically. It is also possible to write automation based on application alarm findings.

Initially, application alarm handling will be simple, creating only email notifications and tickets to the teams. Automation can be added at a later stage, if so desired.



4 Creating Incidents from the Application Layer

There are two ways to create incidents. One is via an HTTPS endpoint that can be called from anywhere, including from other systems and from on-premise infrastructure. The other is by a simple naming convention for CloudWatch alarms. Both can be used by the infrastructural level as well as the application level.

Indeed, for the long-term sustainability of the system, it is essential that application level incidents are created only by either of these two means.

4.1 By Using the API Gateway Incident Ingestion Endpoint

This is the most flexible way of creating incidents for SOC or for the team using an account, as it allows detailed information about the incident to be given for the benefit of the recipients. It is a simple API Gateway endpoint to which an HTTP POST is made, something which allows for easy integration into any type of software.

4.1.1 Usage

Gateway endpoint URL for Security Hub and ServiceNow or Jira incidents:

- https://example-company-domain.io/soar/prod/incidents

Do a POST to the above URL with a JSON body containing the following fields in dictionary form (cf the next section for a full JSON example):

account_id	a 12-character string of decimal digits
region	a string like “eu-central-1”
title	the title of the incident. Will be truncated to 100 chars
description	a multi-line, exhaustive description of the issue
severity	LOW | MEDIUM | HIGH | CRITICAL
remediation_text	instructions on how to fix it. Be as detailed as you can.
remediation_url	a URL giving more remediation info
finding_id	optional (default: request ID)
finding_id_prefix	optional (default: '')
namespace	optional (default: 'alarms')
ticket_destination	optional ('TEAM' or 'SOC', default: 'TEAM')

environment	optional (default: '', max 4 characters)
incident_domain	('APP' or 'INFRA', default: 'APP')

Notes on the fields:

- The **title** should be as brief as possible and will be truncated in emails and by ServiceNow or Jira, so put the important information at the beginning of the title. The environment will automatically be added as a prefix. Make sure different incidents have different titles; don’t use a catch-all title with the important information hidden away in the description. In ServiceNow or Jira, the title is all that is visible before the incident is opened, so make sure it is terse *and* descriptive.

- The **description** should be as informative as you possibly can make it. It can run to as many lines as you wish. You can use <br> and <BR> to create hard breaks if necessary, but no other HTML tags as the description is not HTML.

- The **severity** must be one of LOW, MEDIUM, HIGH, or CRITICAL. The standard severities to use are LOW and MEDIUM.

The two highest severities, HIGH and CRITICAL, are show-stoppers that require the team to address the issue with extreme priority. HIGH and CRITICAL will create P-level incidents in ServiceNow or Jira, so use them with extreme discretion and consult SOC beforehand. Be aware that some P incidents automatically result in reports being sent to external banking authorities which in their turn are obliged to investigate. You do not want to trigger this process unduly.

All of the severities have SLAs in ServiceNow or Jira; when setting up an alarm, make sure you know what the SLA for your alarm is. You are effectively forcing the team operating the account to work according to the schedule prescribed by that SLA.

- The **ticket_destination** can be either TEAM (the default) or SOC. If TEAM, the ticket will be created for, and email sent to, the team operating the account with the given account_id. If SOC, the ticket must be security-related and will be sent to SOC for processing as a security incident. Please confer with SOC before creating security incidents for them to react on. Also be aware that SOC scans all incidents to get an overview of the overall state of the system, including those to the teams. However, they will not normally act on TEAM incidents.

- The **namespace** can be anything you decide, but it’s easiest to use it to differentiate between contexts: application, microservice, team, squad, infra type or even product. The default is ''alarms''. Don't overly differentiate, though, as the namespace value is useful in any future automation that might be created for automatic remediation.

- The **incident_domain** indicates whether the incident belongs to the application domain or to the infrastructural domain. It defaults to APP, meaning the application domain. For an infrastructural incident, the value to supply is INFRA. This allows the system to create the incident for the appropriate team, when two teams split the infra and the application responsibilities for the same account.

If you specify INFRA, the incident will be created on the ServiceNow or Jira queue for the infrastructural team, given by the soar:account:service-now:project-queue tag value on the account. If APP (the default), the ServiceNow or Jira queue will instead be given by the soar:account:service-now:project-queue:app tag on the account. If the soar:account:service-now:project-queue:app tag isn't present on the account, the soar:account:service-now:project-queue tag value will be used instead. (If Jira is used, substitute jira for service-now in the above paragraph.)

Similarly for the email distribution list used: for APP, the DL is given by the account tag soar:account:distribution-list:app. For INFRA, the tag used is soar:account:distribution-list. The APP list tag value, if not present on the account, defaults to the INFRA list value, just as for the ServiceNow or Jira project queue above.

The endpoint returns, on a 200 status, the ASFF finding just created. If the operation failed, a message will be returned and an error HTTP status code.

Also note that API consumers are obliged to act on the returned status code if they want to detect whether an incident was successfully created or not. It is extremely unlikely that ServiceNow or Jira would be unavailable, but make sure consumers verify the 200 return status and take appropriate action for their particular application should incident creation fail for any reason. A simple such action would be to retry the operation with exponential fallback.

Please also make sure that API consumers do not create incidents promiscuously. An incident raised every minute is a data point, not an incident. Indeed, reducing the frequency of incidents should be a continuously ongoing process. If an incident is created 20 times a day, for instance, the API consumer must be made to actively try to reduce incident frequency.

Also, all incidents should be acted upon, or they should not be generated in the first place.

4.1.2 Example in Python

```python
payload = {
    'ticket_destination': 'SOC',
    'namespace': 'Public-API-Gateway',
    'account_id': '111122223333',
    'region': 'eu-central-1',
    'title': 'Intrusion attempt detected',
    'description': 'Skynet has detected an intrusion attempt at <etc, etc>',
    'severity': 'HIGH',
    'remediation_text': 'Please send Arnold Schwarzenegger.',
    'remediation_url': 'https://example.com/somewhere/'
}

response = requests.post(URL, json=payload)
print(response.text)
```

4.1.3 Implementation

The API Gateway Incident Ingestion Endpoint is implemented as a SAM project containing a single Lambda.

4.2 By Using CloudWatch Alarm Naming Conventions

This is the quickest and most lightweight way of creating incidents, provided the incident can be based on a CloudWatch alarm.

OpenSecOps SOAR monitors every CloudWatch alarm in all regions of all member accounts. By simply including certain strings in the CloudWatch alarm name, incidents will be created in Security Hub and ServiceNow or Jira depending on that information.

This method cannot be used to create incidents for SOC. To create security incidents for SOC, the API Gateway Incident Ingestion Point must be used.

4.2.1 Usage

To create an incident when a CloudWatch alarm is raised, simply include the severity level in capitals at the very beginning or end of its name, followed or preceded by a hyphen, respectively.

The regex used is (^|-)(INFORMATIONAL|LOW|MEDIUM|HIGH|CRITICAL)(-|$).

4.2.2 Severity Examples

A few example CloudWatch Alarm names that will create incidents when triggered:

- INFORMATIONAL-App-Alarms_e3540a15f0
- LOW-Alarm-with-ML
- Whatever-Alarm-You-Have-12345-MEDIUM
- HIGH-as-a-kite-2418
- Its-getting-CRITICAL

A few names that will not:

- informational-App-Alarms_e3540a15f0
- Low-Alarm-with-ML
- Whatever-MEDIUM-Alarm-You-Have-12345
- HIGHas-a-kite-2418
- ItsgettingCRITICAL

4.2.3 Infrastructural vs Application Incidents

The incident will by default be created for the application domain, i.e. as if we set the incident_domain in an API Gateway Incident Ingestion entry point call to APP (the default). To achieve this, the account tags soar:service-now:project-queue:app and soar:distribution-list:app will be used to reach the application team.

To instead create an incident pertaining to the infrastructural domain, i.e. as we do by setting incident_domain to INFRA in the API Gateway Incident Ingestion request, just include the string INFRA in capitals anywhere in the alarm name. This will instead use the values supplied by the tags soar:service-now:project-queue and soar:distribution-list for the account.

NB: The tag soar:service-now:project-queue:app will default to the value of soar:service-now:project-queue if not present. Likewise, soar:distribution-list:app will default to the value of soar:distribution-list. Please read Section 6.1.1 for the full details.

4.2.4 Infrastructural Domain Examples

A few names that will create an infrastructural incident:

- INFORMATIONAL-INFRA-App-Alarms_e3540a15f0
- LOW-INFRA-Alarm-with-ML
- Whatever-INFRACTIVE-Alarm-You-Have-12345-MEDIUM

A few names that won't:

- INFORMATIONAL-Infra-App-Alarms_e3540a15f0
- LOW-Infra-Alarm-with-ML
- Whatever-infractive-Alarm-You-Have-12345-MEDIUM

4.2.5 Severities

The standard severities to use are LOW and MEDIUM.

The two highest severities, HIGH and CRITICAL, are show-stoppers that require the team to address the issue with extreme priority. Use them with extreme discretion and consult SOC beforehand.

All of the severities have SLAs in ServiceNow or Jira; when setting up an alarm, do make sure you know the SLA for your chosen severity, because you are effectively forcing the team operating the account to work according to the schedule prescribed by that SLA.

4.2.6 Implementation

The CloudWatch Alarm Bridge is implemented as a SAM project.

Apart from a single lambda doing the main work, the project also sets up an extra event bus from all member accounts to the security account, so that the lambda doesn't need to be duplicated in every account.

5 Email Examples

5.1 Incident

INCIDENT: Stack Set drift for 'AWSControlTowerSecurityResources'

External SOAR ×

no-reply@sunstone.email
to peter.bengtson+accounts-org, peter.bengtson+soar-cc

02:12 (10 hours ago)

MEDIUM INCIDENT in account "Org" (622238921895, OU: ROOT), region eu-north-1:

STACK SET DRIFT FOR 'AWSCONTROLTOWERSECURITYRESOURCES'

The stack set 'AWSControlTowerSecurityResources' is not IN_SYNC:

* Total number of templates: 1
* Drifted: 1
* In sync: 0
* Failed: 0

The full drift specification is available from the CloudFormation console for stack set 'AWSControlTowerSecurityResources' in account 622238921895, region eu-north-1.

