Amazon GuardDuty

Amazon GuardDuty User Guide

Amazon GuardDuty: Amazon GuardDuty User Guide

Table of Contents

What is Amazon GuardDuty?

Amazon GuardDuty is a continuous security monitoring service that analyzes and processes the following Data sources (p.): VPC Flow Logs, AWS CloudTrail management event logs, CloudTrail S3 data event logs, EKS audit logs, and DNS logs. It uses threat intelligence feeds, such as lists of malicious IP addresses and domains, and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment. This can include issues like escalations of privileges, uses of exposed credentials, or communication with malicious IP addresses, or domains. For example, GuardDuty can detect compromised EC2 instances serving malware or mining bitcoin. It also monitors AWS account access behavior for signs of compromise, such as unauthorized infrastructure deployments, like instances deployed in a Region that has never been used, or unusual API calls, like a password policy change to reduce password strength.

GuardDuty informs you of the status of your AWS environment by producing security findings (p. 13) that you can view in the GuardDuty console or through Amazon CloudWatch events (p. 107).

Pricing for GuardDuty

For information about GuardDuty pricing, see Amazon GuardDuty Pricing.

Accessing GuardDuty

You can work with GuardDuty in any of the following ways:

GuardDuty Console

https://console.aws.amazon.com/guardduty

The console is a browser-based interface to access and use GuardDuty.

AWS SDKs

AWS provides software development kits (SDKs) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and more). The SDKs provide a convenient way to create programmatic access to GuardDuty. For information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

GuardDuty HTTPS API

You can access GuardDuty and AWS programmatically by using the GuardDuty HTTPS API, which lets you issue HTTPS requests directly to the service. For more information, see the GuardDuty API reference.

Getting started with GuardDuty

This tutorial provides a hands-on introduction to GuardDuty. The minimum requirements for enabling GuardDuty as a standalone account or as a GuardDuty administrator with AWS Organizations are covered in Step 1. Steps 2 through 5 cover using additional features recommended by GuardDuty to get the most out of your findings.

Topics

Before you begin

GuardDuty is a monitoring service that analyzes AWS CloudTrail management and Amazon S3 data events, VPC Flow Logs, and DNS logs to generate security findings for your account. Once GuardDuty is enabled, it starts monitoring your environment immediately. GuardDuty can be disabled at any time to stop it from processing all AWS CloudTrail events, VPC Flow Logs, and DNS logs.

> **Note**
> You do not need to enable AWS CloudTrail, Amazon S3 data events, VPC Flow Logs, and DNS logs before starting GuardDuty. Amazon GuardDuty pulls independent streams of data directly from those services. For more information see How Amazon GuardDuty uses its data sources (p. 10).

Note the following about enabling GuardDuty:

- GuardDuty is a Regional service, meaning any of the configuration procedures you follow on this page must be repeated in each region that you want to monitor with GuardDuty.

  We highly recommend that you enable GuardDuty in all supported AWS Regions. This enables GuardDuty to generate findings about unauthorized or unusual activity even in Regions that you are not actively using. This also enables GuardDuty to monitor AWS CloudTrail events for global AWS services such as IAM. If GuardDuty is not enabled in all supported Regions, its ability to detect activity that involves global services is reduced. For a full list of regions in which GuardDuty is supported see Regions and endpoints (p. 191).
- Any user with administrator privileges in an AWS Account can enable GuardDuty, however, following the security best practice of least privilege, it is recommended that you create an IAM user, role, or group to manage GuardDuty specifically. For information on the permissions required to enable GuardDuty see Permissions required to enable GuardDuty (p. 160).
- When you enable GuardDuty for the first time in any region, it creates a service-linked role for your account called `AWSServiceRoleForAmazonGuardDuty`. This role includes the permissions and the trust policies that allow GuardDuty to consume and analyze events directly from AWS CloudTrail, VPC Flow logs, and DNS logs in order to generate security findings. For more information, see Service-linked role permissions for GuardDuty (p. 167). For more information about service-linked roles, see Using service-linked roles.

- When you enable GuardDuty for the first time in any Region your AWS account is automatically enrolled in a 30-day GuardDuty free trial for that Region.

Step 1: Enable Amazon GuardDuty

The first step to using GuardDuty is to enable it in your account. Once enabled, GuardDuty will immediately begin to monitor for security threats in the current region.

If you want to manage GuardDuty findings for other accounts within your organization as a GuardDuty administrator, you must add member accounts and enable GuardDuty for them as well. Choose an option to learn how to enable GuardDuty for your environment.

Standalone account environment

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/
2. Choose **Get Started**.
3. Choose **Enable GuardDuty**.

Multi-account environment

> **Important**
> As prerequisites for this process, you must be in the same organization as all the accounts you wish to manage, and have access to the AWS Organizations management account in order to delegate an administrator for GuardDuty within your organization. Additional permissions may be required to delegate an administrator, for more info see Permissions required to designate a delegated administrator (p. 127).

To delegate an Administrator for GuardDuty

1. Log in to the AWS Organizations management account
2. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Is GuardDuty already enabled in your account?

   - If GuardDuty is not already enabled, you can select **Get Started** and then designate a GuardDuty delegated administrator on the **Welcome to GuardDuty** page.
   - If GuardDuty is enabled, you can designate a GuardDuty delegated administrator on the **Settings** page.
3. Enter the twelve-digit AWS account ID of the account that you want to designate as the GuardDuty delegated administrator for the organization and choose **Delegate**.

   > **Note**
   > If GuardDuty is not already enabled, designating a delegated administrator will enable GuardDuty for that account in your current Region.

To add member accounts

This procedure covers adding members accounts to a GuardDuty delegated administrator account through AWS Organizations. There is also the option to add members by invitation. To learn more about both methods for associating members in GuardDuty see Managing multiple accounts in Amazon GuardDuty (p. 124).

1. Log in to the delegated administrator account
2. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
3. In the navigation panel, choose **Settings**, and then choose **Accounts**.

The accounts table displays all of the accounts in the organization.

4. Choose the accounts that you want to add as members by selecting the box next to the account ID. Then from the **Action** menu select **Add member**.

   Tip
   You can automate adding new accounts as members by turning on the **Auto-enable** feature, however, this only applies to accounts that join your organization after the feature has been enabled.

Step 2: Generate sample findings and explore basic operations

When GuardDuty discovers a security issue it generates a finding. A GuardDuty finding is a data set containing details relating to that unique security issue. The finding's details can be used to help you investigate the issue.

GuardDuty supports generating sample findings with placeholder values, which can be used to test GuardDuty functionality and familiarize yourself with findings before needing to respond to a real security issue discovered by GuardDuty. Follow the guide below to generate sample findings for each finding type available in GuardDuty, for additional ways to generate sample findings, including generating a simulated security event within your account, see Sample findings (p. 21).

To create and explore sample findings

1. In the navigation pane, choose **Settings**.
2. On the **Settings** page, under **Sample findings**, choose **Generate sample findings**.
3. In the navigation pane, choose **Findings**. The sample findings are displayed on the **Current findings** page with the prefix **[SAMPLE]**.
4. Select a finding from the list to display details for the finding.

   - You can review the different information fields available in the finding details pane. Different types of findings can have different fields. For more information on the available fields across all finding types see Finding details (p. 13). From the details pane you can take the following actions:

     - Select the **finding ID** at the top of the pane to open the complete JSON details for the finding. The complete JSON file can also be downloaded from this panel. The JSON contains some additional information not included in the console view and is the format that can be ingested by other tools and services.
     - View the **Resource affected** section. In a real finding the information here will help you identify a resource in your account that should be investigated and will include links to the appropriate AWS console for actionable resources.
     - Select the + or - looking glass icons to create an inclusive or exclusive filter for that detail. For more information on finding filters see Filtering findings (p. 91)
5. Archive all your sample findings

   a. Select all findings by selection the check box at the top of the list.
   b. Deselect any findings you wish to keep.
   c. Select the **Actions** menu and then select **Archive** to hide the sample findings.

      Note
      To view the archived findings select **Current** and then **Archived** to switch the findings view.

Step 3: Configure GuardDuty findings export to an S3 bucket

GuardDuty recommends setting up findings export, which allows you to export your findings to an S3 bucket for indefinite storage beyond the GuardDuty 90 day storage limit. This allows you to keep records of findings or track issues within your environment over time. The process outlined here walks you through setting up a new S3 bucket and creating a new KMS key to encrypt findings from within the console, for more information about this, including how to use your own existing bucket or a bucket in another account, see Exporting findings (p. 101).

To configure S3 export

1. In the navigation pane, choose **Settings**.
2. Under **Findings export options** select **Configure now**.
3. Select **New bucket**.
   - Enter a unique name for your bucket.
4. To encrypt findings, you will need a KMS key with a policy that allows GuardDuty to use it. You can attach the policy outlined in the following section to an existing key or create a new KMS key from the console. The key must be in the same Region as your S3 bucket. For more information on KMS keys see create a key.

   To create a new key, Select **Go to KMS console to create a new key** to open the KMS console in a new tab and follow these instructions:

   a. In the KMS console select **Create key**.
   b. Choose **Symmetric** then choose **Next**.
   c. Give your key an alias and choose **Next**.
   d. Choose **Next** and then **Next** again to accept the default administration and usage permissions.
   e. In the **Review and edit key policy** pane add the following statement to the `"Statements":` section.

      Replace *`Region`* with the Region that the KMS key is in. Replace *`111122223333`* with the AWS account number of the account that owns the bucket. Replace *`KMSKeyId`* with the key ID of the key that you chose for encryption and replace *`SourceDetectorID`* with the source account's GuardDuty detector ID for the current Region.

      This statement allows GuardDuty to use only the key that you changed the policy for. When editing the key policy make sure your JSON syntax is valid, if you add the statement before the final statement you must add a comma after the closing bracket.

```
{
    "Sid": "AllowGuardDutyKey",
    "Effect": "Allow",
    "Principal": {
        "Service": "guardduty.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey",
    "Resource": "arn:aws:kms:Region:111122223333:key/KMSKeyId",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "111122223333",
            "aws:SourceArn":
 "arn:aws:guardduty:Region:111122223333:detector/SourceDetectorID"
        }
    }
```

```
}
```

5. Return to the GuardDuty console tab where you were configuring findings export.

   - Refresh the console by selecting the Refresh icon next to **S3 bucket**, now choose the key you just created from the **Key alias** list then choose **Save**.

6. (Optional) you can test your new export settings by generating sample findings with the process in Step 2. The new sample findings will appear within 5 minutes as entries in the S3 bucket created by GuardDuty in step 3.

Step 4: Set up GuardDuty finding alerts through SNS

GuardDuty integrates with Amazon EventBridge which can be used to send findings data to other applications and services for processing. With EventBridge you can use GuardDuty findings to trigger automatic responses to your findings by connecting finding events to targets such as AWS Lambda functions, Amazon EC2 Systems Manager automation, Amazon Simple Notification Service (SNS) and more.

In this example you will create an SNS topic to be the target of an EventBridge rule, then you'll use EventBridge to create a rule that captures findings data from GuardDuty. The resulting rule forwards the finding details to an email address. To learn how you can send findings to Slack or Chime, as well as modify the types of findings alerts are sent for, see Setup an Amazon SNS topic and endpoint (p. 109).

To create an SNS topic for your findings alerts

1. Sign in to the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. Select **Topics** from the navigation pane and then **Create Topic**.
3. In the Create topic section, select **Standard**. Next, enter a Topic name, for example **`GuardDuty`**. Other details are optional.
4. Choose **Create Topic**. The Topic details for your new topic will open.
5. In the Subscriptions section select **Create Subscription**.
6. From the **Protocol** menu, select **Email**.

   a. In the **Endpoint** field add the email address you would like to receive notifications at. You will be required to confirm your subscription through your email client after creating it.
   b. Choose **Create Subscription**.

7. Check for a subscription message in your inbox and choose **Confirm Subscription**.

   Note
   You can check email confirmation status by selecting the subscription name from the **Topics** list in the SNS console.

To create an EventBridge rule to capture GuardDuty findings and format them

1. Open the EventBridge console at https://console.aws.amazon.com/events/.
2. Select **Rules** from the navigation pane and then **Create Rule**.

   a. Choose **Event Pattern** and then **Pre-defined pattern by service**.
   b. From the **Service provider** menu, choose **AWS**.
   c. From the **Service name** menu, choose **GuardDuty**.
   d. From the **Event Type** menu, choose **GuardDuty Finding**.

3. In the **Select targets** pane, from the **Target** menu, choose **SNS Topic**.
4. For **Topic** select the name of the SNS topic you created earlier.
5. Expand **Configure Input** and Choose **Input transformer**. The Input transformer code will allow you to format the JSON finding data sent from GuardDuty into an easy human-readable message.
6. Copy the following code and paste it into the **Input Path** field.

```
{
    "severity": "$.detail.severity",
    "Finding_ID": "$.detail.id",
    "Finding_Type": "$.detail.type",
    "region": "$.region",
    "Finding_description": "$.detail.description"
}
```

7. Copy the following code and paste it into the **Input Template** field to format the email.

```
"You have a severity <severity> GuardDuty finding type <Finding_Type> in the <region>
 region."
"Finding Description:"
"<Finding_description>. "
"For more details open the GuardDuty console at https://console.aws.amazon.com/
guardduty/home?region=<region>#/findings?search=id%3D<Finding_ID>"
```

8. Choose **Create** to finalize your rule.
9. (Optional) you can test your new rule by generating sample findings with the process in Step 2. You will receive an email for each sample finding generated.

Next steps

As you continue to use GuardDuty, you will come to understand the types of findings that are relevant to your environment. Whenever you receive a new finding, you can find information, including remediation recommendations about that finding, by selecting **Learn more** from the finding description in the finding details pane, or by searching for the finding name on the Finding types (p. 26) page.

The following features will help you tune GuardDuty so that it can provide the most relevant findings for your AWS environment:

- To easily sort findings based on specific criteria, such as instance ID, account ID, S3 bucket name, and more, you can create and save filters within GuardDuty. For more information see Filtering findings (p. 91).
- If you are receiving findings for expected behavior in your environment, you can automatically archive findings based on criteria you define with suppression rules (p. 94).
- To prevent findings from being generated from a subset of trusted IPs, or to have GuardDuty monitor IPs outside it's normal monitoring scope you can set up Trusted IP and threat lists (p. 97).

Concepts and terminology

As you get started with Amazon GuardDuty, you can benefit from learning about its key concepts.

Account

A standard Amazon Web Services (AWS) account that contains your AWS resources. You can sign in to AWS with your account and enable GuardDuty.

You can also invite other accounts to enable GuardDuty and become associated with your AWS account in GuardDuty. If your invitations are accepted, your account is designated as the **administrator** GuardDuty account, and the added accounts become your **member** accounts. You can then view and manage those accounts' GuardDuty findings on their behalf.

Users of the administrator account can configure GuardDuty as well as view and manage GuardDuty findings for their own account and all of their member accounts. You can have up to 5000 member accounts in GuardDuty.

Users of member accounts can configure GuardDuty as well as view and manage GuardDuty findings in their account (either through the GuardDuty management console or GuardDuty API). Users of member accounts can't view or manage findings in other members' accounts.

An AWS account can't be a GuardDuty administrator and member account at the same time. An AWS account can accept only one membership invitation. Accepting a membership invitation is optional.

For more information, see Managing multiple accounts in Amazon GuardDuty (p. 124).

Detector

All GuardDuty findings are associated with a detector, which is an object that represents the GuardDuty service. The detector is a regional entity, and a unique detector is required in each region GuardDuty operates in. When you enable GuardDuty in a region a new detector with a unique 32 alphanumeric detector ID is generated in that region. The detector ID format looks like this:

```
12abc34d567e8fa901bc2d34e56789f0
```

You can find your detector ID for your current region in the console from the **Settings** pane, or programmatically using the ListDetectors API.

> **Note**
> In multiple account environments all findings for member accounts roll up to the administrator account's detector.

Some GuardDuty functionality is configured through the detector, such as configuring CloudWatch Events notification frequency and the enabling or disabling of optional data sources for GuardDuty to process.

Data source

The origin or location of a set of data. To detect unauthorized and unexpected activity in your AWS environment, GuardDuty analyzes and processes data from AWS CloudTrail event logs, VPC Flow Logs, and DNS logs. For more information, see How Amazon GuardDuty uses its data sources (p. 10).

Finding

A potential security issue discovered by GuardDuty. For more information, see Understanding Amazon GuardDuty findings (p. 13).

Findings are displayed in the GuardDuty console and contain a detailed description of the security issue. You can also retrieve your generated findings by calling the GetFindings and ListFindings API operations.

You can also see your GuardDuty findings through Amazon CloudWatch events. GuardDuty sends findings to Amazon CloudWatch via HTTPS protocol. For more information, see Creating custom responses to GuardDuty findings with Amazon CloudWatch Events (p. 107).

Suppression rule

Suppression rules allow you to create very specific combinations of attributes to suppress findings. For example, you can define a rule through the GuardDuty filter to auto-archive `Recon:EC2/Portscan` from only those instances in a specific VPC, running a specific AMI, or with a specific EC2 tag. This rule would result in port scan findings being automatically archived from the instances that meet the criteria. However, it still allows alerting if GuardDuty detects those instances conducting other malicious activity, such as crypto-currency mining.

Suppression rules defined in the GuardDuty administrator account apply to the GuardDuty member accounts. GuardDuty member accounts can't modify suppression rules.