Resource ARN: 622238921895
Resource type: AwsAccountId

Type: Software and Configuration Checks/Security Automation/Stack Set Drift

Product name: Default
Finding ID: AWSControlTowerSecurityResources-5fcb47b4-c167-4505-9575-34ae9777a247
Created at: 2022-05-16T00:12:33Z

Email sent by Schibsted Sunstone SOAR to: peter.bengtson+accounts-org@schibsted.com

Ticket ID: bff6a8b2-2284-4ced-a2d0-2d531fea3e54

ACTIONS TAKEN: None.
ACTIONS REQUIRED: Instructions for remediating a drifted CloudFormation stack set can be found here:

Remediation instructions can also be found here: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

Thank you.

/ Schibsted Sunstone SOAR


```
============================================================
Resources:
- Id: '622238921895'
  Region: eu-north-1
  Type: AwsAccountId

============================================================
ProductFields:
  IncidentDomain: INFRA
  TicketDestination: TEAM
  aws/securityhub/CompanyName: Personal
  aws/securityhub/FindingId: arn:aws:securityhub:eu-north-1:622238921895:product/622238921895/default/AWSControlTowerSecurityResources-
5fcb47b4-c167-4505-9575-34ae9777a247
  aws/securityhub/ProductName: Default

============================================================
AccountNew: 'No'
Client: Unspecified
Email: peter.bengtson+accounts-org@schibsted.com
Environment: Unspecified
Id: '622238921895'
Name: Org
OrganizationalUnit: ROOT
ProjectId: Unspecified
ProjectIdApp: Unspecified
ProjectName: Unspecified
Tags: {}
Tallies:
- account:622238921895:Org
- ou:ROOT
- client:Unspecified
- project_name:Unspecified
- project_id:Unspecified
- team:Unspecified
- environment:Unspecified
Team: Unspecified
TeamEmail: peter.bengtson+accounts-org@schibsted.com
TeamEmailApp: peter.bengtson+accounts-org@schibsted.com

============================================================
```

5.2 Auto-Remediation

AUTOFIXED: S3 buckets should have server-side encryption enabled

External SOAR ×

no-reply@sunstone.email
to me, peter.bengtson+soc

Thu, 12 May, 19:40 (4 days ago)

MEDIUM issue AUTOREMEDIATED in account "PeterBengtsonAccount" (139118282244, OU: Sandbox), region eu-north-1:

S3.4 S3 BUCKETS SHOULD HAVE SERVER-SIDE ENCRYPTION ENABLED

This AWS control checks that your Amazon S3 bucket either has Amazon S3 default encryption enabled or that the S3 bucket policy explicitly denies put-object requests without server side encryption.

Resource ARN: arn:aws:s3:::my-unencrypted-public-bucket
Resource type: AwsS3Bucket

Type: Software and Configuration Checks/Industry and Regulatory Standards/AWS-Foundational-Security-Best-Practices

Finding ID: arn:aws:securityhub:eu-north-1:139118282244:subscription/aws-foundational-security-best-practices/v/1.0.0/S3.4/finding/20f402ae-edfa-4d40-a502-7fa3a13325ae
First observed: 2022-05-12T16:42:33.993Z
Last observed: 2022-05-12T16:42:37.147Z

Email sent by Schibsted Sunstone SOAR to: peter.bengtson@schibsted.com

The issue has been resolved according to the following principles:
https://docs.aws.amazon.com/console/securityhub/S3.4/remediation

ACTIONS TAKEN: Encryption at rest using AES-256 has been enabled.

ACTIONS REQUIRED ON YOUR PART: Please update your infrastructural code to prevent this security issue from arising again at the next deployment.

Thank you.

/ Schibsted Sunstone SOAR


```
============================================================
Resources:
- Details:
    AwsS3Bucket:
      CreatedAt: '2022-05-12T16:40:59.000Z'
      OwnerId: aa2e611d8073f9b1d1be192ef445bae371cda0ab2593c2b580ed3cc0a61c78e7
  Id: arn:aws:s3:::my-unencrypted-public-bucket
  Partition: aws
  Region: eu-north-1
  Type: AwsS3Bucket

============================================================
ProductFields:
  ControlId: S3.4
  RecommendationUrl: https://docs.aws.amazon.com/console/securityhub/S3.4/remediation
  RelatedAWSResources:0/name: securityhub-s3-bucket-server-side-encryption-enabled-469a8cc4
```


(etc)

5.3 Ticket Opened

TEAM FIX: API Gateway REST API stages should be configured to use SSL certificates for backend au... External SOAR ×

no-reply@sunstone.email
to peter.bengtson+accounts-org, peter.bengtson+soc

Wed, 11 May, 13:25 (5 days ago)

MEDIUM issue in account "Org" (622238921895, OU: ROOT), region eu-north-1:

APIGATEWAY.2 API GATEWAY REST API STAGES SHOULD BE CONFIGURED TO USE SSL CERTIFICATES FOR BACKEND AUTHENTICATION

This control checks whether Amazon API Gateway REST API stages have SSL certificates configured that backend systems can use to authenticate that incoming requests are from the API Gateway.

Resource ARN: arn:aws:apigateway:eu-north-1::/restapis/3gp488csyh/stages/Prod
Resource type: AwsApiGatewayStage

Type: Software and Configuration Checks/Industry and Regulatory Standards/AWS-Foundational-Security-Best-Practices

Finding ID: arn:aws:securityhub:eu-north-1:622238921895:subscription/aws-foundational-security-best-practices/v/1.0.0/APIGateway.2/finding/f40908ba-5593-4ae4-8d74-f15b46f83eb5

First observed: 2022-05-11T11:24:42.182Z
Last observed: 2022-05-11T11:24:55.178Z

Email sent by Schibsted Sunstone SOAR to: peter.bengtson+accounts-org@schibsted.com

Ticket ID: 30c56406-b11d-4afa-a053-569bf4d0dfff

ACTIONS TAKEN: None.

ACTIONS REQUIRED ON YOUR PART: Please update your infrastructural code to prevent this security issue from arising again at the next deployment.

Remediation instructions can also be found here: https://docs.aws.amazon.com/console/securityhub/APIGateway.2/remediation

Thank you for your immediate attention to this security issue.

/ Schibsted Sunstone SOAR

```
============================================================
Resources:
- Details:
  AwsApiGatewayStage:
   AccessLogSettings:
    DestinationArn: arn:aws:logs:eu-north-1:622238921895:log-group:create-sec-hub-incident-AccessLogGroup-eurfKKBWoPfH
    Format: '{"requestTime":"$context.requestTime","requestId":"$context.requestId","httpMethod":"$context.httpMethod","path":"$
```

(etc)

5.4 Ticket Closed

CLOSED: API Gateway REST API stages should be configured to use SSL certificates for backend auth... External SOAR ×

no-reply@sunstone.email
to peter.bengtson+accounts-org, peter.bengtson+soar-cc

Sat, 14 May, 14:32 (2 days ago)

MEDIUM issue CLOSED in account "Org" (622238921895, OU: ROOT), region eu-north-1:

APIGATEWAY.2 API GATEWAY REST API STAGES SHOULD BE CONFIGURED TO USE SSL CERTIFICATES FOR BACKEND AUTHENTICATION

This control checks whether Amazon API Gateway REST API stages have SSL certificates configured that backend systems can use to authenticate that incoming requests are from the API Gateway.

Resource ARN: arn:aws:apigateway:eu-north-1::/restapis/3gp488csyh/stages/Stage
Resource type: AwsApiGatewayStage

Type: Software and Configuration Checks/Industry and Regulatory Standards/AWS-Foundational-Security-Best-Practices

Finding ID: arn:aws:securityhub:eu-north-1:622238921895:subscription/aws-foundational-security-best-practices/v/1.0.0/APIGateway.2/finding/afe4fa70-2705-44c5-85a2-6415aed66874
First observed: 2022-05-11T11:24:34.697Z
Last observed: 2022-05-14T12:32:29.842Z

Email sent by Schibsted Sunstone SOAR to: peter.bengtson+accounts-org@schibsted.com

Ticket ID: 3c2f32e4-e907-49cc-a14b-7a6c31ab5a1b

Workflow Status: NOTIFIED
Compliance Status: FAILED

Reason for closing: The underlying infrastructural issue may have been fixed, the finding may have been suppressed, or the control may have been disabled in this region.

ACTIONS REQUIRED ON YOUR PART: None

Thank you.

/ Schibsted Sunstone SOAR

==
Resources:

(etc)

6 Enabled Auto-Remediations

The following is a list of the infrastructural remediations performed automatically by OpenSecOps SOAR at the time of writing. Further autoremediations will be added on occasion.

S3.2: S3 Bucket Publicly Readable

S3 buckets set to be publicly readable will be automatically closed for public access unless the tag "soar:s3:request-publicly-readable" is present on the bucket. SOC will be notified of your request via an incident. You will be contacted about your reasons for wanting to create a world-readable file server.

Please note that adding the tag after the fact will not reopen it for world access. You must redeploy the bucket with the correct tag, or manually add the tag then re-enable world access again.

S3.3: S3 Bucket Publicly Writable

S3 buckets set to be publicly writable will be automatically closed for public access unless the tag "soar:s3:request-publicly-writable" is present on the bucket at creation time. SOC will be notified of your request via an incident. You will be contacted about your reasons for wanting to create a world-writable file server.

Please note that adding the tag after the fact will not reopen it for world access. You must redeploy the bucket with the correct tag, or manually add the tag then re-enable world access again.

S3.10: S3 buckets with versioning enabled should have lifecycle policies configured

A lifecycle configuration will be added to the bucket. Noncurrent versions will be deleted after a year, and incomplete uploads after a day..

EC2.7: Default EBS Encryption Not Enabled

Enables default encryption of EBS volumes at the account level if not enabled. This will result in all new EBS volumes being encrypted. It will not affect running EBS volumes or their snapshots; only newly provisioned EBS volumes are affected.

CIS1.3: IAM User Credentials not used for 90 days

IAM Users who haven't used their credentials for 90 days will have their credentials and access codes revoked. The IAM User itself will remain.

RDS.2: Prohibit Public RDS Access

RDS databases set to be publicly accessible will have public access disabled.

RDS.4: RDS cluster snapshots and database snapshots should be encrypted at rest

An unencrypted DB cluster or instance snapshot will be deleted if empty. Otherwise, an attempt to encrypt the snapshot will be made, if the DB engine supports the operation (Aurora does *not*). If not supported, a TEAMFIX ticket will be issued instead.

RDS.6: Enhanced monitoring should be configured for RDS DB instances

Enables enhanced monitoring for your RDS DB instances. Enhanced Monitoring provides real-time metrics of the operating system that your RDS DB instance runs on. The monitoring period is set to 60 seconds.

RDS.11: RDS instances should have automatic backups enabled

Ensures Amazon Relational Database Service instances have automated backups enabled and the backup retention period is greater than or equal to seven days.

RDS.13: RDS automatic minor version upgrades should be enabled

Ensures that the latest minor version updates to the relational database management system (RDBMS) always 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.

RDS.17: RDS DB instances should be configured to copy tags to snapshots

Snapshots should be tagged in the same way as their parent RDS database instances. This auto-remediation ensures that snapshots inherit the tags of their parent database instances.

CIS2.8: Enable Customer CMK Key Rotation

Customer CMK KMS Keys without yearly key rotation will have such rotation enabled automatically.

CIS2.9: Enable VPC Flow Logs

VPCs without Flow Logs will have Flow Logs enabled and dump REJECTed packages to a new CloudWatch Log group using a new Role and Policy.

CIS4.1: Disable World Ingress to Port 22

World ingress to port 22 will be removed entirely. This also applies to cases where port 22 is included in a range (such as ports 0 to 65535) in which case the whole range will be removed.

CIS4.2: Disable World Ingress to Port 3389

World ingress to port 3389 will be removed entirely. This also applies to cases where port 3389 is included in a range (such as ports 0 to 65535) in which case the whole range will be removed.

EC2.2: VPC Default Security Group Should Not Allow Traffic

The default Security Group for a VPC is always created with full ingress and egress rights, which is reason to avoid using them at all. This remediation removes all ingress and egress rights for a VPC Default Security Group, provided the security group is untouched. If any changes have been done to the SG, it will be kept as is and a TEAMFIX ticket will be created instead.

EC2.4: Terminate Instances Stopped For More Than 30 Days

Instances that are left in a stopped state in the system are a security risk. This remediation terminates any instance left stopped for more than 30 days.

EC2.15: EC2 subnets should not automatically assign public IP addresses

If a VPC subnet is configured to automatically assign public IP addresses, then this setting is disabled.

ECR.1: ECR private repositories should have image scanning configured

If an ECR private repository doesn’t have image scanning configured, this remediation will configure it to scan uploaded Docker images using enhanced scanning once when the image is uploaded.

ECR.3: ECR repositories should have at least one lifecycle policy configured

If an ECR repository doesn’t have a lifecycle policy, this remediation will add one that keeps only the last uploaded Docker image.

ECS.2: ECS services should not have public IP addresses assigned to them automatically

If an ECS service assigns public IP addresses to its tasks, this will be disabled.

ECS.12: ECS clusters should use Container Insights

ECS clusters will have Container Insights activated if not enabled.

PCI.EC2.3: Release Unused Elastic IPs Older Than 30 Days

Unattached Elastic IP addresses older than a month are a security risk. This remediation will remove any EIP older than 30 days.

DynamoDB.2: DynamoDB Tables Should Have Point-In-Time Recovery Enabled

Backups help you to recover more quickly from a security incident. They also strengthen the resilience of your systems. DynamoDB point-in-time recovery automates backups for DynamoDB tables. It reduces the time to recover from accidental delete or write operations. DynamoDB tables that have PITR enabled can be restored to any point in time in the last 35 days. To disable the PITR auto-remediation for a specific table, add the tag “soar:dynamodb:no-pit-recovery” to the DynamoDB table at creation time.

ELB.1: Application Load Balancer should be configured to redirect all HTTP requests to HTTPS

This autoremediation will set up HTTP to HTTPS redirection for public facing ALBs with a matching SSL certificate. If no such certificate is detected, a ticket will be created to the team instead.

ELB.4: Application Load Balancer should be configured to drop invalid HTTP headers

By default, Application Load Balancers are not configured to drop invalid HTTP header values. Removing these header values prevents HTTP desync attacks.

ELB.5: Application and Classic Load Balancers logging should be enabled

Creates an S3 bucket for load balancer access logs and sets up replication of its contents to the Log Archive account. This guarantees that load balancer logs always are collected and aggregated for compliance.

7 Security Hub Controls

Below are the Security Hub controls typically enabled in a region, ordered by severity from the highest to the lowest. At the time of writing, Security Hub supports over 400 security controls, the number growing continuously as AWS services evolve and best practices change.

The severities have the following meanings:

Label

The severity value of the finding. The allowed values are the following.

- `INFORMATIONAL` - No issue was found.
- `LOW` - The issue does not require action on its own.
- `MEDIUM` - The issue must be addressed but not urgently.
- `HIGH` - The issue must be addressed as a priority.
- `CRITICAL` - The issue must be remediated immediately to avoid it escalating.

Severity	ID	Title
Critical	EC2.19	Security groups should not allow unrestricted access to ports with high risk
Critical	CodeBuild.1	CodeBuild GitHub or Bitbucket source repository URLs should use OAuth
Critical	CodeBuild.2	CodeBuild project environment variables should not contain clear text credentials
Critical	DMS.1	Database Migration Service replication instances should not be public
Critical	EC2.1	EBS snapshots should not be public, determined by the ability to be restorable by anyone
Critical	ES.2	Elasticsearch Service domains should be in a VPC
Critical	IAM.4	IAM root user access key should not exist
Critical	KMS.3	AWS KMS keys should not be deleted unintentionally
Critical	Lambda.1	Lambda function policies should prohibit public access
Critical	RDS.1	RDS snapshot should be private
Critical	RDS.2	RDS DB Instances should prohibit public access, determined by the PubliclyAccessible configuration
Critical	Redshift.1	Amazon Redshift clusters should prohibit public access
Critical	S3.2	S3 buckets should prohibit public read access
Critical	S3.3	S3 buckets should prohibit public write access
Critical	SSM.4	SSM documents should not be public

■ High	EC2.18	Security groups should only allow unrestricted incoming traffic for authorized ports
■ High	EC2.2	The VPC default security group should not allow inbound and outbound traffic
■ High	ElasticBeanstalk.2	Elastic Beanstalk managed platform updates should be enabled
■ High	SageMaker.1	Amazon SageMaker notebook instances should not have direct internet access
■ High	CloudTrail.1	CloudTrail should be enabled and configured with at least one multi-region trail
■ High	EC2.9	EC2 instances should not have a public IPv4 address
■ High	ECS.1	Amazon ECS task definitions should have secure networking modes and user definitions
■ High	ECS.2	ECS services should not have public IP addresses assigned to them automatically
■ High	EMR.1	Amazon Elastic MapReduce cluster master nodes should not have public IP addresses
■ High	GuardDuty.1	GuardDuty should be enabled
■ High	IAM.1	IAM policies should not allow full "*" administrative privileges
■ High	RDS.18	RDS instances should be deployed in a VPC
■ High	S3.6	S3 permissions granted to other AWS accounts in bucket policies should be restricted
■ High	S3.8	S3 Block Public Access setting should be enabled at the bucket-level
■ High	SSM.2	EC2 instances managed by Systems Manager should have a patch compliance status of COMPLIANT after a patch installation

■ Medium	ACM.1	ACM certificates should be renewed after a specified time period
■ Medium	IAM.3	IAM users' access keys should be rotated every 90 days or less
■ Medium	SecretsManager.2	Secrets Manager secrets configured with automatic rotation should rotate successfully
■ Medium	EC2.15	EC2 subnets should not automatically assign public IP addresses
■ Medium	APIGateway.1	API Gateway REST and WebSocket API execution logging should be enabled
■ Medium	RDS.14	Amazon Aurora clusters should have backtracking enabled
■ Medium	EFS.2	Amazon EFS volumes should be in backup plans
■ Medium	APIGateway.4	API Gateway should be associated with a WAF Web ACL
■ Medium	APIGateway.5	API Gateway REST API cache data should be encrypted at rest
■ Medium	CloudTrail.2	CloudTrail should have encryption at-rest enabled
■ Medium	Config.1	AWS Config should be enabled
■ Medium	DynamoDB.1	DynamoDB tables should automatically scale capacity with demand
■ Medium	DynamoDB.2	DynamoDB tables should have point-in-time recovery enabled
■ Medium	DynamoDB.3	DynamoDB Accelerator (DAX) clusters should be encrypted at rest
■ Medium	EC2.3	Attached EBS volumes should be encrypted at-rest
■ Medium	EC2.4	Stopped EC2 instances should be removed after a specified time period
■ Medium	EC2.6	VPC flow logging should be enabled in all VPCs
■ Medium	EC2.7	EBS default encryption should be enabled
■ Medium	EFS.1	Elastic File System should be configured to encrypt file data at-rest using AWS KMS