With suppression rules, GuardDuty still generates all findings. Suppression rules provide suppression of findings while maintaining a complete and immutable history of all activity.

Typically suppression rules are used to hide findings that you have determined as false positives for your environment, and reduce the noise from low-value findings so you can focus on larger threats. For more information, see Suppression rules (p. 94)

Trusted IP list

A list of trusted IP addresses for highly secure communication with your AWS environment. GuardDuty does not generate findings based on trusted IP lists. For more information, see Working with trusted IP lists and threat lists (p. 97).

Threat list

A list of known malicious IP addresses. GuardDuty generates findings based on threat lists. For more information, see Working with trusted IP lists and threat lists (p. 97).

How Amazon GuardDuty uses its data sources

To detect unauthorized and unexpected activity in your AWS environment, GuardDuty analyzes and processes data from the sources described in this topic. GuardDuty uses these data sources to detect anomalies involving the following AWS resource types: IAM access keys, EC2 instances, S3 buckets, and Amazon EKS resources. While in transit from these data sources to GuardDuty, all of the log data is encrypted. GuardDuty extracts various fields from these logs for profiling and anomaly detection, and then discards the logs.

The following sections describe how GuardDuty uses each supported data source.

Topics

AWS CloudTrail event logs

AWS CloudTrail provides you with a history of AWS API calls for your account, including API calls made using the AWS Management Console, the AWS SDKs, the command line tools, and certain AWS services. CloudTrail also allows you to identify which users and accounts called AWS APIs for services that support CloudTrail, the source IP address that the calls were made from, and when the calls occurred. For more information, see the AWS CloudTrail User Guide. GuardDuty can monitor both CloudTrail management events, and optionally, CloudTrail data events for S3.

When you enable GuardDuty, it immediately starts analyzing your CloudTrail event logs. It consumes CloudTrail management and S3 data events directly from CloudTrail through an independent and duplicative stream of events. There is no additional charge for GuardDuty to access CloudTrail events.

GuardDuty does not manage your CloudTrail events or affect your existing CloudTrail configurations in any way. To manage access and retention of your CloudTrail events directly you must use the CloudTrail service console or API. For more information see Viewing Events with CloudTrail Event History.

How GuardDuty Handles AWS CloudTrail Global Events

Another important detail about the way GuardDuty uses CloudTrail as a data source is the handling and processing of CloudTrail global events. For most services, events are recorded in the Region where the action occurred. For global services such as IAM, AWS Security Token Service, Amazon S3, Amazon CloudFront, and Route 53, events are delivered to any trail that includes global services, and are logged as occurring in the US East (N. Virginia) Region. For more information, see About Global Service Events.

GuardDuty processes all events that come into a Region, including global events that CloudTrail sends to all Regions. This allows GuardDuty to maintain user and role profiles in each Region, and enables it to accurately detect credentials that are being maliciously used across Regions.

We highly recommend that you enable GuardDuty in all supported AWS Regions. This enables GuardDuty to generate findings about unauthorized or unusual activity even in Regions that you are not actively using. This also enables GuardDuty to monitor AWS CloudTrail events for global AWS services. If GuardDuty is not enabled in all supported Regions, its ability to detect activity that involves global services is reduced.

AWS CloudTrail management events

Management events are also known as control plane events. These events provide insight into management operations that are performed on resources in your AWS account.

The following are examples of CloudTrail management events that GuardDuty monitors:

- Configuring security (IAM `AttachRolePolicy` API operations)
- Configuring rules for routing data (Amazon EC2 `CreateSubnet` API operations)
- Setting up logging (AWS CloudTrail `CreateTrail` API operations)

AWS CloudTrail S3 data events

Data events, also known as data plane operations, provide insight into the resource operations performed on or within a resource. They are often high-volume activities.

The following are examples of CloudTrail S3 data events that GuardDuty can monitor:

- `GetObject` API operations
- `PutObject` API operations
- `ListObjects` API operations
- `DeleteObject` API operations

S3 data event monitoring is enabled by default for new accounts. However, this data source is optional and can be enabled or disabled for any account or Region at any time. For more information about configuring Amazon S3 as a data source see Amazon S3 protection in Amazon GuardDuty (p. 141).

Kubernetes audit logs

Kubernetes audit logs capture sequential actions within your Amazon EKS cluster, including activities from users, applications using the Kubernetes API, and the control plane. Audit logging is a component of all Kubernetes clusters. To learn more, see Auditing in the Kubernetes documentation.

Amazon EKS allows Kubernetes audit logs to be ingested as Amazon CloudWatch Logs through the EKS control plane logging feature. When you enable Kubernetes protection in GuardDuty, GuardDuty immediately starts analyzing Kubernetes audit logs for your Amazon EKS resources. It consumes audit log events directly from the Amazon EKS control plane logging feature through an independent and duplicative stream of flow logs. This process does not require any additional set up or affect any existing Amazon EKS control plane logging configurations that you might have.

GuardDuty doesn't manage your Amazon EKS control plane logging or make Kubernetes audit logs accessible in your account if you have not enabled them for Amazon EKS. To manage access to and

retention of your Kubernetes audit logs, you must configure the Amazon EKS control plane logging feature. For more information, see Enabling and disabling control plane logs In the Amazon EKS documentation.

Kubernetes audit log monitoring is enabled by default all GuardDuty accounts. However, this data source is optional and can be enabled or disabled for any account or Region at any time. For more information about configuring Kubernetes audit logs a data source, see Kubernetes protection in Amazon GuardDuty (p. 137).

VPC Flow Logs

The VPC Flow Logs feature of Amazon VPC captures information about the IP traffic going to and from network interfaces within your environment.

When you enable GuardDuty, it immediately starts analyzing your VPC Flow Logs data. It consumes VPC Flow Log events directly from the VPC Flow Logs feature through an independent and duplicative stream of flow logs. This process does not affect any existing flow log configurations that you might have.

GuardDuty doesn't manage your flow logs or make them accessible in your account. To manage access to and retention of your flow logs, you must configure the VPC Flow Logs feature.

There is no additional charge for GuardDuty access to flow logs. However, enabling flow logs for retention or use in your account falls under existing pricing. For more information, see VPC Flow Logs.

DNS logs

If you use AWS DNS resolvers for your Amazon EC2 instances (the default setting), then GuardDuty can access and process your request and response DNS logs through the internal AWS DNS resolvers. If you use another DNS resolver, such as OpenDNS or GoogleDNS, or if you set up your own DNS resolvers, then GuardDuty cannot access and process data from this data source.

When you enable GuardDuty, it immediately starts analyzing your DNS logs from an independent stream of data. This data stream is separate from the data provided through the Route 53 Resolver query logging feature. Configuration of this feature does not effect GuardDuty analysis.

Understanding Amazon GuardDuty findings

A GuardDuty finding represents a potential security issue detected within your network. GuardDuty generates a finding whenever it detects unexpected and potentially malicious activity in your AWS environment.

You can view and manage your GuardDuty findings on the **Findings** page in the GuardDuty console or by using the GuardDuty CLI or API operations. For an overview of the ways you can manage findings see Managing Amazon GuardDuty findings (p. 91).

Topics:

Finding details

In the Amazon GuardDuty console, you can view finding details in a finding summary section. Finding details vary based on finding type.

There are two primary details that will determine what kinds of information are available for any finding. The first is the resource type, which can be `AccessKey`, `Instance`, or `S3Bucket`. The second detail that determines finding information is **Resource Role**. Resource role, can be `Target` for access keys, meaning the resource was the target of suspicious activity. For instance type findings, resource role can also be `Actor`, which means that your resource was the actor carrying out suspicious activity. This topic describes some of the commonly available details for findings.

Finding summary

A finding's summary section contains the most basic identifying features of the finding, including the following information:

- **Finding type** – A formatted string representing the type of activity that triggered the finding. For more information, see GuardDuty finding format (p. 19).

- **Finding ID** – A unique Finding ID for this finding type and set of parameters. New occurrences of activity matching this pattern will be aggregated to the same ID.
- **Severity** – a finding's assigned severity level of either High, Medium, or Low. For more information, see Severity levels for GuardDuty findings (p. 23).
- **Region** – the AWS Region in which the finding was generated. For more information about supported Regions, see Regions and endpoints (p. 191)
- **Count** – The number of times GuardDuty has aggregated an activity matching this pattern to this finding ID.
- **Account ID** – the ID of the AWS account in which the activity took place that prompted GuardDuty to generate this finding.
- **Resource ID** – the ID of the AWS resource against which the activity took place that prompted GuardDuty to generate this finding.
- **Created at** – the time and date when this finding was first created. If this value differs from **Updated at**, it indicates that the activity has occurred multiple times and is an ongoing issue.

  Note
  Timestamps for findings in the GuardDuty console appear in your local time zone, while JSON exports and CLI outputs display timestamps in UTC.
- **Updated at** – The last time this finding was updated with new activity matching the pattern that prompted GuardDuty to generate this finding.

  Note
  Timestamps for findings in the GuardDuty console appear in your local time zone, while JSON exports and CLI outputs display timestamps in UTC.

Resource

The **Resource affected** gives details on the AWS resource that was targeted by the trigger activity. The information available varies based on resource type and action type.

Resource role – The role of the AWS resource that triggered the finding. This value can be **TARGET** or **ACTOR**, and represents whether your resource was the target of the suspicious activity or the actor that preformed the suspicious activity.

Resource type – the type of the affected resource. A finding can include multiple resources types if multiple resources were involved. The resource types are **AccessKey**, **S3 bucket**, **KubernetesCluster** or **Instance**. Depending on the resource type, different finding details are available. Select a resource option tab to learn about the details available for that resource.

Instance

Instance details:

Note
Some instance details may be missing if the instance has already been terminated or if the underlying API invocation originated from an EC2 instance in a different Region when making a cross-Region API call.

- **Instance ID** – the ID of the EC2 instance involved in the activity that prompted GuardDuty to generate the finding.
- **Instance Type** – the type of the EC2 instance involved in the finding.
- **Launch Time** – the time and date the instance was launched.
- **Outpost ARN** – The Amazon Resource Name (ARN) of the AWS Outposts. Only applicable to AWS Outposts instances. For more information, see What is AWS Outposts?
- **Security Group Name** – The name of the Security Group attached to the involved instance.

- **Security Group ID** – The ID of the Security Group attached to the involved instance.
- **Instance state** – The current state of the targeted instance.
- **Availability Zone** – The AWS Region Availability Zone in which the involved instance is located.
- **Image ID** – The ID of the Amazon Machine Image used to build the instance involved in the activity.
- **Image Description** – A description of the ID of the Amazon Machine Image used to build the instance involved in the activity.
- **Tags** – A list of tags attached to this resource, listed in the format of `key:value`.

Access Key

Access Key details:

- **Access key ID** – access key ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
- **Principal ID** – the principal ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
- **User type** – the type of user engaged in the activity that prompted GuardDuty to generate the finding. For more information, see CloudTrail userIdentity element.
- **User name** – The name of the user engaged in the activity that prompted GuardDuty to generate the finding.

S3 bucket

S3 bucket details:

- **Name** – The name of the bucket involved in the finding.
- **ARN** – The ARN of the bucket involved in the finding.
- **Owner** – The canonical user ID of the user that owns the bucket involved in the finding. For more information on canonical user IDs see AWS account identifiers.
- **Type** – The type of bucket finding, can be either **Destination** or **Source**.
- **Default server side encryption** – encryption details for the bucket.
- **Bucket Tags** – a list of tags attached to this resource, listed in the format of `key:value`.
- **Effective Permissions** – An evaluation of all effective permissions and policies on the bucket that indicates whether the involved bucket is publicly exposed. Values can be PUBLIC or NOT PUBLIC.

Kubernetes Cluster

Kubernetes Cluster details:

- **User details** – Details of the Kubernetes user that performed the API action that prompted GuardDuty to generate the finding.
  - **Username** – The name of the Kubernetes user that performed the API action that prompted GuardDuty to generate the finding.
  - **Groups** – A list of permissions groups the Kubernetes user belongs to.
- **Access key details** – This section includes Access key details identifying the IAM User or Role associated with the activity when the user is an IAM identity accessing the cluster through EKS.
  - **Access key ID** – Access key ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
  - **Principal ID** – The principal ID of the user engaged in the activity that prompted GuardDuty to generate the finding.

- **User type** – The type of user engaged in the activity that prompted GuardDuty to generate the finding. For more information, see CloudTrail userIdentity element.
- **User name** – The name of the user engaged in the activity that prompted GuardDuty to generate the finding.

- **Cluster details** – Details on the Kubernetes cluster involved in the finding.
- **EKS details** – Details about the Amazon EKS resource involved in the finding.
- **Workload Details** – Details on the Kubernetes workload resources that were involved in the finding,.
  - Workload Type – The type of workload resource used to manage the pod implicated in the finding, this value can be either `Deployment`, `ReplicaSet`, `StatefulSet`, `DaemonSet`, `Job`, or `CronJob`.

Action

A finding's **Action** gives details on the type of activity that triggered the finding. The information available varies based on action type.

- **Action type** – The finding activity type. This value can be **NETWORK_CONNECTION**, **PORT_PROBE**, **DNS_REQUEST**, or **AWS_API_CALL**. The information available varies based on action type:
  - **NETWORK_CONNECTION** – indicates that network traffic was exchanged between the identified EC2 instance and the remote host. This action type has the following additional information:
    - **Connection direction** – The network connection direction observed in the activity that prompted GuardDuty to generate the finding. The values can be one of the following:
      - **INBOUND** – indicates that a remote host initiated a connection to a local port on the identified EC2 instance in your account.
      - **OUTBOUND** – indicates that the identified EC2 instance initiated a connection to a remote host.
      - **UNKNOWN** – indicates that GuardDuty could not determine the direction of the connection.
    - **Protocol** – the network connection protocol observed in the activity that prompted GuardDuty to generate the finding.
    - **Local IP** – The original source IP of the traffic that triggered the finding. This info can be used to distinguish between the IP address of an intermediate layer through which traffic flows, and the original source IP address of the traffic that triggered the finding. For example the IP address of an EKS pod as opposed to the IP address of the instance on which the EKS pod is running.
    - **Blocked** – Indicates whether the targeted port is blocked.
  - **PORT_PROBE** – indicates that a remote host probed the identified EC2 instance on multiple open ports. This action type has the following additional information:
    - **Local IP** – The original source IP of the traffic that triggered the finding. This info can be used to distinguish between the IP address of an intermediate layer through which traffic flows, and the original source IP address of the traffic that triggered the finding. For example the IP address of an EKS pod as opposed to the IP address of the instance on which the EKS pod is running.
    - **Blocked** – Indicates whether the targeted port is blocked.
  - **DNS_REQUEST** – indicates that the identified EC2 instance queried a domain name. This action type has the following additional information:
    - **Protocol** – the network connection protocol observed in the activity that prompted GuardDuty to generate the finding.
    - **Blocked** – Indicates whether the targeted port is blocked.
  - **AWS_API_CALL** – indicates that an AWS API was invoked. This action type has the following additional information:
    - **API** – the name of the API operation that was invoked and thus prompted GuardDuty to generate this finding.

Note
These operations can also include non-API events captured by AWS CloudTrail. For more information, see Non-API events captured by CloudTrail.

- **User Agent** – The user agent that made the API request. This value tells you whether the call was made from the AWS Management Console, an AWS service, the AWS SDKs or the AWS CLI.
- **ERROR CODE** – if the finding was triggered by a failed API call this displays the error code for that call.
- **Service name** – the DNS name of the service that attempted to make the API call that triggered the finding.

Actor or Target

A finding has an **Actor** section if the **Resource role** was `TARGET`. This indicates that your resource was targeted by suspicious activity, and the **Actor** section contains details on the entity that targeted your resource.

A finding has a **Target** section if the **Resource role** was `ACTOR`. This indicates that your resource was involved in suspicious activity against a remote host, and this section contains information on the IP or domain that your resource targeted.

The information available in the **Actor** or **Target** section can include the following:

- **IP address** – the IP address involved in the activity that prompted GuardDuty to generate the finding.
- **Location** – location information for the IP address involved in the activity that prompted GuardDuty to generate the finding.
- **Organization** – ISP organization information of the IP address involved in the activity that prompted GuardDuty to generate the finding.
- **Port** – the port number involved in the activity that prompted GuardDuty to generate the finding.
- **Domain** – the domain involved in the activity that prompted GuardDuty to generate the finding.

Additional information

All findings have an **Additional information** section that can include the following information:

- **Threat list name** – the name of the threat list that includes the IP address or the domain name involved in the activity that prompted GuardDuty to generate the finding.
- **Sample** – a true or false value that indicates whether this is a sample finding.
- **Archived** – a true or false value that indicates whether this is finding has been archived.
- **Unusual** – activity details that were not observed historically. These can include an unusual (previously not observed) user, or location, or time.
- **Unusual protocol**– the network connection protocol involved in the activity that prompted GuardDuty to generate the finding.

Evidence

Findings based on DNS logs have an **Evidence** section that includes the following information:

- **Threat intelligence details** – the name of the threat list that the recognized `Threat name` appears on.
- **Threat name** – the name of the malware family, or other identifier, associated with the threat.

Anomalous behavior

Findings types that end in **AnomalousBehavior** indicate that the finding was generated by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all API requests to your account and identifies anomalous events that are associated with tactics used by adversaries. The ML model tracks various factors of the API request, such as the user that made the request, the location the request was made from, and the specific API that was requested.