■ Medium	ELB.2	Classic Load Balancers with SSL/HTTPS listeners should use a certificate provided by AWS Certificate Manager
■ Medium	ELB.3	Classic Load Balancer listeners should be configured with HTTPS or TLS termination
■ Medium	ELB.5	Application and Classic Load Balancers logging should be enabled
■ Medium	ELB.6	Application Load Balancer deletion protection should be enabled
■ Medium	ELB.7	Classic Load Balancers should have connection draining enabled
■ Medium	ELB.8	Classic Load Balancers with HTTPS/SSL listeners should use a predefined security policy that has strong configuration
■ Medium	ELBv2.1	Application Load Balancer should be configured to redirect all HTTP requests to HTTPS
■ Medium	ES.1	Elasticsearch domains should have encryption at-rest enabled
■ Medium	ES.3	Elasticsearch domains should encrypt data sent between nodes
■ Medium	ES.4	Elasticsearch domain error logging to CloudWatch Logs should be enabled
■ Medium	ES.5	Elasticsearch domains should have audit logging enabled
■ Medium	ES.6	Elasticsearch domains should have at least three data nodes
■ Medium	ES.7	Elasticsearch domains should be configured with at least three dedicated master nodes
■ Medium	ES.8	Connections to Elasticsearch domains should be encrypted using TLS 1.2
■ Medium	IAM.5	MFA should be enabled for all IAM users that have a console password
■ Medium	IAM.7	Password policies for IAM users should have strong configurations
■ Medium	IAM.8	Unused IAM user credentials should be removed
■ Medium	Lambda.2	Lambda functions should use supported runtimes
■ Medium	RDS.15	RDS DB clusters should be configured for multiple Availability Zones
■ Medium	RDS.3	RDS DB instances should have encryption at-rest enabled

■ Medium	RDS.4	RDS cluster snapshots and database snapshots should be encrypted at rest
■ Medium	RDS.5	RDS DB instances should be configured with multiple Availability Zones
■ Medium	RDS.9	Database logging should be enabled
■ Medium	Redshift.2	Connections to Amazon Redshift clusters should be encrypted in transit
■ Medium	Redshift.3	Amazon Redshift clusters should have automatic snapshots enabled
■ Medium	Redshift.4	Amazon Redshift clusters should have audit logging enabled
■ Medium	Redshift.6	Amazon Redshift should have automatic upgrades to major versions enabled
■ Medium	Redshift.7	Redshift clusters should use enhanced VPC routing
■ Medium	S3.4	S3 buckets should have server-side encryption enabled
■ Medium	SNS.1	SNS topics should be encrypted at-rest using AWS KMS
■ Medium	SQS.1	Amazon SQS queues should be encrypted at rest
■ Medium	SSM.1	EC2 instances should be managed by AWS Systems Manager
■ Medium	SecretsManager.1	Secrets Manager secrets should have automatic rotation enabled
■ Medium	SecretsManager.3	Remove unused Secrets Manager secrets
■ Medium	SecretsManager.4	Secrets Manager secrets should be rotated within a specified number of days
■ Medium	S3.5	S3 buckets should require requests to use Secure Socket Layer

■ Low	RDS.17	RDS DB instances should be configured to copy tags to snapshots
■ Low	IAM.2	IAM users should not have IAM policies attached
■ Low	RDS.16	RDS DB clusters should be configured to copy tags to snapshots
■ Low	EC2.16	Unused Network Access Control Lists should be removed
■ Low	AutoScaling.1	Auto scaling groups associated with a load balancer should use load balancer health checks
■ Low	CloudTrail.4	CloudTrail log file validation should be enabled
■ Low	CloudTrail.5	CloudTrail trails should be integrated with Amazon CloudWatch Logs
■ Low	ElasticBeanstalk.1	Elastic Beanstalk environments should have enhanced health reporting enabled
■ Low	RDS.6	Enhanced monitoring should be configured for RDS DB instances
■ Low	RDS.7	RDS clusters should have deletion protection enabled
■ Low	RDS.8	RDS DB instances should have deletion protection enabled
■ Low	SSM.3	EC2 instances managed by Systems Manager should have an association compliance status of COMPLIANT

8 GuardDuty Findings

The following partial list of GuardDuty findings is meant to give an indication of the depth and breadth of security monitoring GuardDuty provides on AWS, the scope of which is quite often overlooked even by seasoned security professionals.

GuardDuty monitoring provides deep, AI and ML-based security monitoring within the following areas:

- EC2 finding types
- EKS Runtime Monitoring finding types
- IAM finding types
- Kubernetes audit logs finding types
- Lambda Protection finding types
- Malware Protection finding types
- RDS Protection finding types
- S3 finding types

The following sections give examples from the EC2, S3, and IAM sections, but the scope is always expanding and changing according to best practices.

For the full list, see
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html.

8.1 EC2 Finding Types

The following findings are specific to Amazon EC2 resources and always have a Resource Type of Instance. The severity and details of the findings differ based on the Resource Role, which indicates whether the EC2 instance was the target of suspicious activity or the actor performing the activity.

OpenSecOps SOAR will terminate the EC2 instance associated with a finding if the finding is of severity HIGH or CRITICAL. It will also create a ServiceNow or Jira ticket to SOC for investigation, and email notifications to the team as well as to SOC.

- Backdoor:EC2/C&CActivity.B
- Backdoor:EC2/C&CActivity.B!DNS
- Backdoor:EC2/DenialOfService.Dns
- Backdoor:EC2/DenialOfService.Tcp
- Backdoor:EC2/DenialOfService.Udp
- Backdoor:EC2/DenialOfService.UdpOnTcpPorts
- Backdoor:EC2/DenialOfService.UnusualProtocol
- Backdoor:EC2/Spambot
- Behavior:EC2/NetworkPortUnusual
- Behavior:EC2/TrafficVolumeUnusual
- CryptoCurrency:EC2/BitcoinTool.B
- CryptoCurrency:EC2/BitcoinTool.B!DNS
- Impact:EC2/AbusedDomainRequest.Reputation
- Impact:EC2/BitcoinDomainRequest.Reputation
- Impact:EC2/MaliciousDomainRequest.Reputation

- o Impact:EC2/PortSweep
- o Impact:EC2/SuspiciousDomainRequest.Reputation
- o Impact:EC2/WinRMBruteForce
- o Recon:EC2/PortProbeEMRUnprotectedPort
- o Recon:EC2/PortProbeUnprotectedPort
- o Recon:EC2/Portscan
- o Trojan:EC2/BlackholeTraffic
- o Trojan:EC2/BlackholeTraffic!DNS
- o Trojan:EC2/DGADomainRequest.B
- o Trojan:EC2/DGADomainRequest.C!DNS
- o Trojan:EC2/DNSDataExfiltration
- o Trojan:EC2/DriveBySourceTraffic!DNS
- o Trojan:EC2/DropPoint
- o Trojan:EC2/DropPoint!DNS
- o Trojan:EC2/PhishingDomainRequest!DNS
- o UnauthorizedAccess:EC2/MaliciousIPCaller.Custom
- o UnauthorizedAccess:EC2/MetadataDNSRebind
- o UnauthorizedAccess:EC2/RDPBruteForce
- o UnauthorizedAccess:EC2/SSHBruteForce
- o UnauthorizedAccess:EC2/TorClient
- o UnauthorizedAccess:EC2/TorRelay

8.1.1 Backdoor:EC2/C&CActivity.B

An EC2 instance is querying an IP that is associated with a known command and control server.

Default severity: High

This finding informs you that the listed instance within your AWS environment is querying an IP associated with a known command and control (C&C) server. The listed instance might be compromised. Command and control servers are computers that issue commands to members of a botnet. A botnet is a collection of internet-connected devices which might include PCs, servers, mobile devices, and Internet of Things devices, that are infected and controlled by a common type of malware. Botnets are often used to distribute malware and gather misappropriated information, such as credit card numbers. Depending on the purpose and structure of the botnet, the C&C server might also issue commands to begin a distributed denial-of-service (DDoS) attack.

8.1.2 Backdoor:EC2/C&CActivity.B!DNS

An EC2 instance is querying a domain name that is associated with a known command and control server.

Default severity: High

This finding informs you that the listed instance within your AWS environment is querying a domain name associated with a known command and control (C&C) server. The listed instance might be compromised. Command and control servers are computers that issue commands to members of a botnet. A botnet is a collection of internet-connected devices which might include PCs, servers, mobile devices, and Internet of Things devices, that are infected and controlled by a common type of malware. Botnets are often used to distribute malware and gather misappropriated information, such as credit card numbers. Depending on the purpose and structure of the botnet, the C&C server might also issue commands to begin a distributed denial-of-service (DDoS) attack.

To test this finding type, you can make a DNS request from your instance (using dig for Linux or nslookup for Windows) against a test domain guarddutyc2activityb.com.

8.1.3 Backdoor:EC2/DenialOfService.Dns

An EC2 instance is behaving in a manner that may indicate it is being used to perform a Denial of Service (DoS) attack using the DNS protocol.

Default severity: High

This finding informs you that the listed EC2 instance within your AWS environment is generating a large volume of outbound DNS traffic. This may indicate that the listed instance is compromised and being used to perform denial-of-service (DoS) attacks using DNS protocol.

This finding detects DoS attacks only against publicly routable IP addresses, which are primary targets of DoS attacks.

8.1.4 Backdoor:EC2/DenialOfService.Tcp

An EC2 instance is behaving in a manner indicating it is being used to perform a Denial of Service (DoS) attack using the TCP protocol.

Default severity: High

This finding informs you that the listed EC2 instance within your AWS environment is generating a large volume of outbound TCP traffic. This may indicate that the instance is compromised and being used to perform denial-of-service (DoS) attacks using TCP protocol.

This finding detects DoS attacks only against publicly routable IP addresses, which are primary targets of DoS attacks

8.1.5 Backdoor:EC2/DenialOfService.Udp

An EC2 instance is behaving in a manner indicating it is being used to perform a Denial of Service (DoS) attack using the UDP protocol.

Default severity: High

This finding informs you that the listed EC2 instance within your AWS environment is generating a large volume of outbound UDP traffic. This may indicate that the listed instance is compromised and being used to perform denial-of-service (DoS) attacks using UDP protocol.

This finding detects DoS attacks only against publicly routable IP addresses, which are primary targets of DoS attacks.

8.1.6 Backdoor:EC2/DenialOfService.UdpOnTcpPorts

An EC2 instance is behaving in a manner that may indicate it is being used to perform a Denial of Service (DoS) attack using the UDP protocol on a TCP port.

Default severity: High

This finding informs you that the listed EC2 instance within your AWS environment is generating a large volume of outbound UDP traffic targeted to a port that is typically used for TCP communication. This may indicate that the listed instance is compromised and being used to perform a denial-of-service (DoS) attacks using UDP protocol on a TCP port.

This finding detects DoS attacks only against publicly routable IP addresses, which are primary targets of DoS attacks.

8.1.7 Backdoor:EC2/DenialOfService.UnusualProtocol

An EC2 instance is behaving in a manner that may indicate it is being used to perform a Denial of Service (DoS) attack using an unusual protocol.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is generating a large volume of outbound traffic from an unusual protocol type that is not typically used by EC2 instances, such as Internet Group Management Protocol. This may indicate that the instance is compromised and is being used to perform denial-of-service (DoS) attacks using an unusual protocol. This finding detects DoS attacks only against publicly routable IP addresses, which are primary targets of DoS attacks.

8.1.8 Backdoor:EC2/Spambot

An EC2 instance is exhibiting unusual behaviour by communicating with a remote host on port 25.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment is communicating with a remote host on port 25. This behaviour is unusual because this EC2 instance has no prior history of communications on port 25. Port 25 is traditionally used by mail servers for SMTP communications. This finding indicates your EC2 instance might be compromised for use in sending out spam.

8.1.9 Behavior:EC2/NetworkPortUnusual

An EC2 instance is communicating with a remote host on an unusual server port.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment is behaving in a way that deviates from the established baseline. This EC2 instance has no prior history of communications on this remote port.

8.1.10 Behavior:EC2/TrafficVolumeUnusual

An EC2 instance is generating unusually large amounts of network traffic to a remote host.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment is behaving in a way that deviates from the established baseline. This EC2 instance has no prior history of sending this much traffic to this remote host.

8.1.11 CryptoCurrency:EC2/BitcoinTool.B

An EC2 instance is querying an IP address that is associated with cryptocurrency-related activity.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is querying an IP Address that is associated with Bitcoin or other cryptocurrency-related activity. Bitcoin is a worldwide cryptocurrency and digital payment system. Besides being used as a reward for Bitcoin mining, Bitcoin can be exchanged for other currencies, products, and services.

8.1.12 CryptoCurrency:EC2/BitcoinTool.B!DNS

An EC2 instance is querying a domain name that is associated with cryptocurrency-related activity.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is querying a domain name that is associated with Bitcoin or other cryptocurrency-related activity. Bitcoin is a worldwide cryptocurrency and digital payment system. Besides being used as a reward for Bitcoin mining, Bitcoin can be exchanged for other currencies, products, and services.

8.1.13 Impact:EC2/AbusedDomainRequest.Reputation

An EC2 instance is querying a low reputation domain name that is associated with known abused domains.

Default severity: Medium

This finding informs you that the listed Amazon EC2 instance within your AWS environment is querying a low reputation domain name associated with known abused domains or IP addresses. Examples of abused domains are top level domain names (TLDs) and second-level domain names (2LDs) providing free subdomain registrations as well as dynamic DNS providers. Threat actors tend to use these services to register domains for free or at low costs. Low reputation domains in this category may also be expired domains resolving to a registrar's parking IP address and therefore may no longer be active. A parking IP is where a registrar directs traffic for domains that have not been linked to any service. The listed Amazon EC2 instance may be compromised as threat actors commonly use these registrar's or services for C&C and malware distribution.

Low reputation domains are based on a reputation score model developed by GuardDuty, which evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

8.1.14 Impact:EC2/BitcoinDomainRequest.Reputation

An EC2 instance is querying a low reputation domain name that is associated with cryptocurrency-related activity.

Default severity: High

This finding informs you that the listed Amazon EC2 instance within your AWS environment is querying a low reputation domain name associated with Bitcoin or other cryptocurrency-related activity. Bitcoin is a worldwide cryptocurrency and digital payment system. Bitcoin is a reward for bitcoin mining that can be exchanged for other currencies, products, and services, and is highly sought after by threat actors.

Low reputation domains are based on a reputation score model developed by GuardDuty, which evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

8.1.15 Impact:EC2/MaliciousDomainRequest.Reputation

An EC2 instance is querying a low reputation domain that is associated with known malicious domains.

Default severity: High

This finding informs you that the listed Amazon EC2 instance within your AWS environment is querying a low reputation domain name associated with known malicious domains or IP addresses. For example, domains may be associated with a known sinkhole IP address. Sinkholed domains are domains that were previously controlled by a threat actor, and requests made to them can indicate the instance is compromised. These domains may also be correlated with known malicious campaigns or domain generation algorithms.

Low reputation domains are based on a reputation score model developed by GuardDuty, which evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

8.1.16 Impact:EC2/PortSweep

An EC2 instance is probing a port on a large number of IP addresses.

Default severity: High

This finding informs you the listed EC2 instance in your AWS environment is probing a port on a large number of publicly routable IP addresses. This type of activity is typically used to find vulnerable hosts to exploit.

8.1.17 Impact:EC2/SuspiciousDomainRequest.Reputation

An EC2 instance is querying a low reputation domain name that is suspicious in nature due to its age, or low popularity.

Default severity: Low

This finding informs you that the listed Amazon EC2 instance within your AWS environment is querying a low reputation domain name that is suspected of being malicious. GuardDuty noticed characteristics of this domain that were consistent with previously observed malicious domains, however, our reputation model was unable to definitively relate it to a known threat. These domains are typically newly observed or receive a low amount of traffic.

Low reputation domains are based on a reputation score model developed by GuardDuty, which evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

8.1.18 Impact:EC2/WinRMBruteForce

An EC2 instance is performing an outbound Windows Remote Management brute force attack.

Default severity: Low*

This finding's severity is low if your EC2 instance was the target of a brute force attack. This finding's severity is High if your EC2 instance is the actor being used to perform the brute force attack.

This finding informs you that the listed EC2 instance in your AWS environment is performing a Windows Remote Management (WinRM) brute force attack aimed at gaining access to the Windows Remote Management service on Windows-based systems.

8.1.19 Recon:EC2/PortProbeEMRUnprotectedPort

An EC2 instance has an unprotected EMR related port which is being probed by a known malicious host.

Default severity: High

This finding informs you that an EMR related sensitive port on the listed EC2 instance that is part of an cluster in your AWS environment is not blocked by a security group, an access control list (ACL), or an on-host firewall such as Linux IPTables, and that known scanners on the internet are actively probing it. Ports that can trigger this finding, such as port 8088 (YARN Web UI port), could potentially be used for remote code execution.

8.1.20 Recon:EC2/PortProbeUnprotectedPort

An EC2 instance has an unprotected port that is being probed by a known malicious host.

Default severity: Low*

This finding's default severity is Low. However, if the port being probed is used by (9200 or 9300), the finding's severity is High.

This finding informs you that a port on the listed EC2 instance in your AWS environment is not blocked by a security group, access control list (ACL), or an on-host firewall such as Linux IPTables, and that known scanners on the internet are actively probing it.

If the identified unprotected port is 22 or 3389 and you are using these ports to connect to your instance, you can still limit exposure by allowing access to these ports only to the IP addresses from your corporate network IP address space. To restrict access to port 22 on Linux, see Authorising Inbound Traffic for Your Linux Instances. To restrict access to port 3389 on Windows, see Authorising Inbound Traffic for Your Windows Instances.

8.1.21 Recon:EC2/Portscan

An EC2 instance is performing outbound port scans to a remote host.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment is engaged in a possible port scan attack because it is trying to connect to multiple ports over a short period of time. The purpose of a port scan attack is to locate open ports to discover which services the machine is running and to identify its operating system.

8.1.22 Trojan:EC2/BlackholeTraffic

An EC2 instance is attempting to communicate with an IP address of a remote host that is a known black hole.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment might be compromised because it is trying to communicate with an IP address of a black hole (or sinkhole). Black holes are places in the network where incoming or outgoing traffic is silently discarded without informing the source that the data didn't reach its intended recipient. A black hole IP address specifies a host machine that is not running or an address to which no host has been assigned.

8.1.23 Trojan:EC2/BlackholeTraffic!DNS

An EC2 instance is querying a domain name that is being redirected to a black hole IP address.

Default severity: Medium

This finding informs you the listed EC2 instance in your AWS environment might be compromised because it is querying a domain name that is being redirected to a black hole IP address. Black holes are places in the network where incoming or outgoing traffic is silently discarded without informing the source that the data didn't reach its intended recipient.

8.1.24 Trojan:EC2/DGADomainRequest.B

An EC2 instance is querying algorithmically generated domains. Such domains are commonly used by malware and could be an indication of a compromised EC2 instance.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is trying to query domain generation algorithm (DGA) domains. Your EC2 instance might be compromised.

DGAs are used to periodically generate a large number of domain names that can be used as rendezvous points with their command and control (C&C) servers. Command and control servers are computers that issue commands to members of a botnet, which is a collection of internet-connected devices that are infected and controlled by a common type of malware. The large number of potential rendezvous points makes it difficult to effectively shut down botnets because infected computers attempt to contact some of these domain names every day to receive updates or commands.

This finding is based on analysis of domain names using advanced heuristics and may identify new DGA domains that are not present in threat intelligence feeds.

8.1.25 Trojan:EC2/DGADomainRequest.C!DNS

An EC2 instance is querying algorithmically generated domains. Such domains are commonly used by malware and could be an indication of a compromised EC2 instance.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is trying to query domain generation algorithm (DGA) domains. Your EC2 instance might be compromised.

DGAs are used to periodically generate a large number of domain names that can be used as rendezvous points with their command and control (C&C) servers. Command and control servers are computers that issue commands to members of a botnet, which is a collection of internet-connected devices that are infected and controlled by a common type of malware. The large number of potential rendezvous points makes it difficult to effectively shut down botnets because infected computers attempt to contact some of these domain names every day to receive updates or commands.

This finding is based on known DGA domains from GuardDuty's threat intelligence feeds.

8.1.26 Trojan:EC2/DNSDataExfiltration

An EC2 instance is exfiltrating data through DNS queries.