Details about which factors of the API request are unusual for the CloudTrail user identity that invoked the request can be found in the finding details. The identities are defined by the CloudTrail userIdentity Element, possible values are: `Root`, `IAMUser`, `AssumedRole`, `FederatedUser`, `AWSAccount` or `AWSService`.

In addition to the details available for all GuardDuty findings that are associated with API activity, **AnomalousBehavior** findings have additional details that are outlined in the following section. These details can be viewed in the console and are also available in the finding's JSON.

- **Anomalous APIs** – a list of API requests that were invoked by the user identity in proximity to the primary API request associated with the finding. This pane further breaks down the details of the API event in the following ways:
  - The first API listed is the primary API, which is the API request associated with the highest-risk observed activity. This is the API that triggered the finding and correlates to the attack stage of the finding type. This is also the API that is detailed under the **Action** section in the console, and in the finding's JSON.
  - Any other APIs listed are additional anomalous APIs from the listed user identity observed in proximity to the primary API. If there is only one API on the list, the ML model did not identify any additional API requests from that user identity as anomalous.
  - The list of APIs is divided based on whether an API was **successfully called**, or if the API was unsuccessfully called, meaning an error response was received. The type of error response received is listed above each unsuccessfully-called API. Possible error response types are: `access denied`, `access denied exception`, `auth failure`, `instance limit exceeded`, `invalid permission - duplicate`, `invalid permission - not found`, `operation not permitted`.
  - APIs are categorized by their associated service.

    Note
    For more context, select **Usual APIs** to open a pane that gives details on the top APIs, to a maximum of 20, usually seen for both the user identity and all users within the account. The APIs are marked **rare** (less than once a month), **infrequent** (a few times a month), or **frequent** (daily to weekly), depending on how often they are used within your account.
- **Unusual Behavior (Account)** – this section gives additional details on the profiled behavior for your account. The information tracked in this panel includes:
  - **ASN Org** – The ASN Org the anomalous API call was made from.
  - **User Name** – The name of the user that made the anomalous API call.
  - **User Agent**– the user agent used to make the anomalous API call. The user agent is the method used to make the call such as `aws-cli` or `Botocore`.
  - **User Type** – The type of user that made the anomalous API call. Possible values are `AWS_SERVICE`, `ASSUMED_ROLE`, `IAM_USER`, or `ROLE`.
- **Unusual Behavior (User Identity)** – this section gives additional details on the profiled behavior for the **User Identity** involved with the finding. When a behavior is identified as unusual this means GuardDuty's ML model has not previously seen this user identity making this API call in this way within the training period. The following additional details about the **User Identity** are available:
  - **ASN Org** – The ASN Org the anomalous API call was made from.
  - **User Agent**– the user agent used to make the anomalous API call. The user agent is the method used to make the call such as `aws-cli` or `Botocore`.

Note
For more context on Unusual behaviors, select **Usual behaviors** in either the **Account** or **User ID** panel to open a pane that gives details on the expected behavior for you account for each of the following categories: **rare** (less than once a month), **infrequent** (a few times a month), or **frequent** (daily to weekly), depending on how often they are used within your account.

GuardDuty finding format

When GuardDuty detects suspicious or unexpected behavior in your AWS environment, it generates a finding. A finding is a notification that contains the details about a potential security issue that GuardDuty discovers. The finding details (p. 25) include information about what happened, which AWS resources were involved in the suspicious activity, when this activity took place, and other information.

One of the most useful pieces of information in the finding details is a **finding type**. The purpose of the finding type is to provide a concise yet readable description of the potential security issue. For example, the GuardDuty *Recon:EC2/PortProbeUnprotectedPort* finding type quickly informs you that somewhere in your AWS environment, an EC2 instance has an unprotected port that a potential attacker is probing.

GuardDuty uses the following format for naming the various types of findings that it generates:

ThreatPurpose:ResourceTypeAffected/ThreatFamilyName.DetectionMechanism!Artifact

Each part of this format represents an aspect of a finding type. These aspects have the following explanations:

- **ThreatPurpose** - describes the primary purpose of a threat, an attack type or a stage of a potential attack. See the following section for a complete list of GuardDuty threat purposes.
- **ResourceTypeAffected** - describes which AWS resource type is identified in this finding as the potential target of an adversary. Currently GuardDuty can generate findings for EC2, IAM, and S3 resources.
- **ThreatFamilyName** - describes the overall threat or potential malicious activity that GuardDuty is detecting. For example, a value of **NetworkPortUnusual** indicates that an EC2 instance identified in the GuardDuty finding has no prior history of communications on a particular remote port that also is identified in the finding.
- **DetectionMechanism** - describes the method in which GuardDuty detected the finding. This can be used to indicate a variation on a common finding type or a finding that GuardDuty used a specific mechanism to detect. For example, **Backdoor:EC2/DenialOfService.Tcp** indicates denial of service (DoS) was detected over TCP. The UDP variant is **Backdoor:EC2/DenialOfService.Udp**.

  A value of **.Custom** indicates that GuardDuty detected the finding based on your custom threat lists, while **.Reputation** indicates that GuardDuty detected the finding using a domain reputation score model.
- **Artifact** - describes a specific resource that is owned by a tool that is used in the malicious activity. For example, **DNS** in the finding type *CryptoCurrency:EC2/BitcoinTool.B!DNS* indicates that an EC2 instance is communicating with a known Bitcoin-related domain.

Threat Purposes

In GuardDuty a *threat purpose* describes the primary purpose of a threat, an attack type, or a stage of a potential attack. For example, some threat purposes, such as **Backdoor**, indicate a type of attack. However some threat purposes, such as **Impact** align with MITRE ATT&CK tactics. The MITRE ATT&CK tactics indicate different phases in an adversary's attack cycle. In the current release of GuardDuty, ThreatPurpose can have the following values:

Backdoor

This value indicates that an adversary has compromised an AWS resource and altered the resource so that it is capable of contacting its home command and control (C&C) server to receive further instructions for malicious activity.

Behavior

This value indicates that GuardDuty has detected activity or activity patterns that are different from the established baseline for the AWS resources involved.

CredentialAccess

This value indicates that GuardDuty has detected activity patterns that an adversary may use to steal credentials, such as account IDs or passwords, from your environment. This threat purpose is based on MITRE ATT&CK tactics

Cryptocurrency

This value indicates that GuardDuty has detected that an AWS resource in your environment is hosting software that is associated with cryptocurrencies (for example, Bitcoin).

DefenseEvasion

This value indicates that GuardDuty has detected activity or activity patterns that an adversary may use to avoid detection while infiltrating your environment. This threat purpose is based on MITRE ATT&CK tactics

Discovery

This value indicates that GuardDuty has detected activity or activity patterns that an adversary may use to expand their knowledge of your systems and internal networks. This threat purpose is based on MITRE ATT&CK tactics.

Exfiltration

This value indicates that GuardDuty has detected activity or activity patterns that an adversary may use when attempting to steal data from your network. This threat purpose is based on MITRE ATT&CK tactics.

Impact

This value indicates that GuardDuty has detected activity or activity patterns that suggest that an adversary is attempting to manipulate, interrupt, or destroy your systems and data. This threat purpose is based on MITRE ATT&CK tactics

InitialAccess

This threat purpose is based on MITRE ATT&CK tactics

Pentest

Sometimes owners of AWS resources or their authorized representatives intentionally run tests against AWS applications to find vulnerabilities, such as open security groups or access keys that are overly-permissive. These pen tests are done in an attempt to identify and lock down vulnerable resources before they are discovered by adversaries. However, some of the tools used by authorized pen testers are freely available and therefore can be used by unauthorized users or adversaries to run probing tests. Although GuardDuty can't identify the true purpose behind such activity, the **Pentest** value indicates that GuardDuty is detecting such activity, that it is similar to the activity generated by known pen testing tools, and that it could indicate malicious probing of your network.

Persistence

This value indicates that GuardDuty has detected activity or activity patterns that an adversary may use to try and maintain access to your systems even if their initial access route is cut off. For

example, this could include creating a new IAM user after gaining access through an existing user's compromised credentials. When the existing user's credentials are deleted, the adversary will retain access on the new user that was not detected as part of the original event. This threat purpose is based on MITRE ATT&CK tactics.

Policy

This value indicates that your AWS account is exhibiting behavior that goes against recommended security best practices.

PrivilegeEscalation

This value informs you that the involved principal within your AWS environment is exhibiting behavior that an adversary may use to gain higher-level permissions to your network. This threat purpose is based on MITRE ATT&CK tactics.

Recon

This value indicates that GuardDuty has detected activity or activity patterns that an adversary may use when preforming reconnaissance of your network to determine how they can broaden their access or utilize your resources. For example, this activity can include scoping out vulnerabilities in your AWS environment by probing ports, listing users, database tables, and so on.

Stealth

This value indicates that an adversary is actively trying to hide their actions. For example, they might use an anonymizing proxy server, making it extremely difficult to gauge the true nature of the activity.

Trojan

This value indicates that an attack is using Trojan programs that silently carry out malicious activity. Sometimes this software takes on an appearance of a legitimate program. Sometimes users accidentally run this software. Other times this software might run automatically by exploiting a vulnerability.

UnauthorizedAccess

This value indicates that GuardDuty is detecting suspicious activity or a suspicious activity pattern by an unauthorized individual.

Generating sample findings in GuardDuty

You can generate sample findings with Amazon GuardDuty to help you visualize and understand the various finding types that GuardDuty can generate. When you generate sample findings, GuardDuty populates your current findings list with one sample finding for each supported finding type.

The generated samples are approximations populated with placeholder values. These samples may look different from real findings for your environment, but you can use them to test various configurations for GuardDuty, such as your CloudWatch Events or filters. For a list of available values for finding types are listed in Finding types (p. 26) table.

To generate some common findings based on simulated activity within your environment see Automatically generating common GuardDuty findings (p. 22) below.

Generating sample findings through the GuardDuty console or API

Choose an access method to learn how to generate sample findings through that method.

Note
The console method generates one of each finding type. Single sample findings can only be generated through the API.

Console

Use the following procedure to generate sample findings. This process generates one sample finding for each GuardDuty finding type.

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Settings**.
3. On the **Settings** page, under **Sample findings**, choose **Generate sample findings**.
4. In the navigation pane, choose **Findings**. The sample findings are displayed on the **Current findings** page with the prefix **[SAMPLE]**.

API

You can generate a single sample finding matching any of the GuardDuty finding types through the CreateSampleFindings API, the available values for finding types are listed in Finding types (p. 26) table.

This is useful for the testing of CloudWatch Events rules or automation based on findings. The following example shows how to generate a single sample finding of the `Backdoor:EC2/DenialOfService.Tcp` type using the AWS CLI.

```
                    AWS  guardduty create-sample-findings --detector-
id yourRegionalDetectorId --finding-types Backdoor:EC2/DenialOfService.Tcp
```

The title of sample findings generated through these methods always begins with **`[SAMPLE]`** in the console. Additionally, sample findings have a value of `"sample": true` in the `additionalInfo` section of the finding JSON details.

Automatically generating common GuardDuty findings

You can use the following scripts to automatically generate several common GuardDuty findings. The **guardduty-tester.template** uses AWS CloudFormation to create an isolated environment with a bastion host, a tester Amazon EC2 instance that you can access through SSH, and two target EC2 instances. Then you can run **guardduty_tester.sh** to start an interaction between the tester EC2 instance, the target Windows EC2 instance, and the target Linux EC2 instance, to simulate five types of common attacks that GuardDuty can detect and notify you about with generated findings.

1. As a prerequisite, you must enable GuardDuty in the account and Region in which you want to run **guardduty-tester.template** and **guardduty_tester.sh**. For more information about enabling GuardDuty, see Getting started with GuardDuty (p. 2).

   You must also generate a new or use an existing EC2 key pair in each Region in which you want to run these scripts. This EC2 key pair is used as a parameter in the **guardduty-tester.template** script that you use to create a new CloudFormation stack. For more information about generating key pairs, see Amazon EC2 key pairs.
2. Create a new CloudFormation stack using **guardduty-tester.template**. For detailed instructions about creating a stack, see Creating a stack. Before you run **guardduty-tester.template**, modify it

with values for the following parameters: Stack Name to identify your new stack, Availability Zone where you want to run the stack, and Key Pair that you can use to launch the EC2 instances. Then you can use the corresponding private key to access EC2 instances through SSH.

The **guardduty-tester.template** takes around 10 minutes to run and complete. It creates your environment and copies **guardduty_tester.sh** onto your tester EC2 instance.

3. In the AWS CloudFormation console, choose the checkbox next to your new running AWS CloudFormation stack. In the displayed set of tabs, select the **Output** tab. Note the IP addresses assigned to the bastion host and the tester EC2 instance. You need both of these IP addresses in order to access the tester EC2 instance through SSH.
4. Create the following entry in your ~/.ssh/config file to log into your instance through the bastion host.

```
Host bastion
        HostName {Elastic IP Address of Bastion}
        User ec2-user
        IdentityFile ~/.ssh/{your-ssh-key.pem}
Host tester
        ForwardAgent yes
        HostName {Local IP Address of RedTeam Instance}
        User ec2-user
        IdentityFile ~/.ssh/{your-ssh-key.pem}
        ProxyCommand ssh bastion nc %h %p
        ServerAliveInterval 240
```

Now you can call $ ssh tester to log into your target EC2 instance. For more information about configuring and connecting to EC2 instances through bastion hosts, see https://aws.amazon.com/blogs/security/securely-connect-to-linux-instances-running-in-a-private-amazon-vpc/.

5. After you connect to the tester EC2 instance, run **guardduty_tester.sh** to initiate interaction between your tester and target EC2 instances, simulate attacks, and generate GuardDuty findings.

Severity levels for GuardDuty findings

Each GuardDuty finding has an assigned severity level and value that reflects the potential risk the finding could have to your network as determined by our security engineers. The value of the severity can fall anywhere within the 0.1 to 8.9 range, with higher values indicating greater security risk. To help you determine a response to a potential security issue that is highlighted by a finding, GuardDuty breaks down this range into, High, Medium, and Low severity levels.

Note
Values 0 and 9.0 to 10.0 are currently reserved for future use.

The following are the currently defined severity levels and values for the GuardDuty findings as well as general recommendations for each:

Severity level	Value range
High	8.9 - 7.0
A High severity level indicates that the resource in question (an EC2 instance or a set of IAM user credentials) is compromised and is actively being used for unauthorized purposes. We recommend that you treat any High severity finding security issue as a priority and take immediate remediation steps to prevent further unauthorized use of your resources. For example, clean up your EC2 instance or terminate it, or rotate the IAM credentials. See Remediation Steps (p. 116) for more details.	

Severity level	Value range
Medium	6.9 - 4.0
A Medium severity level indicates suspicious activity that deviates from normally observed behavior and, depending on your use case, may be indicative of a resource compromise. We recommend that you investigate the implicated resource at your earliest convenience. Remediation steps will vary by resource and Finding family, but in general, you should be looking to confirm that the activity is authorized and consistent with your use case. If you cannot identify the cause, or confirm the activity was authorized, you should consider the resource compromised and follow Remediation Steps (p. 116) to secure the resource. Here are some things to consider when reviewing a Medium level finding: • Check if an authorized user has installed new software that changed the behavior of a resource (for example, allowed higher than normal traffic, or enabled communication on a new port). • Check if an authorized user changed the control panel settings, for example, modified a security group setting. • Run an anti-virus scan on the implicated resource to detect unauthorized software. • Verify the permissions that are attached to the implicated IAM role, user, group, or set of credentials. These might have to be changed or rotated.	
Low	3.9 - 1.0
A low severity level indicates attempted suspicious activity that did not compromise your network, for example, a port scan or a failed intrusion attempt. There is no immediate recommended action, but it is worth making note of this information as it may indicate someone is looking for weak points in your network.	

GuardDuty finding aggregation

All findings are dynamic, meaning that, if GuardDuty detects new activity related to the same security issue it will update the original finding with the new information, instead of generating a new finding. This behaviour allows you to identify ongoing issues, without needing to look through multiple similar reports, and reduces the overall noise from security issues you are already aware of.

For example, for a UnauthorizedAccess:EC2/SSHBruteForce finding, multiple access attempts against your instance will be aggregated to the same finding ID, increasing the Count number in the finding's details. This is because that finding represents a single security issue with the instance indicating that the SSH port on the instance is not properly secured against this type of activity. However, if GuardDuty detects SSH access activity targeting a new instance in your environment, it will create a new finding with a unique finding ID to alert you to the fact that there is a security issue associated with the new resource.

When a finding is aggregated it is updated with information from the latest occurrence of that activity. This means that in the above example, if your instance is the target of a brute force attempt from a new actor, the finding details will be updated to reflect the remote IP of the most recent source and older information will be replaced. Full Information about individual activity attempts will still be available in your CloudTrail or VPC Flow Logs.

The criteria that alert GuardDuty to generate a new finding instead of aggregating an existing one is dependent on the finding type. The aggregation criteria for each finding type are determined by our security engineers to give you the best overview of distinct security issues within your account.

Locating and analyzing GuardDuty findings

Use the following procedure to view and analyze your GuardDuty findings.

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Choose **Findings** and then select a specific finding to view its details.

   The details for each finding will differ depending on the Finding type, resources involved, and nature of the activity. For more information on available finding fields see Finding details (p. 13).
3. (Optional) If you wish to archive a finding, select it from the list of your findings and then choose the **Actions** menu. Then choose **Archive**.

   Archived findings can be viewed by choosing **Archived** from the **Current** dropdown.