Default severity: High

This finding informs you that the listed EC2 instance in your AWS environment is running malware that uses DNS queries for outbound data transfers. This type of data transfer is indicative of a compromised instance and could result in the exfiltration of data. DNS traffic is not typically blocked by firewalls. For example, malware in a compromised EC2 instance can encode data, (such as your credit card number), into a DNS query and send it to a remote DNS server that is controlled by an attacker.

8.1.27 Trojan:EC2/DriveBySourceTraffic!DNS

An EC2 instance is querying a domain name of a remote host that is a known source of Drive-By download attacks.

Default severity: Medium

This finding informs you that the listed EC2 instance in your AWS environment might be compromised because it is querying a domain name of a remote host that is a known source of drive-by download attacks. These are unintended downloads of computer software from the internet that can trigger an automatic installation of a virus, spyware, or malware.

8.1.28 Trojan:EC2/DropPoint

An EC2 instance is attempting to communicate with an IP address of a remote host that is known to hold credentials and other stolen data captured by malware.

Default severity: Medium

This finding informs you that an EC2 instance in your AWS environment is trying to communicate with an IP address of a remote host that is known to hold credentials and other stolen data captured by malware.

8.1.29 Trojan:EC2/DropPoint!DNS

An EC2 instance is querying a domain name of a remote host that is known to hold credentials and other stolen data captured by malware.

Default severity: Medium

This finding informs you that an EC2 instance in your AWS environment is querying a domain name of a remote host that is known to hold credentials and other stolen data captured by malware.

8.1.30 Trojan:EC2/PhishingDomainRequest!DNS

An EC2 instance is querying domains involved in phishing attacks. Your EC2 instance might be compromised.

Default severity: High

This finding informs you that there is an EC2 instance in your AWS environment that is trying to query a domain involved in phishing attacks. Phishing domains are set up by someone posing as a legitimate institution in order to induce individuals to provide sensitive data, such as personally identifiable information, banking and credit card details, and passwords. Your EC2 instance may be trying to retrieve sensitive data stored on a phishing website, or it may be attempting to set up a phishing website. Your EC2 instance might be compromised.

8.1.31 UnauthorizedAccess:EC2/MaliciousIPCaller.Custom

An EC2 instance is making connections to an IP address on a custom threat list.

Default severity: Medium

This finding informs you that an EC2 instance in your AWS environment is communicating with an IP address included on a threat list that you uploaded. In GuardDuty, a threat list consists of known malicious IP addresses. GuardDuty generates findings based on uploaded threat lists. The threat list used to generate this finding will be listed in the finding's details.

8.1.32 UnauthorizedAccess:EC2/MetadataDNSRebind

An EC2 instance is performing DNS lookups that resolve to the instance metadata service.

Default severity: High

This finding informs you that an EC2 instance in your AWS environment is querying a domain that resolves to the EC2 metadata IP address (169.254.169.254). A DNS query of this kind may indicate that the instance is a target of a DNS rebinding technique. This technique can be used to obtain metadata from an EC2 instance, including the IAM credentials associated with the instance.

DNS rebinding involves tricking an application running on the EC2 instance to load return data from a URL, where the domain name in the URL resolves to the EC2 metadata IP address (169.254.169.254). This causes the application to access EC2 metadata and possibly make it available to the attacker.

It is possible to access EC2 metadata using DNS rebinding only if the EC2 instance is running a vulnerable application that allows injection of URLs, or if someone accesses the URL in a web browser running on the EC2 instance.

8.1.33 UnauthorizedAccess:EC2/RDPBruteForce

An EC2 instance has been involved in RDP brute force attacks.

Default severity: Low*

This finding's severity is low if your EC2 instance was the target of a brute force attack. This finding's severity is high if your EC2 instance is the actor being used to perform the brute force attack.

This finding informs you that an EC2 instance in your AWS environment was involved in a brute force attack aimed at obtaining passwords to RDP services on Windows-based systems. This can indicate unauthorised access to your AWS resources.

8.1.34 UnauthorizedAccess:EC2/SSHBruteForce

An EC2 instance has been involved in SSH brute force attacks.

Default severity: Low*

This finding's severity is low if a brute force attack is aimed at one of your EC2 instances. This finding's severity is High if your EC2 instance is being used to perform the brute force attack.

This finding informs you that an EC2 instance in your AWS environment was involved in a brute force attack aimed at obtaining passwords to SSH services on Linux-based systems. This can indicate unauthorised access to your AWS resources.

This finding is generated only through monitoring traffic on port 22. If your SSH services are configured to use other ports, this finding is not generated.

8.1.35 UnauthorizedAccess:EC2/TorClient

Your EC2 instance is making connections to a Tor Guard or an Authority node.

Default severity: High

This finding informs you that an EC2 instance in your AWS environment is making connections to a Tor Guard or an Authority node. Tor is software for enabling anonymous communication. Tor Guards and Authority nodes act as initial gateways into a Tor network. This traffic can indicate that this EC2 instance has been compromised and is acting as a client on a Tor network. This finding may indicate unauthorised access to your AWS resources with the intent of hiding the attacker's true identity.

8.1.36 UnauthorizedAccess:EC2/TorRelay

Your EC2 instance is making connections to a Tor network as a Tor relay.

Default severity: High

This finding informs you that an EC2 instance in your AWS environment is making connections to a Tor network in a manner that suggests that it's acting as a Tor relay. Tor is software for enabling anonymous communication. Tor increases anonymity of communication by forwarding the client's possibly illicit traffic from one Tor relay to another.

8.2 S3 Finding Types

The following findings are specific to S3 bucket resources and will always have a Resource Type of S3Bucket. The severity and details of the findings will differ based on the finding type and the permission associated with the bucket.

For all S3 Bucket type findings it is recommended that you examine the permissions on the bucket in question and the permissions of any users involved in the finding, if the activity is unexpected see the remediation recommendations detailed in Remediating a Compromised S3 Bucket.

- Discovery:S3/BucketEnumeration.Unusual
- Discovery:S3/MaliciousIPCaller
- Discovery:S3/MaliciousIPCaller.Custom
- Discovery:S3/TorIPCaller
- Exfiltration:S3/MaliciousIPCaller
- Exfiltration:S3/ObjectRead.Unusual
- Impact:S3/MaliciousIPCaller
- Impact:S3/ObjectDelete.Unusual
- Impact:S3/PermissionsModification.Unusual
- PenTest:S3/KaliLinux
- PenTest:S3/ParrotLinux
- PenTest:S3/PentooLinux
- Policy:S3/AccountBlockPublicAccessDisabled
- Policy:S3/BucketAnonymousAccessGranted
- Policy:S3/BucketBlockPublicAccessDisabled

- o Policy:S3/BucketPublicAccessGranted
- o Stealth:S3/ServerAccessLoggingDisabled
- o UnauthorizedAccess:S3/MaliciousIPCaller.Custom
- o UnauthorizedAccess:S3/TorIPCaller

8.2.1 Discovery:S3/BucketEnumeration.Unusual

An IAM entity invoked an S3 API used to discover S3 buckets within your network.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding informs you that an IAM entity has invoked an S3 API to discover S3 buckets in your environment, such as ListBuckets. This type of activity is associated with the discovery stage of an attack wherein an attacker is gathering information to determine if your AWS environment is susceptible to a broader attack. This activity is suspicious because the way the IAM entity invoked the API was unusual. For example, this IAM entity had no prior history of invoking this type of API, or the API was invoked from an unusual location.

8.2.2 Discovery:S3/MaliciousIPCaller

An S3 API commonly used to discover resources in an AWS environment was invoked from a known malicious IP address.

Default severity: High

This finding informs you that an S3 API operation was invoked from an IP address that is associated with known malicious activity. The observed API is commonly associated with the discovery stage of an attack when an adversary is gathering information on your AWS environment. Examples include GetObjectAcl or ListObjects.

8.2.3 Discovery:S3/MaliciousIPCaller.Custom

An S3 API was invoked from an IP address on a custom threat list.

Default severity: High

This finding informs you that an S3 API, such as GetObjectAcl or ListObjects, was invoked from an IP address that is included on a threat list that you uploaded. The threat list associated with this finding is listed in the Additional information section of a finding's details. This type of activity is associated with the discovery stage of an attack wherein an attacker is gathering information to determine if your AWS environment is susceptible to a broader attack.

8.2.4 Discovery:S3/TorIPCaller

An S3 API was invoked from a Tor exit node IP address.

Default severity: Medium

This finding informs you that an S3 API, such as GetObjectAcl or ListObjects, was invoked from a Tor exit node IP address. This type of activity is associated with the discovery stage of an attack wherein an attacker is gathering information to determine if your AWS environment is susceptible to a broader attack. Tor is software for enabling anonymous communication. It encrypts and randomly bounces communications through relays between a series of network nodes. The last Tor node is called the exit node. This can indicate unauthorised access to your AWS resources with the intent of hiding the attacker's true identity.

8.2.5 Exfiltration:S3/MaliciousIPCaller

An S3 API commonly used to collect data from an AWS environment was invoked from a known malicious IP address.

Default severity: High

This finding informs you that an S3 API operation was invoked from an IP address that is associated with known malicious activity. The observed API is commonly associated with exfiltration tactics where an adversary is trying to collect data from your network. Examples include GetObject and CopyObject.

8.2.6 Exfiltration:S3/ObjectRead.Unusual

An IAM entity invoked an S3 API in a suspicious way.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding informs you that an IAM entity in your AWS environment is making API calls that involve an S3 bucket and that differ from that entity's established baseline. The API call used in this activity is associated with the exfiltration stage of an attack, wherein an attacker is attempting to collect data. This activity is suspicious because the way the IAM entity invoked the API was unusual. For example, this IAM entity had no prior history of invoking this type of API, or the API was invoked from an unusual location.

8.2.7 Impact:S3/MaliciousIPCaller

An S3 API commonly used to tamper with data or processes in an AWS environment was invoked from a known malicious IP address.

Default severity: High

This finding informs you that an S3 API operation was invoked from an IP address that is associated with known malicious activity. The observed API is commonly associated with impact tactics where an adversary is trying to manipulate, interrupt, or destroy data within your AWS environment. Examples include, PutObject or PutObjectAcl.