   Currently in GuardDuty users from GuardDuty member accounts can't archive findings.

   Important
   If you archive a finding manually using the procedure above, all subsequent occurrences of this finding (generated after the archiving is complete) are added to the list of your current findings. To never see this finding in your current list, you can auto-archive it. For more information, see Suppression rules (p. 94).
4. (Optional) To download a finding, select it from the list of your findings and then choose the **Actions** menu. Then choose **Export**. When you **Export** a finding, you can see its full JSON document.

   Note
   If, and only if, the confidence level of a GuardDuty finding is set to 0, the **Confidence** field is displayed in the full finding JSON. The presence of the **Confidence** field set to 0 indicates that this GuardDuty finding is a false positive.

Finding types

For information about important changes to the GuardDuty finding types, including newly added or retired finding types, see Document history for Amazon GuardDuty (p. 192).

For information about retired finding types see Retired finding types (p. 76).

GuardDuty 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 resource was the target of suspicious activity or the actor performing the activity.

The findings listed here include the data sources and models used to generate that finding type. For more information data sources and models see How Amazon GuardDuty uses its data sources (p. 10).

> **Note**
> Instance details may be missing for some EC2 findings if the instance has already been terminated or if the underlying API call was part of a cross-Region API call that originated from an EC2 instance in a different Region.

For all EC2 findings, it is recommended that you examine the resource in question to determine if it is behaving in an expected manner. If the activity is authorized, you can use Suppression Rules or Trusted IP lists to prevent false positive notifications for that resource. If the activity is unexpected, the security best practice is to assume the instance has been compromised and take the actions detailed in Remediating a compromised EC2 instance (p. 116).

Topics

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

- **Data source:** VPC Flow Logs

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.

> **Note**
> If the IP queried is log4j-related, then fields of the associated finding will include the following values:
>
> - service.additionalInfo.threatListName = Amazon
> - service.additionalInfo.threatName = Log4j Related

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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.

> **Note**
> If the domain name queried is log4j-related, then the fields of the associated finding will include the following values:
>
> - service.additionalInfo.threatListName = Amazon
> - service.additionalInfo.threatName = Log4j Related

> **Note**
> To test how GuardDuty generates 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`.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

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

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

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

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

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

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

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

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Backdoor:EC2/Spambot

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

Default severity: Medium

- **Data source:** VPC Flow Logs

This finding informs you that the listed EC2 instance in your AWS environment is communicating with a remote host on port 25. This behavior 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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Behavior:EC2/NetworkPortUnusual

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

Default severity: Medium

- **Data source:** VPC Flow Logs

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.

> **Note**
> If the EC2 instance communicated on port 389 or port 1389, then the associated finding severity will be modified to High, and the finding fields will include the following value:
>
> - service.additionalInfo.context = Possible log4j callback

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Behavior:EC2/TrafficVolumeUnusual

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

Default severity: Medium

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

CryptoCurrency:EC2/BitcoinTool.B

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

Default severity: High

- **Data source:** VPC Flow Logs

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 that can be exchanged for other currencies, products, and services. Bitcoin is a reward for bitcoin-mining and is highly sought after by threat actors.

Remediation recommendations:

If you use this EC2 instance to mine or manage cryptocurrency, or this instance is otherwise involved in blockchain activity, this finding could be expected activity for your environment. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `CryptoCurrency:EC2/BitcoinTool.B`. The second filter criteria should be the **Instance ID** of the instance involved in blockchain activity. To learn more about creating suppression rules see Suppression rules (p. 94).

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

CryptoCurrency:EC2/BitcoinTool.B!DNS

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

Default severity: High

- **Data source:** DNS logs

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 that can be exchanged for other currencies, products, and services. Bitcoin is a reward for bitcoin-mining and is highly sought after by threat actors.

Remediation recommendations:

If you use this EC2 instance to mine or manage cryptocurrency, or this instance is otherwise involved in blockchain activity, this finding could be expected activity for your environment. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `CryptoCurrency:EC2/BitcoinTool.B!DNS`. The second filter criteria should be the **Instance ID** of the instance involved in blockchain activity. To learn more about creating suppression rules see Suppression rules (p. 94).

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Impact:EC2/AbusedDomainRequest.Reputation

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

Default severity: Medium

- **Data source:** DNS logs

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. This model evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Impact:EC2/BitcoinDomainRequest.Reputation

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

Default severity: High

- **Data source:** DNS logs

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 that can be exchanged for other currencies, products, and services. Bitcoin is a reward for bitcoin-mining and is highly sought after by threat actors.

Low reputation domains are based on a reputation score model. This model evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

Remediation recommendations:

If you use this EC2 instance to mine or manage cryptocurrency, or this instance is otherwise involved in blockchain activity, this finding could represent expected activity for your environment. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `Impact:EC2/BitcoinDomainRequest.Reputation`. The second filter criteria should be the **Instance ID** of the instance involved in blockchain activity. To learn more about creating suppression rules see Suppression rules (p. 94).

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Impact:EC2/MaliciousDomainRequest.Reputation

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

Default severity: High

- **Data source:** DNS logs

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. This model evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Impact:EC2/PortSweep

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

Default severity: High

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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. 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. This model evaluates and ranks the characteristics of a domain to determine its likelihood of being malicious.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Impact:EC2/WinRMBruteForce

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

Default severity: Low*

> **Note**
> 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.

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Recon:EC2/PortProbeEMRUnprotectedPort

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

Default severity: High

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

You should block open access to ports on clusters from the internet and restrict access only to specific IP addresses that require access to these ports. For more information see, Security Groups for EMR Clusters.

Recon:EC2/PortProbeUnprotectedPort

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

Default severity: Low*

> **Note**
> 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.

- **Data source:** VPC Flow Logs

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 Authorizing Inbound Traffic for Your Linux Instances. To restrict access to port 3389 on Windows, see Authorizing Inbound Traffic for Your Windows Instances.

Remediation recommendations:

There may be cases in which instances are intentionally exposed, for example if they are hosting web servers. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `Recon:EC2/PortProbeUnprotectedPort`. The second

filter criteria should match the instance or instances that serve as a bastion host. You can use either the **Instance image ID** attribute or the **Tag** value attribute, depending on which criteria is identifiable with the instances that host these tools. For more information about creating suppression rules see Suppression rules (p. 94).

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Recon:EC2/Portscan

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

Default severity: Medium

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

This finding can be a false positive when vulnerability assessment applications are deployed on EC2 instances in your environment because these applications conduct port scans to alert you about misconfigured open ports. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `Recon:EC2/Portscan`. The second filter criteria should match the instance or instances that host these vulnerability assessment tools. You can use either the **Instance image ID** attribute or the **Tag** value attribute depending on which criteria are identifiable with the instances that host these tools. For more information about creating suppression rules see Suppression rules (p. 94).

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

This finding informs you 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 sink hole). 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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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.

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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.

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

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Trojan:EC2/DNSDataExfiltration

An EC2 instance is exfiltrating data through DNS queries.

Default severity: High

- **Data source:** DNS logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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: High

- **Data source:** DNS logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

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

- **Data source:** DNS logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Trojan:EC2/PhishingDomainRequest!DNS

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

Default severity: High

- **Data source:** DNS logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

UnauthorizedAccess:EC2/MaliciousIPCaller.Custom

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

Default severity: Medium

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

UnauthorizedAccess:EC2/MetadataDNSRebind

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

Default severity: High

- **Data source:** DNS logs

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.

Remediation recommendations:

In response to this finding, you should evaluate if there is a vulnerable application running on the EC2 instance, or if someone used a browser to access the domain identified in the finding. If the root cause is a vulnerable application, you should fix the vulnerability. If someone browsed the identified domain, you should block the domain or prevent users from accessing it. If you determine this finding was related to either case above, you should revoke the session associated with the EC2 instance.

Some AWS customers intentionally map the metadata IP address to a domain name on their authoritative DNS servers. If this is the case in your environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `UnauthorizedAccess:EC2/MetaDataDNSRebind`. The second filter criteria should be **DNS request domain** and the value should match the domain you have mapped to the metadata IP address (169.254.169.254). For more information on creating suppression rules see Suppression rules (p. 94).

UnauthorizedAccess:EC2/RDPBruteForce

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

Default severity: Low*

> **Note**
> 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.

- **Data source:** VPC Flow Logs

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 unauthorized access to your AWS resources.

Remediation recommendations:

If your instance's **Resource Role** is `ACTOR`, this indicates your instance has been used to perform RDP brute force attacks. Unless this instance has a legitimate reason to be contacting the IP address listed as the `Target`, it is recommended that you assume your instance has been compromised and take the actions listed in Remediating a compromised EC2 instance (p. 116).

If your instance's **Resource Role** is `TARGET`, this finding can be remediated by securing your RDP port to only trusted IPs through Security Groups, ACLs, or firewalls. For more information see Tips for securing your EC2 instances (Linux).

UnauthorizedAccess:EC2/SSHBruteForce

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

Default severity: Low*

> **Note**
> 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.

- **Data source:** VPC Flow Logs

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 unauthorized access to your AWS resources.

> **Note**
> 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.

Remediation recommendations:

If the target of the brute force attempt is a bastion host, this may represent expected behavior for your AWS environment. If this is the case, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `UnauthorizedAccess:EC2/SSHBruteForce`. The second filter criteria should match the instance or instances that serve as a bastion host. You can use either the **Instance image ID** attribute or the **Tag** value attribute depending on which criteria is identifiable with the instances that host these tools. For more information about creating suppression rules see Suppression rules (p. 94).

If this activity is not expected for your environment and your instance's **Resource Role** is `TARGET`, this finding can be remediated by securing your SSH port to only trusted IPs through Security Groups, ACLs, or firewalls. For more information, see Tips for securing your EC2 instances (Linux).

If your instance's **Resource Role** is `ACTOR`, this indicates the instance has been used to perform SSH brute force attacks. Unless this instance has a legitimate reason to be contacting the IP address listed as the `Target`, it is recommended that you assume your instance has been compromised and take the actions listed in Remediating a compromised EC2 instance (p. 116).

UnauthorizedAccess:EC2/TorClient

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

Default severity: High

- **Data source:** VPC Flow Logs

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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

UnauthorizedAccess:EC2/TorRelay

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

Default severity: High

- **Data source:** VPC Flow Logs

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.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

GuardDuty S3 finding types

The following findings are specific to S3 resources and will have a **Resource Type** of `S3Bucket` if the data source is **S3 CloudTrail data events**, or `AccessKey` if the data source is **CloudTrail management events**. The severity and details of the findings will differ based on the finding type and the permission associated with the bucket.

The findings listed here include the data sources and models used to generate that finding type. For more information data sources and models see How Amazon GuardDuty uses its data sources (p. 10).

> **Important**
> Findings with a data source of **S3 CloudTrail data events** are only generated if you have S3 protection enabled for GuardDuty. S3 protection is enabled by default in all accounts created after July 31, 2020. For information on how to enable or disable S3 protection see Amazon S3 protection in Amazon GuardDuty (p. 141)

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 (p. 116).

Topics

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

- **Data source:** S3 CloudTrail data events

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`.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Discovery:S3/MaliciousIPCaller.Custom

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

Default severity: High

- **Data source:** S3 CloudTrail data events

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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Discovery:S3/TorIPCaller

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

Default severity: Medium

- **Data source:** S3 CloudTrail data events

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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

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

- **Data source:** S3 CloudTrail data events

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`.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Exfiltration:S3/ObjectRead.Unusual

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

Default severity: Medium*

Note
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.

- **Data source:** S3 CloudTraildata events

This finding informs you that a 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 and 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

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

- **Data source:** S3 CloudTrail data events

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 manipulate, interrupt, or destroy data within your AWS environment. Examples include, `PutObject` or `PutObjectAcl`.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

PenTest:S3/KaliLinux

An S3 API was invoked from a Kali Linux machine.

Default severity: Medium

- **Data source:** S3 CloudTrail data events

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

PenTest:S3/ParrotLinux

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

Default severity: Medium

- **Data source:** S3 CloudTrail data events

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

PenTest:S3/PentooLinux

An S3 API was invoked from a Pentoo Linux machine.

Default severity: Medium

- **Data source:** S3 CloudTrail data events

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Policy:S3/AccountBlockPublicAccessDisabled

An IAM entity invoked an API used to disable S3 Block Public Access on an account.

Default severity: Low

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

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

- **Data source:** CloudTrail management events

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, uses automated reasoning powered by Zelkova, to determine if the bucket is publicly accessible.

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Policy:S3/BucketBlockPublicAccessDisabled

An IAM entity invoked an API used to disable S3 Block Public Access on a bucket.

Default severity: Low

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

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

- **Data source:** CloudTrail management events

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, uses automated reasoning powered by Zelkova, to determine if the bucket is publicly accessible.

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Stealth:S3/ServerAccessLoggingDisabled

S3 server access logging was disabled for a bucket.

Default severity: Low

- **Data source:** CloudTrail management events

This finding informs you that S3 server access logging is disabled for a bucket within your AWS environment. If disabled, no web request logs are created for any attempts to access the identified S3 bucket, however, S3 management API calls to the bucket, such as DeleteBucket, are still tracked. If S3 data event logging is enabled through CloudTrail for this bucket, web requests for objects within the bucket will still be tracked. Disabling logging is a technique used by unauthorized users in order to evade detection. To learn more about S3 logs, see S3 Server Access Logging and S3 Logging Options .

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

UnauthorizedAccess:S3/MaliciousIPCaller.Custom

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

Default severity: High

- **Data source:** S3 CloudTrail data events

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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

UnauthorizedAccess:S3/TorIPCaller

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

Default severity: High

- **Data source:** S3 CloudTrail data events

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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

GuardDuty 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.

The findings listed here include the data sources and models used to generate that finding type. For more information data sources and models see How Amazon GuardDuty uses its data sources (p. 10).

For all IAM related findings, we recommend that you examine the entity in question and ensure that their permissions follow the best practice of least privilege. If the activity is unexpected, the credentials may be compromised. See actions detailed in Remediating compromised AWS credentials (p. 118).

Topics

CredentialAccess:IAMUser/AnomalousBehavior

An API used to gain access to an AWS environment was invoked in an anomalous way.

Default severity: Medium

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with the credential access stage of an attack when an adversary is attempting to collect passwords, user names, and access keys for your environment. The APIs in this category are `GetPasswordData`, `GetSecretValue`, and `GenerateDbAuthToken`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API

that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

DefenseEvasion:IAMUser/AnomalousBehavior

An API used to evade defensive measures was invoked in an anomalous way.

Default severity: Medium

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with defense evasion tactics where an adversary is trying to cover their tracks and avoid detection. APIs in this category are typically delete, disable, or stop operations, such as, `DeleteFlowLogs`, `DisableAlarmActions`, or `StopLogging`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Discovery:IAMUser/AnomalousBehavior

An API commonly used to discover resources was invoked in an anomalous way.

Default severity: Low

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with the discovery stage of an attack when an adversary is gathering information to determine if your AWS environment is susceptible to a broader attack. APIs in this category are typically get, describe, or list operations, such as, `DescribeInstances`, `GetRolePolicy`, or `ListAccessKeys`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are

associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Exfiltration:IAMUser/AnomalousBehavior

An API commonly used to collect data from an AWS environment was invoked in an anomalous way.

Default severity: High

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with exfiltration tactics where an adversary is trying to collect data from your network using packaging and encryption to avoid detection. APIs for this finding type are management (control-plane) operations only and are typically related to S3, snapshots, and databases, such as, `PutBucketReplication`, `CreateSnapshot`, or `RestoreDBInstanceFromDBSnapshot`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Impact:IAMUser/AnomalousBehavior

An API commonly used to tamper with data or processes in an AWS environment was invoked in an anomalous way.

Default severity: High

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with impact tactics where an adversary is trying to disrupt operations and manipulate, interrupt, or destroy data in your account. APIs for this finding type are typically delete, update, or put operations, such as, `DeleteSecurityGroup`, `UpdateUser`, or `PutBucketPolicy`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

InitialAccess:IAMUser/AnomalousBehavior

An API commonly used to gain unauthorized access to an AWS environment was invoked in an anomalous way.

Default severity: Medium

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with the initial access stage of an attack when an adversary is attempting to establish access to your environment. APIs in this category are typically get token, or session operations, such as, `GetFederationToken`, `StartSession`, or `GetAuthorizationToken`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

PenTest:IAMUser/KaliLinux

An API was invoked from a Kali Linux EC2 machine.

Default severity: Medium

- **Data source:** CloudTrail management event

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

PenTest:IAMUser/ParrotLinux

An API was invoked from a Parrot Security Linux machine.

Default severity: Medium

- **Data source:** CloudTrail management event

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

PenTest:IAMUser/PentooLinux

An API was invoked from a Pentoo Linux machine.

Default severity: Medium

- **Data source:** CloudTrail management event

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 unauthorized access to your AWS environment.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Persistence:IAMUser/AnomalousBehavior

An API commonly used to maintain unauthorized access to an AWS environment was invoked in an anomalous way.

Default severity: Medium

- **Data source:** CloudTrail management event

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with persistence tactics where an adversary has gained access to your environment and is attempting to maintain that access. APIs in this category are typically create, import, or modify operations, such as, `CreateAccessKey`, `ImportKeyPair`, or `ModifyInstanceAttribute`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Policy:IAMUser/RootCredentialUsage

An API was invoked using root credentials.