8.2.8 Impact:S3/ObjectDelete.Unusual

An IAM entity invoked an API used to delete data in an S3 bucket.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding informs you that a specific IAM entity in your AWS environment is making API calls designed to delete data in the listed S3 bucket by deleting the bucket itself. This activity is suspicious because the way the IAM entity invoked the API was unusual. For example, this IAM entity had no prior history of invoking this type of API, or the API was invoked from an unusual location.

8.2.9 Impact:S3/PermissionsModification.Unusual

An IAM entity invoked an API to modify permissions on one or more S3 resources.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding informs you that an IAM entity is making API calls designed to modify the permissions on one or more buckets or objects in your AWS environment. This action may be performed by an attacker to allow information to be shared outside of the account. This activity is suspicious because the way the IAM entity invoked the API was unusual. For example, this IAM entity had no prior history of invoking this type of API, or the API was invoked from an unusual location.

8.2.10 PenTest:S3/KaliLinux

An S3 API was invoked from a Kali Linux machine.

Default severity: Medium

This finding informs you that a machine running Kali Linux is making S3 API calls using credentials that belong to your AWS account. Your credentials might be compromised. Kali Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.2.11 PenTest:S3/ParrotLinux

An S3 API was invoked from a Parrot Security Linux machine.

Default severity: Medium

This finding informs you that a machine running Parrot Security Linux is making S3 API calls using credentials that belong to your AWS account. Your credentials might be compromised. Parrot Security Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.2.12 PenTest:S3/PentooLinux

An S3 API was invoked from a Pentoo Linux machine

Default severity: Medium

This finding informs you that a machine running Pentoo Linux is making S3 API calls using credentials that belong to your AWS account. Your credentials might be compromised. Pentoo Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.2.13 Policy:S3/AccountBlockPublicAccessDisabled

An IAM entity invoked an API used to disable S3 block public access on an account.

Default severity: Low

This finding informs you that Amazon S3 Block Public Access was disabled at the account level. When S3 Block Public Access settings are enabled, they are used to filter the policies or access control lists (ACLs) on buckets as a security measure to prevent inadvertent public exposure of data.

Typically, S3 Block Public Access is turned off in an account to allow public access to a bucket or to the objects in the bucket. When S3 Block Public Access is disabled for an account, access to your buckets is controlled by the policies, ACLs, or bucket-level Block Public Access settings applied to your individual buckets. This does not necessarily mean that the buckets are shared publicly, but that you should audit the permissions applied to the buckets to confirm that they provide the appropriate level of access.

8.2.14 Policy:S3/BucketAnonymousAccessGranted

An IAM principal has granted access to an S3 bucket to the internet by changing bucket policies or ACLs.

Default severity: High

This finding informs you that the listed S3 bucket has been made publicly accessible on the internet because an IAM entity has changed a bucket policy or ACL on that bucket. After a policy or ACL change is detected, GuardDuty uses automated reasoning powered by Zelkova to determine if the bucket is publicly accessible.

If a bucket's ACLs or bucket policies are configured to explicitly deny or to deny all, this finding cannot be generated for that bucket.

8.2.15 Policy:S3/BucketBlockPublicAccessDisabled

An IAM entity invoked an API used to disable S3 block public access on a bucket.

Default severity: Low

This finding informs you that Block Public Access was disabled for the listed S3 bucket. When enabled, S3 Block Public Access settings are used to filter the policies or access control lists (ACLs) applied to buckets as a security measure to prevent inadvertent public exposure of data.

Typically, S3 Block Public Access is turned off on a bucket to allow public access to the bucket or to the objects within. When S3 Block Public Access is disabled for a bucket, access to the bucket is controlled by the policies or ACLs applied to it. This does not mean that the bucket is shared publicly, but you should audit the policies and ACLs applied to the bucket to confirm that appropriate permissions are applied.

8.2.16 Policy:S3/BucketPublicAccessGranted

An IAM principal has granted public access to an S3 bucket to all AWS users by changing bucket policies or ACLs.

Default severity: High

This finding informs you that the listed S3 bucket has been publicly exposed to all authenticated AWS users because an IAM entity has changed a bucket policy or ACL on that S3 bucket. After a policy or ACL change is detected, GuardDuty uses automated reasoning powered by Zelkova to determine if the bucket is publicly accessible.

If a bucket's ACLs or bucket policies are configured to explicitly deny or to deny all, this finding cannot be generated for that bucket.

8.2.17 Stealth:S3/ServerAccessLoggingDisabled

S3 server access logging was disabled for a bucket.

Default severity: Low

This finding informs you that S3 server access logging is disabled for a bucket within your AWS environment. If disabled, no logs are created for any actions taken on the identified S3 bucket or on the objects in the bucket, unless S3 object level logging is enabled for this bucket. Disabling logging is a technique used by unauthorised users in order to cover their tracks. This finding is triggered when server access logging is disabled for a bucket. To learn more, see S3 Server Access Logging.

8.2.18 UnauthorizedAccess:S3/MaliciousIPCaller.Custom

An S3 API was invoked from an IP address on a custom threat list.

Default severity: High

This finding informs you that an S3 API operation, for example, PutObject or PutObjectAcl, was invoked from an IP address that is included on a threat list that you uploaded. The threat list associated with this finding is listed in the Additional information section of a finding's details.

8.2.19 UnauthorizedAccess:S3/TorIPCaller

An S3 API was invoked from a Tor exit node IP address.

Default severity: High

This finding informs you that an S3 API operation, such as PutObject or PutObjectAcl, was invoked from a Tor exit node IP address. Tor is software for enabling anonymous communication. It encrypts and randomly bounces communications through relays between a series of network nodes. The last Tor node is called the exit node. This finding can indicate unauthorised access to your AWS resources with the intent of hiding the attacker's true identity.

8.3 IAM Finding Types

The following findings are specific to IAM entities and access keys and always have a Resource Type of AccessKey. The severity and details of the findings differ based on the finding type.

For all IAM related findings, it is recommended that you examine the permissions on the entity in question and ensure that this entity follows the best practice of least privilege. If the activity is unexpected, the credentials may be compromised.

- PenTest:IAMUser/KaliLinux
- PenTest:IAMUser/ParrotLinux
- PenTest:IAMUser/PentooLinux
- Persistence:IAMUser/NetworkPermissions
- Persistence:IAMUser/ResourcePermissions
- Persistence:IAMUser/UserPermissions
- Policy:IAMUser/RootCredentialUsage
- PrivilegeEscalation:IAMUser/AdministrativePermissions
- Recon:IAMUser/MaliciousIPCaller
- Recon:IAMUser/MaliciousIPCaller.Custom
- Recon:IAMUser/NetworkPermissions
- Recon:IAMUser/ResourcePermissions
- Recon:IAMUser/TorIPCaller
- Recon:IAMUser/UserPermissions
- ResourceConsumption:IAMUser/ComputeResources
- Stealth:IAMUser/CloudTrailLoggingDisabled
- Stealth:IAMUser/LoggingConfigurationModified
- Stealth:IAMUser/PasswordPolicyChange
- UnauthorizedAccess:IAMUser/ConsoleLogin
- UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B
- UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration
- UnauthorizedAccess:IAMUser/MaliciousIPCaller
- UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom
- UnauthorizedAccess:IAMUser/TorIPCaller

8.3.1 PenTest:IAMUser/KaliLinux

An API was invoked from a Kali Linux EC2 machine.

Default severity: Medium

This finding informs you that a machine running Kali Linux is making API calls using credentials that belong to the listed AWS account in your environment. Kali Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.3.2 PenTest:IAMUser/ParrotLinux

An API was invoked from a Parrot Security Linux machine.

Default severity: Medium

This finding informs you that a machine running Parrot Security Linux is making API calls using credentials that belong to the listed AWS account in your environment. Parrot Security Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.3.3 PenTest:IAMUser/PentooLinux

An API was invoked from a Pentoo Linux machine.

Default severity: Medium

This finding informs you that a machine running Pentoo Linux is making API calls using credentials that belong to the listed AWS account in your environment. Pentoo Linux is a popular penetration testing tool that security professionals use to identify weaknesses in EC2 instances that require patching. Attackers also use this tool to find EC2 configuration weaknesses and gain unauthorised access to your AWS environment.

8.3.4 Persistence:IAMUser/NetworkPermissions

An IAM entity invoked an API commonly used to change the network access permissions for security groups, routes, and ACLs in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding indicates that a specific principal (AWS account root user, IAM role, or IAM user) in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API.

This finding is triggered when network configuration settings are changed under suspicious circumstances, such as when a principal invokes the CreateSecurityGroup API with no prior history of doing so. Attackers often attempt to change security groups to allow certain inbound traffic on various ports to improve their ability to access an EC2 instance.

8.3.5 Persistence:IAMUser/ResourcePermissions

A principal invoked an API commonly used to change the security access policies of various resources in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding indicates that a specific principal (AWS account root user, IAM role, or IAM user) in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API.

This finding is triggered when a change is detected to policies or permissions attached to AWS resources, such as when a principal in your AWS environment invokes the PutBucketPolicy API with no prior history of doing so. Some services, such as Amazon S3, support resource-attached permissions that grant one or more principals access to the resource. With stolen credentials, attackers can change the policies attached to a resource in order to gain access to that resource.

8.3.6 Persistence:IAMUser/UserPermissions

A principal invoked an API commonly used to add, modify, or delete IAM users, groups or policies in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding indicates that a specific principal (AWS account root user, IAM role, or IAM user) in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API.

This finding is triggered by suspicious changes to the user-related permissions in your AWS environment, such as when a principal in your AWS environment invokes the AttachUserPolicy API with no prior history of doing so. Attackers may use stolen credentials to create new users, add access policies to existing users, or create access keys to maximise their access to an account, even if their original access point is closed. For example, the owner of the account might notice that a particular IAM user or password was stolen and delete it from the account. However, they might not delete other users that were created by a fraudulently created admin principal, leaving their AWS account accessible to the attacker.

8.3.7 Policy:IAMUser/RootCredentialUsage

An API was invoked using root credentials.

Default severity: Low

This finding informs you that the root credentials of the listed AWS account in your environment are being used to make requests to AWS services. It is recommended that users never use root credentials to access AWS services. Instead, AWS services should be accessed using least-privilege temporary credentials from AWS Security Token Service (STS). For situations where STS is not supported, IAM user credentials are recommended. For more information, see IAM Best Practices

If S3 threat detection is enabled for the account this finding may be generated in response to attempts to run S3 data plane operations on S3 resources using the root credentials of the AWS account. The API call used will be listed in the finding details. If S3 threat detection is not enabled this finding can only be triggered by Event log APIs. For more information on S3 threat detection see Amazon S3 protection in Amazon GuardDuty.

8.3.8 PrivilegeEscalation:IAMUser/AdministrativePermissions

A principal has attempted to assign a highly permissive policy to themselves.

Default severity: Low*

This finding's severity is Low if the attempt at privilege escalation was unsuccessful, and Medium if the attempt at privilege escalation was successful.

This finding indicates that a specific IAM entity in your AWS environment is exhibiting behaviour that can be indicative of a privilege escalation attack. This finding is triggered when an IAM user or role attempts to assign a highly permissive policy to themselves. If the user or role in question is not meant to have administrative privileges, either the user's credentials may be compromised or the role's permissions may not be configured properly.

Attackers will use stolen credentials to create new users, add access policies to existing users, or create access keys to maximise their access to an account even if their original access point is closed. For example, the owner of the account might notice that a particular IAM user or password was stolen and delete it from the account, but might not delete other users that were created by a fraudulently created admin principal, leaving their AWS account still accessible to the attacker.

8.3.9 Recon:IAMUser/MaliciousIPCaller

An API was invoked from a known malicious IP address.

Default severity: Medium

This finding informs you that an API operation that can list or describe AWS resources in an account within your environment was invoked from an IP address that is included on a GuardDuty threat list. An attacker may use stolen credentials to perform this type of reconnaissance of your AWS resources in order to find more valuable credentials or determine the capabilities of the credentials they already have.

8.3.10 Recon:IAMUser/MaliciousIPCaller.Custom

An API was invoked from a known malicious IP address.

Default severity: Medium

This finding informs you that an API operation that can list or describe AWS resources in an account within your environment was invoked from an IP address that is included on a custom threat list. The threat list used will be listed in the finding's details. An attacker might use stolen credentials to perform this type of reconnaissance of your AWS resources in order to find more valuable credentials or determine the capabilities of the credentials they already have.

8.3.11 Recon:IAMUser/NetworkPermissions

A principal invoked an API commonly used to change the network access permissions for security groups, routes, and ACLs in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding indicates that a specific principal (AWS account root user, IAM role, or IAM user) in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API.

This finding is triggered when resource access permissions in your AWS account are probed under suspicious circumstances. For example, if a principal invoked the DescribeInstances API with no prior history of doing so. An attacker might use stolen credentials to perform this type of reconnaissance of your AWS resources in order to find more valuable credentials or determine the capabilities of the credentials they already have.

8.3.12 Recon:IAMUser/ResourcePermissions

A principal invoked an API commonly used to change the security access policies of various resources in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding indicates that a specific principal (AWS account root user, IAM role, or IAM user) in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API.

This finding is triggered when resource access permissions in your AWS account are probed under suspicious circumstances. For example, if a principal invoked the DescribeInstances API with no prior history of doing so. An attacker might use stolen credentials to perform this type of reconnaissance of your AWS resources in order to find more valuable credentials or determine the capabilities of the credentials they already have.

8.3.13 Recon:IAMUser/TorIPCaller

An API was invoked from a Tor exit node IP address.

Default severity: Medium

This finding informs you that an API operation that can list or describe AWS resources in an account within your environment was invoked from a Tor exit node IP address. Tor is software for enabling anonymous communication. It encrypts and randomly bounces communications through relays between a series of network nodes. The last Tor node is called the exit node. An attacker would use Tor to mask their true identity.

8.3.14 Recon:IAMUser/UserPermissions

A principal invoked an API commonly used to add, modify, or delete IAM users, groups or policies in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding is triggered when user permissions in your AWS environment are probed under suspicious circumstances. For example, if a principal (AWS account root user, IAM role, or IAM user) invoked the ListInstanceProfilesForRole API with no prior history of doing so. An attacker might use stolen credentials to perform this type of reconnaissance of your AWS resources in order to find more valuable credentials or determine the capabilities of the credentials they already have.

This finding indicates that a specific principal in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of invoking this API in this way.

8.3.15 ResourceConsumption:IAMUser/ComputeResources

A principal invoked an API commonly used to launch Compute resources like EC2 Instances.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding is triggered when EC2 instances in the listed account within your AWS environment are launched under suspicious circumstances. This finding indicates that a specific principal in your AWS environment is exhibiting behaviour that is different from the established baseline; for example, if a principal (AWS account root user, IAM role, or IAM user) invoked the RunInstances API with no prior history of doing so. This might be an indication of an attacker using stolen credentials to steal compute time (possibly for cryptocurrency mining or password cracking). It can also be an indication of an attacker using an EC2 instance in your AWS environment and its credentials to maintain access to your account.

8.3.16 Stealth:IAMUser/CloudTrailLoggingDisabled

AWS CloudTrail trail was disabled.

Default severity: Low

This finding informs you that a CloudTrail trail within your AWS environment was disabled. This can be an attacker's attempt to disable logging to cover their tracks by eliminating any trace of their activity while gaining access to your AWS resources for malicious purposes. This finding can be triggered by a successful deletion or update of a trail. This finding can also be triggered by a successful deletion of an S3 bucket that stores the logs from a trail that is associated with GuardDuty.

8.3.17 Stealth:IAMUser/LoggingConfigurationModified

A principal invoked an API commonly used to stop CloudTrail Logging, delete existing logs, and otherwise eliminate traces of activity in your AWS account.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding is triggered when the logging configuration in the listed AWS account within your environment is modified under suspicious circumstances. This finding informs you that a specific principal in your AWS environment is exhibiting behaviour that is different from the established baseline; for example, if a principal (AWS account root user, IAM role, or IAM user) invoked the StopLogging API with no prior history of doing so. This can be an indication of an attacker trying to cover their tracks by eliminating any trace of their activity.

8.3.18 Stealth:IAMUser/PasswordPolicyChange

Account password policy was weakened.

Default severity: Low

The AWS account password policy was weakened on the listed account within your AWS environment. For example, it was deleted or updated to require fewer characters, not require symbols and numbers, or required to extend the password expiration period. This finding can also be triggered by an attempt to update or delete your AWS account password policy. The AWS account password policy defines the rules that govern what kinds of passwords can be set for your IAM users. A weaker password policy permits the creation of passwords that are easy to remember and potentially easier to guess, thereby creating a security risk.

8.3.19 UnauthorizedAccess:IAMUser/ConsoleLogin

An unusual console login by a principal in your AWS account was observed.

Default severity: Medium*

This finding's default severity is Medium. However, if the API is invoked using temporary AWS credentials that are created on an instance, the finding's severity is High.

This finding is triggered when a console login is detected under suspicious circumstances. For example, if a principal with no prior history of doing so, invoked the ConsoleLogin API from a never-before-used client or an unusual location. This could be an indication of stolen credentials being used to gain access to your AWS account, or a valid user accessing the account in an invalid or less secure manner (for example, not over an approved VPN).

This finding informs you that a specific principal in your AWS environment is exhibiting behaviour that is different from the established baseline. This principal has no prior history of login activity using this client application from this specific location.

8.3.20 UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B

Multiple worldwide successful console logins were observed.

Default severity: Medium

This finding informs you that multiple successful console logins for the same IAM user were observed around the same time in various geographical locations. Such anomalous and risky access location patterns indicate potential unauthorised access to your AWS resources.

8.3.21 UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration

Credentials that were created exclusively for an EC2 instance through an Instance launch role are being used from an external IP address.

Default severity: High

This finding informs you of attempts to run AWS API operations from a host outside of EC2, using temporary AWS credentials that were created on an EC2 instance in your AWS environment. The listed EC2 instance might be compromised, and the temporary credentials from this instance might have been exfiltrated to a remote host outside of AWS. AWS does not recommend redistributing temporary credentials outside of the entity that created them (for example, AWS applications, EC2, or Lambda). However, authorised users can export credentials from their EC2 instances to make legitimate API calls. To rule out a potential attack and verify the legitimacy of the activity, contact the IAM user to whom these credentials are assigned.

If S3 threat detection is enabled for the account this finding may be generated in response to attempts to run S3 data plane operations on the S3 resources using EC2 credentials. The API call used will be listed in the finding details. If S3 threat detection is not enabled this finding can only be triggered by Event log APIs. For more information on S3 threat detection see Amazon S3 protection in Amazon GuardDuty.

8.3.22 UnauthorizedAccess:IAMUser/MaliciousIPCaller

An API was invoked from a known malicious IP address.

Default severity: Medium

This finding informs you that an API operation (for example, an attempt to launch an EC2 instance, create a new IAM user, modify your AWS privileges) was invoked from a known malicious IP address. This can indicate unauthorised access to AWS resources within your environment.

8.3.23 UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom

An API was invoked from an IP address on a custom threat list.

Default severity: Medium

This finding informs you that an API operation (for example, an attempt to launch an EC2 instance, create a new IAM user, modify AWS privileges) was invoked from an IP address that is included on a threat list that you uploaded. In , a threat list consists of known malicious IP addresses. generates findings based on uploaded threat lists. This can indicate unauthorised access to your AWS resources within your environment.

8.3.24 UnauthorizedAccess:IAMUser/TorIPCaller

An API was invoked from a Tor exit node IP address.

Default severity: Medium

This finding informs you that an API operation (for example, an attempt to launch an EC2 instance, create a new IAM user, or modify your AWS privileges) was invoked from a Tor exit node IP address. Tor is software for enabling anonymous communication. It encrypts and randomly bounces communications through relays between a series of network nodes. The last Tor node is called the exit node. This can indicate unauthorised access to your AWS resources with the intent of hiding the attacker's true identity.