Default severity: Low

- **Data source:** CloudTrail management events or CloudTrail data events

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 AWS STS is not supported, IAM user credentials are recommended. For more information, see IAM Best Practices

> **Note**
> 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 S3 protection.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

PrivilegeEscalation:IAMUser/AnomalousBehavior

An API commonly used to obtain high-level permissions to an AWS environment was invoked in an anomalous way.

Default severity: Medium

- **Data source:** CloudTrail management events

This finding informs you that an anomalous API request was observed in your account. This finding may include a single API or a series of related API requests made in proximity by a single user identity. The API observed is commonly associated with privilege escalation tactics where an adversary is attempting to gain higher-level permissions to an environment . APIs in this category typically involve operations that change IAM policies, roles, and users, such as, `AssociateIamInstanceProfile`, `AddUserToGroup`, or `PutUserPolicy`.

This API request was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the API request, such as, the user that made the request, the location the request was made from, and the specific API that was requested. Details on which factors of the API request are unusual for the user identity that invoked the request can be found in the finding details.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Recon:IAMUser/MaliciousIPCaller

An API was invoked from a known malicious IP address.

Default severity: Medium

- **Data source:** CloudTrail management events

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 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Recon:IAMUser/MaliciousIPCaller.Custom

An API was invoked from a known malicious IP address.

Default severity: Medium

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Recon:IAMUser/TorIPCaller

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

Default severity: Medium

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Stealth:IAMUser/CloudTrailLoggingDisabled

AWS CloudTrail logging was disabled.

Default severity: Low

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Stealth:IAMUser/PasswordPolicyChange

Account password policy was weakened.

Default severity: Low*

> **Note**
> This finding's severity can be Low, Medium, or High depending on the severity of the changes made to password policy.

- **Data source:** CloudTrail management events

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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B

Multiple worldwide successful console logins were observed.

Default severity: Medium

- **Data source:** CloudTrail management events

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 unauthorized access to your AWS resources.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS

Credentials that were created exclusively for an EC2 instance through an Instance launch role are being used from another account within AWS.

Default severity: High*

> **Note**
> This finding's default severity is High. However, if the API was invoked by an account affiliated with your AWS environment, the severity is Medium.

- **Data source:** CloudTrail management events or S3 data events

This finding informs you when your EC2 instance credentials are used to invoke APIs from an IP address that is owned by a different AWS account than the one that the associated EC2 instance is running in.

AWS does not recommend redistributing temporary credentials outside of the entity that created them (for example, AWS applications, EC2, or Lambda). However, authorized users can export credentials from their EC2 instances to make legitimate API calls. If the `remoteAccountDetails.Affiliated` field is `True` the API was invoked from an account associated with your AWS environment. To rule out a potential attack and verify the legitimacy of the activity, contact the IAM user to whom these credentials are assigned.

Remediation recommendations:

In response to this finding you can use the following workflow to determine a course of action:

1. Identify the remote account involved from the `service.action.awsApiCallAction.remoteAccountDetails.accountId` field.
2. Next determine if that account is affiliated with your GuardDuty environment from the `service.action.awsApiCallAction.remoteAccountDetails.affiliated` field.
3. If the account is affiliated, contact the remote account owner, and the owner of the EC2 instance credentials to investigate.
4. If the account is not affiliated, first evaluate is that account is associated with your organization but is not a part of your GuardDuty multi-account set up, or if GuardDuty has not yet been enabled in the account. Otherwise contact the owner of the EC2 credentials to determine if there is a use case for a remote account to use these credentials.
5. If the owner of the credentials does not recognize the remote account the credentials may have been compromised by a threat actor operating within AWS. You should take the steps recommended in Remediating a compromised EC2 instance (p. 116) to secure your environment. Additionally you can submit an abuse report to the AWS Trust and Safety team to begin an investigation into the remote account. When submitting your report to AWS Trust and Safety please include the full JSON details of the finding.

UnauthorizedAccess:IAMUser/ InstanceCredentialExfiltration.OutsideAWS

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

- **Data source:** CloudTrail management events or S3 data events

This finding informs you that a host outside of AWS has attempted to run AWS API operations 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, authorized 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, validate if the use of instance credentials from the remote IP in the finding is expected.

Remediation recommendations:

This finding is generated when networking is configured to route internet traffic such that it egresses from an on-premises gateway rather than from a VPC Internet Gateway (IGW). Common configurations, such as using AWS Outposts, or VPC VPN connections, can result in traffic routed this way. If this is expected behavior, we recommend that you use suppression rules and create a rule that consists of two filter criteria. The first criteria is **finding type**, which should be `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS`. The second filter criteria is **API caller IPv4 Address** with the IP address or CIDR range of your on-premises internet gateway. To learn more about creating suppression rules see Suppression rules (p. 94).

Note
If GuardDuty observes continued activity from an external source its machine learning model will identify this as expected behavior and stop generating this finding for activity from that source. GuardDuty will continue to generate findings for new behavior from other sources, and will reevaluate learned sources as behavior changes over time.

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/MaliciousIPCaller

An API was invoked from a known malicious IP address.

Default severity: Medium

- **Data source:** CloudTrail management events

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 unauthorized access to AWS resources within your environment.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom

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

Default severity: Medium

- **Data source:** CloudTrail management events

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. This can indicate unauthorized access to AWS resources within your environment.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/TorIPCaller

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

Default severity: Medium

- **Data source:** CloudTrail management events

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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

GuardDuty Kubernetes finding types

The following findings are specific to Kubernetes resources and have a **Resource type** of `EKScluster`. The severity and details of the findings differ based on finding type.

For all Kubernetes type findings we recommend that you examine the resource in question to determine if the activity is expected or potentially malicious. For guidance on remediating a compromised Kubernetes resource identified by a GuardDuty finding, see Remediating Kubernetes security issues discovered by GuardDuty (p. 118).

Topics

Note
Before Kubernetes version 1.14, the `system:unauthenticated` group was associated to `system:discovery` and `system:basic-user` **ClusterRoles** by default. This association may allow unintended access from anonymous users. Cluster updates do not revoke these permissions. Even if you updated your cluster to version 1.14 or higher, these permissions may still be enabled. We recommend that you disassociate these permissions from the `system:unauthenticated` group. For guidance on revoking these permissions, see Review and revoke unnecessary anonymous access in the Amazon EKS best practice guide.

CredentialAccess:Kubernetes/MaliciousIPCaller

An API commonly used to access credentials or secrets in a Kubernetes cluster was invoked from a known malicious IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was invoked from an IP address that is associated with known malicious activity. The API observed is commonly associated with the credential access tactics where an adversary is attempting to collect passwords, user names, and access keys for your Kubernetes cluster.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

CredentialAccess:Kubernetes/MaliciousIPCaller.Custom

An API commonly used to access credentials or secrets in a Kubernetes cluster was invoked from an IP address on a custom threat list.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation 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. The API observed is commonly associated with the credential access tactics where an adversary is attempting to collect passwords, user names, and access keys for your Kubernetes cluster.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

CredentialAccess:Kubernetes/SuccessfulAnonymousAccess

An API commonly used to access credentials or secrets in a Kubernetes cluster was invoked by an unauthenticated user.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was successfully invoked by the `system:anonymous` user. API calls made by `system:anonymous` are unauthenticated. The observed API is commonly associated with the credential access tactics where an adversary is attempting to collect passwords, user names, and access keys for your Kubernetes cluster. This activity indicates that anonymous or unauthenticated access is permitted on the API action reported in the finding and may be permitted on other actions. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user on your cluster and ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, you should revoke access of the user and reverse any changes made by an adversary to your cluster. See Review and revoke unnecessary anonymous access for guidance.

CredentialAccess:Kubernetes/TorIPCaller

An API commonly used to access credentials or secrets in a Kubernetes cluster was invoked from a Tor exit node IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API was invoked from a Tor exit node IP address. The API observed is commonly associated with the credential access tactics where an adversary is attempting to collect passwords, user names, and access keys for your environment. 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 unauthorized access to your Kubernetes cluster resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or

malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

DefenseEvasion:Kubernetes/MaliciousIPCaller

An API commonly used to evade defensive measures was invoked from a known malicious IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was invoked from an IP address that is associated with known malicious activity. The API observed is commonly associated with defense evasion tactics where an adversary is trying to hide their actions to avoid detection.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

DefenseEvasion:Kubernetes/MaliciousIPCaller.Custom

An API commonly used to evade defensive measures was invoked from an IP address on a custom threat list.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation 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. The API observed is commonly associated with defense evasion tactics where an adversary is trying to hide their actions to avoid detection.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

DefenseEvasion:Kubernetes/SuccessfulAnonymousAccess

An API commonly used to evade defensive measures was invoked by an unauthenticated user.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was successfully invoked by the `system:anonymous` user. API calls made by `system:anonymous` are unauthenticated. The observed API is commonly associated with defense evasion tactics where an adversary is trying to hide their actions to avoid detection. This activity indicates that anonymous or unauthenticated access is permitted on the API action reported in the finding and may be permitted on other actions. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user on your cluster and ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, you should revoke access of the user and reverse any changes made by an adversary to your cluster. See Review and revoke unnecessary anonymous access for guidance.

DefenseEvasion:Kubernetes/TorIPCaller

An API commonly used to evade defensive measures was invoked from a Tor exit node IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API was invoked from a Tor exit node IP address. The API observed is commonly associated with defense evasion tactics where an adversary is trying to hide their actions to avoid detection. 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 unauthorized access to your Kubernetes cluster with the intent of hiding the adversary's true identity.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Discovery:Kubernetes/MaliciousIPCaller

An API commonly used to discover resources in a Kubernetes cluster was invoked from an IP address on a custom threat list.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was invoked from an IP address that is associated with known malicious activity. The observed API is commonly used with the discovery stage of an attack wherein an attacker is gathering information to determine if your Kubernetes cluster is susceptible to a broader attack.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Discovery:Kubernetes/MaliciousIPCaller.Custom

An API commonly used to discover resources in a Kuberentes cluster was invoked from an IP address on a custom threat list.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API 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. The observed API is commonly used with the discovery stage of an attack wherein an attacker is gathering information to determine if your Kubernetes cluster is susceptible to a broader attack.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Discovery:Kubernetes/SuccessfulAnonymousAccess

An API commonly used to discover resources in a Kubernetes cluster was invoked by an unauthenticated user.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was successfully invoked by the `system:anonymous` user. API calls made by `system:anonymous` are unauthenticated. The observed API is commonly associated with the discovery stage of an attack when an adversary is gathering information on your Kubernetes cluster. This activity indicates that anonymous or unauthenticated access is permitted on the API action reported in the finding and may be permitted on other actions. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user on your cluster and ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, you should revoke access of the user and reverse any changes made by an adversary to your cluster. See Review and revoke unnecessary anonymous access for guidance.

Discovery:Kubernetes/TorIPCaller

An API commonly used to discover resources in a Kubernetes cluster was invoked from a Tor exit node IP address.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API was invoked from a Tor exit node IP address. The observed API is commonly used with the discovery stage of an attack wherein an attacker is gathering information to determine if your Kubernetes cluster 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 unauthorized access to your Kubernetes cluster with the intent of hiding the adversary's true identity.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Execution:Kubernetes/ExecInKubeSystemPod

A command was executed inside a pod within the `kube-system` namespace

Default severity: Medium

- **Data source:** Kubernetes audit logs

This findings informs you that a command was executed in a pod within the `kube-system` namespace using **Kubernetes exec API**. `kube-system` namespace is a default namespaces, which is primarily used for system level components such as `kube-dns` and `kube-proxy`. It is very uncommon to execute commands inside pods or containers under `kube-system` namespace and may indicate suspicious activity.

Remediation recommendations:

If the execution of this command is unexpected, the credentials of the user identity used to execute the command may be compromised. Revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Impact:Kubernetes/MaliciousIPCaller

An API commonly used to tamper with resources in a Kubernetes cluster was invoked from a known malicious IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an 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.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Impact:Kubernetes/MaliciousIPCaller.Custom

An API commonly used to tamper with resources in a Kubernetes cluster was invoked from an IP address on a custom threat list.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation 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. The observed API is commonly associated with impact tactics where an adversary is trying to manipulate, interrupt, or destroy data within your AWS environment.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Impact:Kubernetes/SuccessfulAnonymousAccess

An API commonly used to tamper with resources in a Kubernetes cluster was invoked by an unauthenticated user.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was successfully invoked by the `system:anonymous` user. API calls made by `system:anonymous` are unauthenticated. The observed API is commonly associated with the impact stage of an attack when an adversary is tampering with resources in your cluster. This activity indicates that anonymous or unauthenticated access is permitted on the API action reported in the finding and may be permitted on other actions. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user on your cluster and ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, you should revoke access of the user and reverse any changes made by an adversary to your cluster. See Review and revoke unnecessary anonymous access for guidance.

Impact:Kubernetes/TorIPCaller

An API commonly used to tamper with resources in a Kubernetes cluster was invoked from a Tor exit node IP address.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API was invoked from a Tor exit node IP address. The API observed is commonly associated with impact tactics where an adversary is trying to manipulate, interrupt, or destroy data within your AWS environment. 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 unauthorized access to your Kubernetes cluster with the intent of hiding the adversary's true identity.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Persistence:Kubernetes/ContainerWithSensitiveMount

A container was launched with a sensitive external host path mounted inside.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that a container was launched with a configuration that included a sensitive host path with write access in the `volumeMounts` section. This makes the sensitive host path accessible and writable from inside the container. This technique is commonly used by adversaries to gain access to the host's filesystem.

Remediation recommendations:

If this container launch is unexpected, the credentials of the user identity used to launch the container may be compromised. Revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings for guidance. If this container launch is expected, it's recommended that you use a suppression rule consisting of a filter criteria based on the `resource.kubernetesDetails.kubernetesWorkloadDetails.containers.imagePrefix` field. In the filter criteria the `imagePrefix` field should be same as the `imagePrefix` specified in the finding. To learn more about creating suppression rules see Suppression rules.

Persistence:Kubernetes/MaliciousIPCaller

An API commonly used to obtain persistent access to a Kubernetes cluster was invoked from a known malicious IP address.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was invoked from an IP address that is associated with known malicious activity. The API observed is commonly associated with persistence tactics where an adversary has gained access to your Kubernetes cluster and is attempting to maintain that access.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Persistence:Kubernetes/MaliciousIPCaller.Custom

An API commonly used to obtain persistent access to a Kubernetes cluster was invoked from an IP address on a custom threat list.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation 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. The API observed is commonly associated with persistence

tactics where an adversary has gained access to your Kubernetes cluster and is attempting to maintain that access.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Persistence:Kubernetes/SuccessfulAnonymousAccess

An API commonly used to obtain high-level permissions to a Kubernetes cluster was invoked by an unauthenticated user.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that an API operation was successfully invoked by the `system:anonymous` user. API calls made by `system:anonymous` are unauthenticated. The observed API is commonly associated with the persistence tactics where an adversary has gained access to your cluster and is attempting to maintain that access. This activity indicates that anonymous or unauthenticated access is permitted on the API action reported in the finding and may be permitted on other actions. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user on your cluster and ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, you should revoke access of the user and reverse any changes made by an adversary to your cluster. See Review and revoke unnecessary anonymous access for guidance.

Persistence:Kubernetes/TorIPCaller

An API commonly used to obtain persistent access to a Kubernetes cluster was invoked from a Tor exit node IP address.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that an API was invoked from a Tor exit node IP address. The API observed is commonly associated with persistence tactics where an adversary has gained access to your Kubernetes cluster and is attempting to maintain that access. 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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If the user reported in the finding under the `kubernetesUserDetails` section is `system:anonymous`, investigate why the anonymous user was permitted to invoke the API and revoke the permissions if needed. If the user is an authenticated user, investigate to determine if the activity was legitimate or malicious. If the activity was malicious revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Policy:Kubernetes/AdminAccessToDefaultServiceAccount

The default service account was granted admin privileges on a Kubernetes cluster.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that the default service account for a namespace in your Kubernetes cluster was granted admin privileges. Kubernetes creates a default service account for all the namespaces in the cluster. It automatically assigns the default service account as an identity to pods that have not been explicitly associated to another service account. If the default service account has admin privileges, it may result in pods being unintentionally launched with admin privileges. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised.

Remediation recommendations:

You should not use the default service account to grant permissions to pods. Instead you should create a dedicated service account for each workload and grant permission to that account on a needs basis. To fix this issue, you should create dedicated service accounts for all your pods and workloads and update the pods and workloads to migrate from the default service account to their dedicated accounts. Then you should remove the admin permission from the default service account. See Remediating Kubernetes findings (p. 119) for additional guidance and resources.

Policy:Kubernetes/AnonymousAccessGranted

The `system:anonymous` user was granted API permission on a Kubernetes cluster.

Default severity: High

- **Data source:** Kubernetes audit logs

This finding informs you that a user on your Kubernetes cluster successfully created a `ClusterRoleBinding` or `RoleBinding` to bind the user `system:anonymous` to a role. This enables unauthenticated access to the API operations permitted by the role. If this behavior is not expected, it may indicate a configuration mistake or that your credentials are compromised

Remediation recommendations:

You should examine the permissions that have been granted to the `system:anonymous` user or `system:unauthenticated` group on your cluster and revoke unnecessary anonymous access. See Review and revoke unnecessary anonymous access for guidance. If the permissions were granted

maliciously, you should revoke access of the user that granted the permissions and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings for guidance.

Policy:Kubernetes/ExposedDashboard

The dashboard for a Kubernetes cluster was exposed to the internet

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that Kubernetes dashboard for your cluster was exposed to the internet by a Load Balancer service. An exposed dashboard makes the management interface of your cluster accessible from the internet and allows adversaries to exploit any authentication and access control gaps that may be present.

Remediation recommendations:

You should ensure that strong authentication and authorization is enforced on Kubernetes Dashboard. You should also implement network access control to restrict access to the dashboard from specific IP addresses.

Policy:Kubernetes/KubeflowDashboardExposed

The **Kubeflow** dashboard for a Kubernetes cluster was exposed to the Internet

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that **Kubeflow** dashboard for your cluster was exposed to the Internet by a Load Balancer service. An exposed **Kubeflow** dashboard makes the management interface of your **Kubeflow** environment accessible from the Internet and allows adversaries to exploit any authentication and access control gaps that may be present.

Remediation recommendations:

You should ensure that strong authentication and authorization is enforced on **Kubeflow** Dashboard. You should also implement network access control to restrict access to the dashboard from specific IP addresses.

PrivilegeEscalation:Kubernetes/PrivilegedContainer

A privileged container with root level access was launched on your Kubernetes cluster.

Default severity: Medium

- **Data source:** Kubernetes audit logs

This finding informs you that a privileged container was launched on your Kubernetes cluster using an image has never before been used to launch privileged containers in your cluster. A privileged container has root level access to the host. Adversaries can launch privileged containers as a privilege escalation tactic to gain access to and then compromise the host.

Remediation recommendations:

If this container launch is unexpected, the credentials of the user identity used to launch the container may be compromised. Revoke access of the user and reverse any changes made by an adversary to your cluster. See Remediating Kubernetes findings (p. 118) for guidance.

Retired finding types

Important
For information about important changes to the GuardDuty finding types, including newly added or retired finding types, see Document history for Amazon GuardDuty (p. 192).

In the current release of GuardDuty, the following finding types are retired (no longer generated). You CANNOT reactivate retired GuardDuty findings types.

Topics

Impact:S3/PermissionsModification.Unusual

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

Default severity: Medium*

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Impact:S3/ObjectDelete.Unusual

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

Default severity: Medium*

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

Discovery:S3/BucketEnumeration.Unusual

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

Default severity: Medium*

> **Note**
> 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.

Remediation recommendations:

If this activity is unexpected for the associated principal it may indicate the credentials have been exposed or your S3 permissions are not restrictive enough, see Remediating a compromised S3 Bucket (p. 116).

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*

> **Note**
> 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> This finding's default severity is Medium. However, if the API is invoked is 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> 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 behavior 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 maximize 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

PrivilegeEscalation:IAMUser/AdministrativePermissions

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

Default severity: Low*

> **Note**
> 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 behavior 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 maximize 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> 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 behavior that is different from the established baseline. This principal has no prior history of invoking this API in this way.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

ResourceConsumption:IAMUser/ComputeResources

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

Default severity: Medium*

> **Note**
> 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

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*

> **Note**
> 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 behavior 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.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:IAMUser/ConsoleLogin

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

Default severity: Medium*

> **Note**
> 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 behavior that is different from the established baseline. This principal has no prior history of login activity using this client application from this specific location.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

UnauthorizedAccess:EC2/TorIPCaller

Your EC2 instance is receiving inbound connections from a Tor exit node.

Default severity: Medium

This finding informs you that an EC2 instance in your AWS environment is receiving inbound connections from a Tor exit node. 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 unauthorized access to your AWS resources with the intent of hiding the attacker's true identity.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Backdoor:EC2/XORDDOS

An EC2 instance is attempting to communicate with an IP address that is associated with XorDDos malware.

Default severity: High

This finding informs you that an EC2 instance in your AWS environment is attempting to communicate with an IP address that is associated with XorDDos malware. This EC2 instance might be compromised. XOR DDoS is Trojan malware that hijacks Linux systems. To gain access to the system, it launches a brute force attack in order to discover the password to Secure Shell (SSH) services on Linux. After SSH credentials are acquired and the login is successful, it uses root privileges to run a script that downloads and installs XOR DDoS. This malware is then used as part of a botnet to launch distributed denial of service (DDoS) attacks against other targets.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

Behavior:IAMUser/InstanceLaunchUnusual

An IAM user launched an EC2 instance of an unusual type.

Default severity: High

This finding informs you that a specific IAM user in your AWS environment is exhibiting behavior that is different from the established baseline. This IAM user has no prior history of launching an EC2 instance of this type. Your IAM user credentials might be compromised.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

CryptoCurrency:EC2/BitcoinTool.A

EC2 instance is communicating with Bitcoin mining pools.

Default severity: High

This finding informs you that an EC2 instance in your AWS environment is communicating with Bitcoin mining pools. In the field of cryptocurrency mining, a mining pool is the pooling of resources by miners who share their processing power over a network to split the reward according to the amount of work they contributed to solving a block. Unless you use this EC2 instance for Bitcoin mining, your EC2 instance might be compromised.

Remediation recommendations:

If this activity is unexpected, your instance is likely compromised, see Remediating a compromised EC2 instance (p. 116).

UnauthorizedAccess:IAMUser/UnusualASNCaller

An API was invoked from an IP address of an unusual network.

Default severity: High

This finding informs you that certain activity was invoked from an IP address of an unusual network. This network was never observed throughout the AWS usage history of the described user. This activity can include a console login, an attempt to launch an EC2 instance, create a new IAM user, modify your AWS privileges, etc. This can indicate unauthorized access to your AWS resources.

Remediation recommendations:

If this activity is unexpected your credentials may be compromised, see Remediating compromised AWS credentials (p. 118).

Findings by resource type

The following pages are broken down by each resource type GuardDuty currently generates findings for. The pages contain detailed information on all finding types for that resources type.

- EC2 finding types (p. 26)
- IAM finding types (p. 51)
- S3 finding types (p. 44)
- Kubernetes finding types (p. 63)

Findings table

The following table lists all finding types by name, resource, data source and severity. A severity listed with an asterisk (*) indicates the finding has variable severities depending the circumstances of the

finding, which are described in the details for that finding. Choose the finding name to open more info about that finding.

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
Backdoor:EC2/C&CActivity.B (p. 27)	EC2	VPC Flow Logs	High
Backdoor:EC2/C&CActivity.B!DNS (p. 28)	EC2	DNS logs	High
Backdoor:EC2/DenialOfService.Dns (p. 28)	EC2	VPC Flow Logs	High
Backdoor:EC2/DenialOfService.Tcp (p. 29)	EC2	VPC Flow Logs	High
Backdoor:EC2/DenialOfService.Udp (p. 29)	EC2	VPC Flow Logs	High
Backdoor:EC2/DenialOfService.UdpOnTcpPorts (p. 30)	EC2	VPC Flow Logs	High
Backdoor:EC2/DenialOfService.UnusualProtocol (p. 30)	EC2	VPC Flow Logs	High
Backdoor:EC2/Spambot (p. 31)	EC2	VPC Flow Logs	Medium
Behavior:EC2/NetworkPortUnusual (p. 31)	EC2	VPC Flow Logs	Medium
Behavior:EC2/TrafficVolumeUnusual (p. 31)	EC2	VPC Flow Logs	Medium
CredentialAccess:IAMUser/AnomalousBehavior (p. 52)	IAM	CloudTrail management event	Medium
CredentialAccess:Kubernetes/MaliciousIPCaller (p. 64)	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/MaliciousIPCaller.Custom (p. 64)	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/SuccessfulAnonymousAccess (p. 65)	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/TorIPCaller (p. 65)	Kubernetes	Kubernetes audit logs	High
CryptoCurrency:EC2/BitcoinTool.B (p. 32)	EC2	VPC Flow Logs	High
CryptoCurrency:EC2/BitcoinTool.B!DNS (p. 32)	EC2	DNS logs	High
DefenseEvasion:IAMUser/AnomalousBehavior (p. 53)	IAM	CloudTrail management event	Medium

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
DefenseEvasion:Kubernetes/MaliciousIPCaller (p. 66)	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/MaliciousIPCaller.Custom (p. 66)	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/SuccessfulAnonymousAccess (p. 66)	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/TorIPCaller (p. 67)	Kubernetes	Kubernetes audit logs	High
Discovery:IAMUser/AnomalousBehavior (p. 53)	IAM	CloudTrail management event	Low
Discovery:Kubernetes/MaliciousIPCaller (p. 67)	Kubernetes	Kubernetes audit logs	Medium
Discovery:Kubernetes/MaliciousIPCaller.Custom (p. 68)	Kubernetes	Kubernetes audit logs	Medium
Discovery:Kubernetes/SuccessfulAnonymousAccess (p. 68)	Kubernetes	Kubernetes audit logs	Medium
Discovery:Kubernetes/TorIPCaller (p. 69)	Kubernetes	Kubernetes audit logs	Medium
Discovery:S3/MaliciousIPCaller (p. 45)	S3	CloudTrail S3 data event	High
Discovery:S3/MaliciousIPCaller.Custom (p. 45)	S3	CloudTrail S3 data event	High
Discovery:S3/TorIPCaller (p. 46)	S3	CloudTrail S3 data event	Medium
Execution:Kubernetes/ExecInKubeSystemPod (p. 69)	Kubernetes	Kubernetes audit logs	Medium
Exfiltration:IAMUser/AnomalousBehavior (p. 54)	IAM	CloudTrail management event	High
Exfiltration:S3/MaliciousIPCaller (p. 46)	S3	CloudTrail S3 data event	High
Exfiltration:S3/ObjectRead.Unusual (p. 46)	S3	S3 CloudTraildata events	Medium*
Impact:EC2/AbusedDomainRequest.Reputation (p. 33)	EC2	DNS logs	Medium
Impact:EC2/BitcoinDomainRequest.Reputation (p. 33)	EC2	DNS logs	High
Impact:EC2/MaliciousDomainRequest.Reputation (p. 34)	EC2	DNS logs	High
Impact:EC2/PortSweep (p. 34)	EC2	VPC Flow Logs	High

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
Impact:EC2/SuspiciousDomainRequest.Reputation (p. 35)	EC2	DNS logs	Low
Impact:EC2/WinRMBruteForce (p. 35)	EC2	VPC Flow Logs	Low*
Impact:IAMUser/AnomalousBehavior (p. 54)	IAM	CloudTrail management event	High
Impact:Kubernetes/MaliciousIPCaller (p. 70)	Kubernetes	Kubernetes audit logs	High
Impact:Kubernetes/MaliciousIPCaller.Custom (p. 70)	Kubernetes	Kubernetes audit logs	High
Impact:Kubernetes/SuccessfulAnonymousAccess (p. 70)	Kubernetes	Kubernetes audit logs	High
Impact:Kubernetes/TorIPCaller (p. 71)	Kubernetes	Kubernetes audit logs	High
Impact:S3/MaliciousIPCaller (p. 47)	S3	CloudTrail S3 data event	High
InitialAccess:IAMUser/AnomalousBehavior (p. 55)	IAM	CloudTrail management event	Medium
PenTest:IAMUser/KaliLinux (p. 55)	IAM	CloudTrail management event	Medium
PenTest:IAMUser/ParrotLinux (p. 56)	IAM	CloudTrail management event	Medium
PenTest:IAMUser/PentooLinux (p. 56)	IAM	CloudTrail management event	Medium
PenTest:S3/KaliLinux (p. 47)	S3	CloudTrail S3 data event	Medium
PenTest:S3/ParrotLinux (p. 48)	S3	CloudTrail S3 data event	Medium
PenTest:S3/PentooLinux (p. 48)	S3	CloudTrail S3 data event	Medium
Persistence:IAMUser/AnomalousBehavior (p. 56)	IAM	CloudTrail management event	Medium
Persistence:Kubernetes/ContainerWithSensitiveMount (p. 71)	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/MaliciousIPCaller (p. 72)	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/MaliciousIPCaller.Custom (p. 72)	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/SuccessfulAnonymousAccess (p. 73)	Kubernetes	Kubernetes audit logs	High

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
Persistence:Kubernetes/TorIPCaller (p. 73)	Kubernetes	Kubernetes audit logs	Medium
Policy:IAMUser/RootCredentialUsage (p. 57)	IAM	CloudTrail management events or CloudTrail data events	Low
Policy:Kubernetes/AdminAccessToDefaultServiceAccount (p. 74)	Kubernetes	Kubernetes audit logs	High
Policy:Kubernetes/AnonymousAccessGranted (p. 74)	Kubernetes	Kubernetes audit logs	High
Policy:Kubernetes/KubeflowDashboardExposed (p. 75)	Kubernetes	Kubernetes audit logs	Medium
Policy:Kubernetes/ExposedDashboard (p. 75)	Kubernetes	Kubernetes audit logs	Medium
Policy:S3/AccountBlockPublicAccessDisabled (p. 48)	S3	CloudTrail management events	Low
Policy:S3/BucketAnonymousAccessGranted (p. 49)	S3	CloudTrail management events	High
Policy:S3/BucketBlockPublicAccessDisabled (p. 49)	S3	CloudTrail management events	Low
Policy:S3/BucketPublicAccessGranted (p. 50)	S3	CloudTrail management events	High
PrivilegeEscalation:IAMUser/AnomalousBehavior (p. 57)	IAM	CloudTrail management events	Medium
PrivilegeEscalation:Kubernetes/PrivilegedContainer (p. 75)	Kubernetes	Kubernetes audit logs	Medium
Recon:EC2/PortProbeEMRUnprotectedPort (p. 36)	EC2	VPC Flow Logs	High
Recon:EC2/PortProbeUnprotectedPort (p. 36)	EC2	VPC Flow Logs	Low*
Recon:EC2/Portscan (p. 37)	EC2	VPC Flow Logs	Medium
Recon:IAMUser/MaliciousIPCaller (p. 58)	IAM	CloudTrail management events	Medium
Recon:IAMUser/MaliciousIPCaller.Custom (p. 58)	IAM	CloudTrail management events	Medium
Recon:IAMUser/TorIPCaller (p. 59)	IAM	CloudTrail management events	Medium
Stealth:IAMUser/CloudTrailLoggingDisabled (p. 59)	IAM	CloudTrail management events	Low

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
Stealth:IAMUser/PasswordPolicyChange (p. 59)	IAM	CloudTrail management event	Low
Stealth:S3/ServerAccessLoggingDisabled (p. 50)	S3	CloudTrail management events	Low
Trojan:EC2/BlackholeTraffic (p. 37)	EC2	VPC Flow Logs	Medium
Trojan:EC2/BlackholeTraffic!DNS (p. 38)	EC2	DNS logs	Medium
Trojan:EC2/DGADomainRequest.B (p. 38)	EC2	DNS logs	High
Trojan:EC2/DGADomainRequest.C!DNS (p. 39)	EC2	DNS logs	High
Trojan:EC2/DNSDataExfiltration (p. 39)	EC2	DNS logs	High
Trojan:EC2/DriveBySourceTraffic!DNS (p. 39)	EC2	DNS logs	High
Trojan:EC2/DropPoint (p. 40)	EC2	VPC Flow Logs	Medium
Trojan:EC2/DropPoint!DNS (p. 40)	EC2	DNS logs	Medium
Trojan:EC2/PhishingDomainRequest!DNS (p. 41)	EC2	DNS logs	High
UnauthorizedAccess:EC2/MaliciousIPCaller.Custom (p. 41)	EC2	VPC Flow Logs	Medium
UnauthorizedAccess:EC2/MetadataDNSRebind (p. 41)	EC2	DNS logs	High
UnauthorizedAccess:EC2/RDPBruteForce (p. 42)	EC2	VPC Flow Logs	Low*
UnauthorizedAccess:EC2/SSHBruteForce (p. 43)	EC2	VPC Flow Logs	Low*
UnauthorizedAccess:EC2/TorClient (p. 43)	EC2	VPC Flow Logs	High
UnauthorizedAccess:EC2/TorRelay (p. 44)	EC2	VPC Flow Logs	High
UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B (p. 60)	IAM	CloudTrail management events	Medium

FINDING TYPE	RESOURCE	DATA SOURCE	SEVERITY
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS (p. 60)	IAM	CloudTrail management event	High*
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS (p. 61)	IAM	CloudTrail management events or S3 data events	High
UnauthorizedAccess:IAMUser/MaliciousIPCaller (p. 62)	IAM	CloudTrail management events	Medium
UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom (p. 62)	IAM	CloudTrail management events	Medium
UnauthorizedAccess:IAMUser/TorIPCaller (p. 62)	IAM	CloudTrail management events	Medium
UnauthorizedAccess:S3/MaliciousIPCaller.Custom (p. 51)	S3	CloudTrail S3 data event	High
UnauthorizedAccess:S3/TorIPCaller (p. 51)	S3	CloudTrail S3 data event	High

Managing Amazon GuardDuty findings

GuardDuty offers several important features to help you sort, store, and manage your findings. These features will help you tailor findings to your specific environment, reduce noise from low value findings, and help you focus on threats to your unique AWS environment. Review the topics on this page to understand how you can use these features to increase the value of GuardDuty's findings.

Topics:

Filtering findings

A finding filter allows you to view findings that match the criteria you specify and filter out any unmatched findings. You can easily create finding filters using the Amazon GuardDuty console, or you can create them with the CreateFilter API using JSON. Review the following sections to understand how to create a filter in the console. To use these filters to automatically archive incoming findings see Suppression rules (p. 94).

Creating filters in the GuardDuty console

Finding filters can be created and tested through the GuardDuty console. You can save filters created through the console for use in suppression rules or future filter operations. A filter is made up of at least one filter criteria, which consists of one filter attribute paired with at least one value.

When you create filters, be aware of the following:

- Filters do not accept wild cards.
- You can specify a minimum of one attribute and up to a maximum of 50 attributes as the criteria for a particular filter.

- When you use the **equal to** or **not equal to** condition to filter on an attribute value, such as Account ID, you can specify a maximum of 50 values.
- Each filter criteria attribute is evaluated as an `AND` operator. Multiple values for the same attribute are evaluated as `AND/OR`.

To filter findings (console)

1. Choose **Add filter criteria** above the displayed list of your GuardDuty findings.
2. In the expanded list of attributes, select the attribute that you want to specify as the criteria for your filter, such as **Account ID** or **Action type**.

   Note
   See the filter attribute table on this page for a list of attributes that you can use to create filter criteria.
3. In the displayed text field, specify a value for each selected attribute and then choose **Apply**.

   Note
   After you apply a filter, you can convert the filter to exclude findings that match the filter by choosing the black dot to the left of the filter name. This effectively creates a "not equals" filter for the selected attribute.
4. To save the specified attributes and their values (filter criteria) as a filter, select **Save**. Enter the filter name and description, and then choose **Done**.

Filter attributes

When you create filters or sort findings using the API operations, you must specify filter criteria in JSON. These filter criteria correlate to a finding's details JSON. The following table contains a list of the console display names for filter attributes and their equivalent JSON field names.

Console field name	JSON field name
Account ID	accountId
Confidence	confidence
Finding ID	id
Region	region
Access Key ID	resource.accessKeyDetails.accessKeyId
Principal ID	resource.accessKeyDetails.principalId
Username	resource.accessKeyDetails.userName
User type	resource.accessKeyDetails.userType
IAM instance profile ID	resource.instanceDetails.iamInstanceProfile.id
Instance ID	resource.instanceDetails.instanceId
Instance Type	resource.instanceDetails.instanceType
Launch Time	resource.instanceDetails.launchTime
Instance image ID	resource.instanceDetails.imageId

Console field name	JSON field name
IPv6 address	resource.instanceDetails.networkInterfaces.ipv6Addresses
Private IPv4 address	resource.instanceDetails.networkInterfaces.privateIpAddresses.pr
Public DNS name	resource.instanceDetails.networkInterfaces.publicDnsName
Public IP	resource.instanceDetails.networkInterfaces.publicIp
Security group ID	resource.instanceDetails.networkInterfaces.securityGroups.group
Security group name	resource.instanceDetails.networkInterfaces.securityGroups.group
Subnet ID	resource.instanceDetails.networkInterfaces.subnetId
VPC ID	resource.instanceDetails.networkInterfaces.vpcId
Outpost ARN	resource.instanceDetails.outpostARN
Tag key	resource.instanceDetails.tags.key
Tag value	resource.instanceDetails.tags.value
Resource type	resource.resourceType
Bucket permissions	resource.s3BucketDetails.publicAccess.effectivePermissions
Bucket name	resource.s3BucketDetails.name
Bucket tag key	resource.s3BucketDetails.tags.key
Bucket tag value	resource.s3BucketDetails.tags.value
Bucket type	resource.s3BucketDetails.type
Action type	service.action.actionType
API called	service.action.awsApiCallAction.api
API caller type	service.action.awsApiCallAction.callerType
API Error Code	service.action.awsApiCallAction.errorCode
API caller city	service.action.awsApiCallAction.remoteIpDetails.city.cityName
API caller country	service.action.awsApiCallAction.remoteIpDetails.country.country
API caller IPv4 address	service.action.awsApiCallAction.remoteIpDetails.ipAddressV4
API caller ASN ID	service.action.awsApiCallAction.remoteIpDetails.organization.asn
API caller ASN name	service.action.awsApiCallAction.remoteIpDetails.organization.asn
API caller service name	service.action.awsApiCallAction.serviceName
DNS request domain	service.action.dnsRequestAction.domain
Network connection blocked	service.action.networkConnectionAction.blocked
Network connection direction	service.action.networkConnectionAction.connectionDirection
Network connection local port	service.action.networkConnectionAction.localPortDetails.port

Console field name	JSON field name
Network connection protocol	service.action.networkConnectionAction.protocol
Network connection city	service.action.networkConnectionAction.remoteIpDetails.city.city
Network connection country	service.action.networkConnectionAction.remoteIpDetails.country
Network connection remote IPv4 address	service.action.networkConnectionAction.remoteIpDetails.ipAddre
Network connection remote IP ASN ID	service.action.networkConnectionAction.remoteIpDetails.organiz
Network connection remote IP ASN name	service.action.networkConnectionAction.remoteIpDetails.organiz
Network connection remote port	service.action.networkConnectionAction.remotePortDetails.port
Threat list name	service.additionalInfo.threatListName
Archived	service.archived
Local IP	service.localIpDetails.ipAddressV4
Resource role	service.resourceRole
Severity	severity
Finding type	type
Updated at	updatedAt

Suppression rules

A suppression rule is a set of criteria, consisting of a filter attribute paired with a value, used to filter findings by automatically archiving new findings that match the specified criteria. Suppression rules can be used to filter low-value findings, false positive findings, or threats you do not intend to act on, to make it easier to recognize the security threats with the most impact to your environment.

After you create a suppression rule, new findings that match the criteria defined in the rule are automatically archived as long as the suppression rule is in place. You can use an existing filter to create a suppression rule or create a suppression rule from a new filter you define. You can configure suppression rules to suppress entire finding types, or define more granular filter criteria to suppress only specific instances of a particular finding type. Your suppression rules can be edited at any time.

Suppressed findings are not sent to AWS Security Hub, Amazon S3, Detective, or CloudWatch, reducing finding noise level if you consume GuardDuty findings via Security Hub, a third-party SIEM, or other alerting and ticketing applications.

GuardDuty continues to generate findings even when they match your suppression rules, however, those findings are automatically marked as **archived**. The archived finding is stored in GuardDuty for 90-days and can be viewed at any time during that period. You can view suppressed findings in the GuardDuty console by selecting **Archived** from the findings table, or through the GuardDuty API using the ListFindings API with a `findingCriteria` criterion of `service.archived` equal to true.

> **Note**
> In a multi-account environment only the GuardDuty administrator can create suppression rules.

Common use cases for suppression rules and examples

The following finding types have common use cases for applying suppression rules, select the finding name to learn more about that finding, or review the info to build a suppression rule for that finding type from the console.

Important
GuardDuty recommends that you build suppression rules reactively and only for findings you have repeatedly identified false positives for.

- UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS (p. 61) – Use a suppression rule to automatically archive findings generated when VPC networking is configured to route internet traffic such that it egresses from an on-premises gateway rather than from a VPC Internet Gateway.

  This finding is generated when networking is configured to route internet traffic such that it egresses from an on-premises gateway rather than from a VPC Internet Gateway (IGW). Common configurations, such as using AWS Outposts, or VPC VPN connections, can result in traffic routed this way. If this is expected behavior, it's recommended that you use suppression rules in and create a rule that consists of two filter criteria. The first criteria is **finding type**, which should be `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration`. The second filter criteria is **API caller IPv4 address** with the IP address or CIDR range of your on-premises internet gateway. The example below represents the filter you would use to suppress this finding type based on API caller IP address.

  ```
  Finding type: UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration API caller IPv4
   address: 198.51.100.6
  ```

  Note
  To include multiple API caller IPs you can add a new API Caller IPv4 address filter for each.

- Recon:EC2/Portscan (p. 37) – Use a suppression rule to automatically archive findings when using a vulnerability assessment application.

  The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `Recon:EC2/Portscan`. The second filter criteria should match the instance or instances that host these vulnerability assessment tools. You can use either the **Instance image ID** attribute or the **Tag** value attribute depending on which criteria are identifiable with the instances that host these tools. The example below represents the filter you would use to suppress this finding type based on instances with a certain AMI.

  ```
  Finding type: Recon:EC2/Portscan Instance image ID: ami-999999999
  ```

- UnauthorizedAccess:EC2/SSHBruteForce (p. 43) – Use a suppression rule to automatically archive findings when it is targeted to bastion instances.

  If the target of the brute force attempt is a bastion host, this may represent expected behavior for your AWS environment. If this is the case, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `UnauthorizedAccess:EC2/SSHBruteForce`. The second filter criteria should match the instance or instances that serve as a bastion host. You can use either the **Instance image ID** attribute or the **Tag** value attribute depending on which criteria is identifiable with the instances that host these tools. The example below represents the filter you would use to suppress this finding type based on instances with a certain instance tag value.

  ```
  Finding type: UnauthorizedAccess:EC2/SSHBruteForce Instance tag value: devops
  ```

- Recon:EC2/PortProbeUnprotectedPort (p. 36) – Use a suppression rule to automatically archive findings when it is targeted to intentionally exposed instances.

  There may be cases in which instances are intentionally exposed, for example if they are hosting web servers. If this is the case in your AWS environment, we recommend that you set up a suppression rule for this finding. The suppression rule should consist of two filter criteria. The first criteria should use the **Finding type** attribute with a value of `Recon:EC2/PortProbeUnprotectedPort`. The second filter criteria should match the instance or instances that serve as a bastion host. You can use either the **Instance image ID** attribute or the **Tag** value attribute, depending on which criteria is identifiable with the instances that host these tools. The example below represents the filter you would use to suppress this finding type based on instances with a certain instance tag key in the console.

```
Finding type: Recon:EC2/PortProbeUnprotectedPort Instance tag key: prod
```

To create suppression rules in GuardDuty

Console

The GuardDuty console allows you to easily visualize, create and manage suppression rules. Suppression rules are generated in the same manner as filters, and your existing saved filters can be used as suppression rules. For more information on creating filters see Filtering findings (p. 91).

To create a suppression rule using the console:

1. On the GuardDuty **Findings** page, choose **Suppress findings** to open the suppression rule panel.
2. Select **Add filter criteria** in the filter bar to open the filter criteria menu. Select a criterion from the list and then enter a valid value for that criterion. To determine the correct values to use you can select a finding that you'd like to suppress from the findings table and review it's details from the same page in the console. You can continue adding filter criteria until only those findings you want to suppress are listed in the findings table.
3. Enter a name and a description for the suppression rule.
4. Choose **Save**.

You can also create a suppression rule from an existing saved filter. For more information on creating filters see Filtering findings (p. 91).

To create a suppression rule from a saved filter:

1. On the GuardDuty **Findings** page, choose **Suppress findings** to open the suppression rule panel.
2. from the **Saved rules** drop down select a saved filter.
3. Enter a name and a description for the suppression rule.
4. Choose **Save**.

Your suppression rules can be viewed, edited, or deleted at any time by selecting the rule from the **Saved rules** drop down in the console.

API

To create a suppression rule using API:

1. You can create suppression rules through the CreateFilter API. To do so, specify the filter criteria in a JSON file following the format of the example detailed below. The below example will suppress any unarchived findings where a DNS request to test.example.com was made.

```
{
        "Criterion": {
            "service.archived": {
                "Eq": [
                    "false"
                ]
            },
            "service.action.dnsRequestAction.domain": {
                "Eq": [
                    "test.example.com"
                ]
            }
        }
    }
```

For a list of JSON field names and their console equivalent see Filter attributes (p. 92).

You can test your filter criteria first by using the same JSON criterion in the ListFindings API and confirming that the correct findings have been selected or you can do this through the AWS CLI by following the below example using your own detector ID, and .json file.

```
 AWS  guardduty list-findings --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
finding-criteria file://criteria.json
```

2. Upload your filter to be used as suppression rule with the CreateFilter API or by using the AWS CLI following the example below with your own detector ID, a name for the suppression rule, and .json file.

```
 AWS  guardduty create-filter --action ARCHIVE --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --name yourfiltername --finding-criteria
 file://criteria.json
```

You can view a list of your filters programmatically with the ListFilter API, you can see details of an individual filter by supplying the filter name to the GetFilter API. Update filters using UpdateFilter or delete them with the DeleteFilter API.

Working with trusted IP lists and threat lists

Amazon GuardDuty monitors the security of your AWS environment by analyzing and processing VPC Flow Logs, AWS CloudTrail event logs, and DNS logs. You can customize this monitoring scope by configuring GuardDuty to stop alerts for trusted IPs from your own *trusted IP lists* and alert on known malicious IPs from your own *threat lists*.

Trusted IP lists and threat lists apply only to traffic destined for publicly routable IP addresses. The effects of a list apply to all VPC Flow Log and CloudTrail findings, but do not apply to DNS findings.

GuardDuty can be configured to use the following types of lists.

Trusted IP list

Trusted IP lists consist of IP addresses that you have trusted for secure communication with your AWS infrastructure and applications. GuardDuty does not generate VPC Flow Log or CloudTrail

findings for IP addresses on trusted IP lists. You can include a maximum of 2000 IP addresses and CIDR ranges in a single trusted IP list. At any given time, you can have only one uploaded trusted IP list per AWS account per Region.

Threat list

Threat lists consist of known malicious IP addresses. These list can be supplied by third party threat intelligence or created specifically for your organization. GuardDuty generates findings based on threat lists. You can include a maximum of 250,000 IP addresses and CIDR ranges in a single threat list. GuardDuty only generates findings based on activity that involves IP addresses and CIDR ranges in your threat lists, findings will not be generated based of domain names. At any given time, you can have up to six uploaded threat lists per AWS account per Region.

Note
If you include the same IP on both a trusted IP list and threat list it will be processed by the trusted IP list first, and will not generate a finding.

In multi-account environments only users from GuardDuty administrator accounts can upload and manage trusted IP lists and threat lists. Trusted IP lists and threat lists that are uploaded by the administrator account are imposed on GuardDuty functionality in its member accounts. In other words, in member accounts GuardDuty generates findings based on activity that involves known malicious IP addresses from the administrator's threat lists and does not generate findings based on activity that involves IP addresses from the administrator's trusted IP lists. For more information, see Managing multiple accounts in Amazon GuardDuty (p. 124).

List formats

GuardDuty accepts lists in the following formats:

- `TXT`
- `STIX`
- `OTX_CSV`
- `ALIEN_VAULT`
- `PROOF_POINT`
- `FIRE_EYE`

The maximum size of the file that hosts your trusted IP list or threat list is 35MB.

In your trusted IP lists and threat lists, IP addresses and CIDR ranges must appear one per line. The following is a sample list in Plaintext (TXT) format:

```
54.20.175.217
205.0.0.0/8
```

Permissions required to upload trusted IP lists and threat lists

Various IAM identities require special permissions to work with trusted IP lists and threat lists in GuardDuty. An identity with the **AmazonGuardDutyFullAccess** managed policy attached can only rename and deactivate uploaded trusted IP lists and threat lists.

To grant various identities full access to working with trusted IP lists and threat lists (in addition to renaming and deactivating, this includes uploading, activating, deleting, and updating the location of the

lists), make sure that the following actions are present in the permissions policy attached to an IAM user, group, or role:

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::123456789123:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty"
        }
```

Important
These actions are not included in the **AmazonGuardDutyFullAccess** managed policy.

Using server-side encryption for trusted IP lists and threat lists

GuardDuty supports the following encryption types for lists: SSE-AES256 and SSE-KMS. SSE-C is not supported. For more information on encryption types for S3 see Protecting data using server-side encryption.

If your list is encrypted using server-side encryption SSE-KMS you must grant the GuardDuty service-linked role **AWSServiceRoleForAmazonGuardDuty** permission to decrypt the file in order to activate the list. Add the following statement to the KMS key policy and replace the account ID with your own:

```
{
 "Sid": "AllowGuardDutyServiceRole",
 "Effect": "Allow",
 "Principal": {
  "AWS": "arn:aws:iam::123456789123:role/aws-service-role/guardduty.amazonaws.com/
AWSServiceRoleForAmazonGuardDuty"
 },
 "Action": "kms:Decrypt*",
 "Resource": "*"
}
```

To upload trusted IP lists and threat lists

The following procedure describes how you can upload trusted IP lists and threat lists using the GuardDuty console.

To upload trusted IP lists and threat lists (console)

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Settings**, choose **Lists**.
3. On the **List management** page, choose **Add a trusted IP list** or **Add a threat list**.
4. In the dialog box, do the following:
   - For **List name**, enter ` a name for the list.
   - For **Location**, specify the location of the list - this is the S3 bucket where you store your trusted IP list or threat list and the file that contains your list.

Note
You can specify the location URL in the following formats:

- https://s3.amazonaws.com/bucket.name/file.txt
- https://s3-aws-region.amazonaws.com/bucket.name/file.txt
- http://bucket.s3.amazonaws.com/file.txt
- http://bucket.s3-aws-region.amazonaws.com/file.txt
- s3://bucket.name/file.txt

- For **Format**, choose your list's file type.
- Select the **I agree** check box.
- Choose **Add list**.

To activate or deactivate trusted IP lists and threat lists

The following procedures describe how you can activate or deactivate trusted IP lists and threat lists in GuardDuty once they are uploaded. GuardDuty includes the uploaded lists in its monitoring of your AWS environment only if they are active.

To activate trusted IP lists and threat lists (console)

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Settings**, choose **Lists**.
3. On the **List management** page, locate the trusted IP set or a threat list that you want to activate, and then choose the radio button under the **Active** column.

To deactivate trusted IP lists and threat lists (API or CLI)

- You can deactivate your trusted IP lists or threat lists by running the UpdateThreatIntelSet and UpdateIPSet operations, or the update-ip-set and update-threat-intel-set CLI commands.

  For example, you can run the following command:

```
 AWS  guardduty update-ip-set --detector-id <detector-id> --ip-set-id <ip-set-id> --no-activate
```

  Make sure to replace <detector-id> and <ip-set-id> with a valid detector ID and trusted IP list ID.

To update trusted IP lists and threat lists

If you make changes to a trusted IP list or a threat list that is already uploaded and activated in GuardDuty (for example, rename the list or add more IP addresses to it), you must update this list in GuardDuty and reactivate it in order for GuardDuty to use the latest version of the list in its security monitoring scope. To update a trusted IP or threat list, you can either use the procedure below or run the UpdateThreatIntelSet and UpdateIPSet operations of the GuardDuty API.

To update trusted IP lists and threat lists (console)

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Settings**, choose **Lists**.

3. On the **List management** page, locate the trusted IP set or a threat list that you want to update, and then choose the pencil icon under the **Active** column.
4. In the **Update list** pop up window, verify all specified list information, choose **I agree**, and then choose **Update list**.
5. On the **List management** page, locate the trusted IP set or a threat list that you want to activate again, and then choose the radio button under the **Active** column.

To learn how to import threat intel feeds programatically, see the How to automate the import of third-party threat intelligence feeds blog article.

Exporting findings

GuardDuty supports exporting active findings to CloudWatch Events and, optionally, to an Amazon S3 bucket. New Active findings that GuardDuty generates are automatically exported within about 5 minutes after the finding is generated. You can set the frequency for how often updates to Active findings are exported to CloudWatch Events. The frequency that you select applies to the exporting of new occurences of existing findings to CloudWatch Events, your S3 bucket (if configured), and Detective (if integrated). For more information on updates to existing findings see GuardDuty finding aggregation (p. 24)

Note the following about export settings for findings:

- Export settings are regional, which means you need to configure export options for each Region in which you're using GuardDuty. However, you can use the same bucket in a single Region as the export destination for each Region you use GuardDuty in.
- Archived findings, including new instances of suppressed findings, aren't exported. If you unarchive a finding, its status is updated to **Active**, and it will be exported at the next interval.
- If you enable findings export in a GuardDuty administrator account all findings from associated member accounts that are generated in the current Region are also exported to the same location that you configured for the administrator account.

To configure settings for exporting Active findings to an Amazon S3 bucket you will need a KMS key that GuardDuty can use to encrypt findings, and an S3 bucket with permissions that allows GuardDuty to upload objects. Review this topic to learn how to configure findings export and frequency.

Permissions required to configure findings export

When you configure options for exporting findings, you select a bucket to store the findings in and a KMS key to use for data encryption. In addition to permissions to GuardDuty actions, you must also have permissions to the following actions to successfully configure options for exporting findings.

- kms:ListAliases
- s3:CreateBucket
- s3:GetBucketLocation
- s3:ListAllMyBuckets
- s3:PutBucketAcl
- s3:PutBucketPublicAccessBlock
- s3:PutBucketPolicy
- s3:PutObject

Important
If your policy explicitly denies `putObjectAcl` you will be unable to publish findings.

Granting GuardDuty permission to a KMS key

GuardDuty encrypts the findings data in your bucket by using an AWS KMS key. To successfully configure findings export, you must first give GuardDuty permission to use a KMS key. You grant the permissions by changing the key policy for the key you use.

If you plan to use a new key for GuardDuty findings, create a key before proceeding. If you are using a key in another account, you need to log in to the account that owns the key to apply the key policy. You also need the key ARN for a key from another account when you configure export settings.

To modify the key policy to allow GuardDuty to use the key

1. Open the AWS KMS console at https://console.aws.amazon.com/kms.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. Create a new key or choose an existing key that you plan to use for encrypting exported findings. The key must be in the same Region as the bucket, however, you can use this same bucket and key pair for each region you want to export findings from.
4. Select your key then copy the key ARN from the **General configuration** panel.
5. Under **Key policy**, choose **Edit**.

   Tip
   If **Switch to policy view** is displayed, choose that to display the key policy, and then choose **Edit**.
6. Add the following key policy granting GuardDuty access to your key. Replace the values in red to match your environment.

   Replace *Region* with the Region that the KMS key is in. Replace *111122223333* with the AWS account number of the source account that owns the GuardDuty detector. Replace *KMSKeyId* with the key ID of the key that you chose for encryption and replace *SourceDetectorID* with the source account's GuardDuty detector ID for the current Region.

   This statement allows GuardDuty to use only the key that you changed the policy for. When editing the key policy make sure your JSON syntax is valid, if you add the statement before the final statement you must add a comma after the closing bracket.

```
{
    "Sid": "AllowGuardDutyKey",
    "Effect": "Allow",
    "Principal": {
        "Service": "guardduty.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey",
    "Resource": "arn:aws:kms:Region:111122223333:key/KMSKeyId",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "111122223333",
            "aws:SourceArn":
 "arn:aws:guardduty:Region:111122223333:detector/SourceDetectorID"
        }
    }
}
```

Note
If you're using GuardDuty in an manually-enabled Region, replace the value for the "Service" with the Regional endpoint for that Region. For example, if you're using GuardDuty in the Middle East (Bahrain) (me-south-1) Region, replace `"Service": "guardduty.amazonaws.com"` with `"Service": "guardduty.me-south-1.amazonaws.com"`.

7. Choose **Save**.
8. (Optional) If you plan on using an existing bucket, copy the key ARN to a notepad for use in later steps. To locate the key ARN, see Finding the key ID and ARN.

Granting GuardDuty permissions to a bucket

When using a pre-existing bucket withing your account, or in a different AWS account, you must grant GuardDuty permission to upload objects to that bucket. You grant these permissions by adding an S3 bucket policy. If you are using a pre-existing bucket, expand the following section for step-by-step instructions on adding a bucket policy.

To add a bucket policy that allows GuardDuty to upload objects your bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the bucket you plan to use for exported findings.
3. Choose **Permissions**, and then choose **Bucket Policy**.
4. Copy the **example policy** and paste it into the **Bucket policy editor**.
5. Replace the placeholder values in the example policy with the values appropriate for your environment.

   Replace *myBucketName* . Replace *[optional prefix]* with a folder location for your exported findings (GuardDuty will create this location during set up if it does not already exist). Replace *Region* with the Region that the KMS key is in. Replace *111122223333* with the AWS account number of the account that owns the bucket. Replace *KMSKeyId* with the key ID of the key that you chose for encryption and replace *SourceDetectorID* with the source account's GuardDuty detector ID for the current Region.

Example policy

The following example policy shows how to grant GuardDuty permission to send findings to your Amazon S3 bucket. If you change the path after you configure findings export, you must modify the policy to grant permission to the new location.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGuardDutygetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
            "Resource": "arn:aws:s3:::myBucketName",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn":
"arn:aws:guardduty:Region:111122223333:detector/SourceDetectorID"

                }
            }
        },
        {
            "Sid": "AllowGuardDutyPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::myBucketName/[optional prefix]/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn":
"arn:aws:guardduty:Region:111122223333:detector/SourceDetectorID"

                }
            }
        },
        {
            "Sid": "DenyUnencryptedUploadsThis is optional",
            "Effect": "Deny",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::myBucketName/[optional prefix]/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyIncorrectHeaderThis is optional",
            "Effect": "Deny",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::myBucketName/[optional prefix]/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption-aws-kms-key-id":
"arn:aws:kms:Region:111122223333:key/KMSKeyId"
                }
            }
        },
        {
            "Sid": "DenyNon-HTTPS",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::myBucketName/[optional prefix]/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
```

```
}
```

Note
If you're using GuardDuty in an manually-enabled Region, replace the value for the service with the Regional endpoint for the Region. For example, if you're using GuardDuty in the Middle East (Bahrain) (me-south-1) Region, replace `"Service": "guardduty.amazonaws.com"` with `"guardduty.me-south-1.amazonaws.com"`.

Exporting findings to a bucket with the Console

When you configure findings export, you can choose an existing S3 bucket or have GuardDuty create a new bucket to store exported findings in. If you choose to use a new bucket, GuardDuty applies all necessary permissions to the created bucket. If you use an existing bucket, you must first update the bucket policy to allow GuardDuty to put findings into the bucket.

You may also export findings to an existing bucket in another account.

When choosing a new or existing bucket in your account, you can add a prefix. When configuring findings export GuardDuty creates a new folder in the S3 bucket for your findings. The prefix will prepend the default folder structure created by GuardDuty, which is `/AWSLogs/111122223333/GuardDuty/Region`.

Important
The KMS key and S3 bucket must be in the same Region.

Before completing these steps makes sure you have configured a KMS key and, if using an existing bucket, added a bucket policy to allow GuardDuty to create objects.

New bucket in your account

To configure findings export using a new bucket

1. Add a policy to the KMS key GuardDuty will use to encrypt findings. For an example policy see Granting GuardDuty permission to a KMS key (p. 102)
2. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
3. Choose **Settings**.
   a. Choose **New bucket** to create a new bucket to store exported findings.

      In the **Name the bucket** field, enter a name for the bucket. The name must be unique across all S3 buckets. Bucket names must start with a lowercase letter or a number.
   b. If you used an `[optional prefix]` in your bucket policy you must enter that prefix under **Log file prefix**, otherwise this is optional. When you enter a value, the example path below the field is updated to reflect the path to bucket location where your exported findings will be stored.
   c. Under **KMS encryption**, do one of the following:
      - Select **Choose key from your account**.

        Then choose the key alias of the key that you changed the policy for from the **Key alias** list.
      - Select **Choose key from another account**.

        Then enter the full ARN to the key that you changed the policy for.

        The key that you choose must be in the same Region as the bucket. To learn how to find the key ARN, see Finding the key ID and ARN.

d. Choose **Save**.

Existing bucket in your account

To configure findings export using an existing bucket

1. Add a policy to the KMS key GuardDuty will use to encrypt findings. For an example policy see Granting GuardDuty permission to a KMS key (p. 102)
2. Attach a policy granting GuardDuty permission to upload objects to your S3 bucket. For an example policy see Granting GuardDuty permissions to a bucket (p. 103)
3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
4. Choose **Settings**.
   a. Choose **Existing bucket in your account**.

      In the **Name the bucket** field enter the name of your bucket.
   b. Optional. Under **Log file prefix**, enter a path prefix to use. GuardDuty will create a new folder in the bucket with specified prefix name. When you enter a value, the example path below the field is updated to reflect the path to exported findings in the bucket.
   c. Under **KMS encryption**, do one of the following:
      - Select **Choose key from your account**.

        Then choose the key alias of the key that you changed the policy for from the **Key alias** list.
      - Select **Choose key from another account**.

        Then enter the full ARN to the key that you changed the policy for.

        The key that you choose must be in the same Region as the bucket. To learn how to find the key ARN, see Finding the key ID and ARN.
   d. Choose **Save**.

Existing bucket in another account

To configure findings export using an existing bucket in another account

1. Add a policy to the KMS key GuardDuty will use to encrypt findings. For an example policy see Granting GuardDuty permission to a KMS key (p. 102)
2. Attach a policy granting GuardDuty permission to upload objects to the S3 bucket in another account. For an example policy see Granting GuardDuty permissions to a bucket (p. 103).

   Note
   Use the account ID of the account that owns the bucket in the policy
3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
4. Choose **Settings**.
   a. Choose **Existing bucket in another account**.
   b. In the **Bucket ARN field**, enter the ARN for the bucket from another account to use.
   c. Under **KMS encryption** enter the full ARN of the key that you changed the policy for.

      The key you choose must be in the same Region as the bucket. To learn how to find the key ARN, see Finding the key ID and ARN.
   d. Choose **Save**.

Export access error

After you configure finding export options, if GuardDuty is unable to export findings, an error message is displayed on the **Settings** page. This can happen when GuardDuty can no longer access the target resource, such as when the S3 bucket is deleted or the permissions to the bucket are changed. This can also happen when the KMS key used to encrypt data in the bucket becomes inaccessible.

When exporting fails, GuardDuty sends a notification to the email associated with the account to let you know about the issue. If you don't resolve the issue, GuardDuty disables finding export in the account. You can update the configuration to restart finding export at any time.

If you receive this error, review the information in this topic about how to enable and configure findings export. For example, review the key policy and confirm that the correct policy is applied to the KMS key that you chose for encryption.

Setting the frequency for exporting updated active findings

Configure the frequency for exporting updated Active findings as appropriate for your environment. By default, updated findings are exported every 6 hours. This means that any findings that are updated after the most recent export are included in the next export. If updated findings are exported every 6 hours and the export occurs at 12:00, any finding that you update after 12:00 is exported at 18:00.

To set the frequency

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Choose **Settings**.
3. In the **Findings export options** section, select **Frequency for updated findings**. This sets the frequency for exporting updated Active findings to both CloudWatch Events and Amazon S3. You can choose from the following:
   - **Update CWE and S3 every 15 minutes**
   - **Update CWE and S3 every 1 hour**
   - **Update CWE and S3 every 6 hours (default)**
4. Choose **Save**.

Creating custom responses to GuardDuty findings with Amazon CloudWatch Events

GuardDuty creates an event for Amazon CloudWatch Events when any change in findings takes place. Finding changes that will create a CloudWatch event include newly generated findings or newly aggregated findings. Events are emitted on a best effort basis.

Every GuardDuty finding is assigned a finding ID. GuardDuty creates a CloudWatch event for every finding with a unique finding ID. All subsequent occurrences of an existing finding are aggregated to the original finding.

> **Note**
> If your acccount is a GuardDuty delegated administrator CloudWatch events are published to your account in addition to the member account that it originated from.

By using CloudWatch events with GuardDuty, you can automate tasks to help you respond to security issues revealed by GuardDuty findings.

In order to receive notifications about GuardDuty findings based on CloudWatch Events, you must create a CloudWatch Events rule and a target for GuardDuty. This rule enables CloudWatch to send notifications for findings that GuardDuty generates to the target that is specified in the rule. For more information, see Creating a CloudWatch Events rule and target for GuardDuty (CLI) (p. 113).

Topics

CloudWatch Events notification frequency for GuardDuty

Notifications for newly generated findings with a unique finding ID – GuardDuty sends a notification based on its CloudWatch event within 5 minutes of the finding. This event (and this notification) also includes all subsequent occurrences of this finding that take place in the first 5 minutes since this finding with a unique ID is generated.

> **Important**
> You CANNOT customize the default frequency (5 minutes) of notifications sent about the newly generated findings.

Notifications for subsequent finding occurrences – By default, for every finding with a unique finding ID, GuardDuty aggregates all subsequent occurrences of a particular finding that take place in 6-hour intervals into a single event. GuardDuty then sends a notification about these subsequent occurrences based on this event. In other words, by default, for the subsequent occurrences of the existing findings, GuardDuty sends notifications based on CloudWatch events every 6 hours.

> **Important**
> You can customize the default frequency of notifications sent about the subsequent finding occurrences. Possible values are 15 minutes, 1 hour, or the default 6 hours. You can update this value using the CreateDetector or the UpdateDetector API operations. You can also update this value through the GuardDuty console - choose **Settings** and then under **CloudWatch Events**, choose one of the available values from the **Updated findings** pull-down menu.
> Only users from an administrator account can customize the default frequency of notifications sent about the subsequent finding occurrences to CloudWatch Events. Users from member accounts CANNOT customize this frequency value. The frequency value set by the administrator account in its own account is imposed on GuardDuty functionality in all its member accounts. In other words, if a user from an administrator account sets this frequency value to 1 hour, all member accounts will also have the 1 hour frequency of notifications about the subsequent finding occurrences sent to CloudWatch Events. For more information, see Managing multiple accounts in Amazon GuardDuty (p. 124).

Monitoring archived GuardDuty findings with CloudWatch Events

For the manually archived findings, the initial and all subsequent occurrences of these findings (generated after the archiving is complete) are sent to CloudWatch Events per frequency described above.

For the auto-archived findings, the initial and all subsequent occurrences of these findings (generated after the archiving is complete) are *not* sent to CloudWatch Events.

CloudWatch event format for GuardDuty

The CloudWatch event for GuardDuty has the following format.

```
{
 "version": "0",
 "id": "cd2d702e-ab31-411b-9344-793ce56b1bc7",
 "detail-type": "GuardDuty Finding",
 "source": "aws.guardduty",
 "account": "111122223333",
 "time": "1970-01-01T00:00:00Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {GUARDDUTY_FINDING_JSON_OBJECT}
}
```

Note
The detail value returns the JSON details of a single finding as an object, as opposed to returning the "findings" value which can support multiple findings within an array.

For the complete list of all the parameters included in `GUARDDUTY_FINDING_JSON_OBJECT`, see GetFindings. The `id` parameter that appears in `GUARDDUTY_FINDING_JSON_OBJECT` is the finding ID previously described.

Creating a CloudWatch Events rule to notify you of GuardDuty findings (console)

You can use CloudWatch Events with GuardDuty to set up automated finding alerts by sending GuardDuty finding events to a messaging hub to help increase the visibility of GuardDuty findings. This topic shows you how to send findings alerts to email, Slack, or Amazon Chime by setting up an SNS topic and then connecting that topic to an CloudWatch Events event rule.

Setup an Amazon SNS topic and endpoint

To begin, you must first set up a topic in Amazon Simple Notification Service and add an endpoint. For more information on refer to the SNS guide.

This procedure establishes where you want to send GuardDuty finding data. The SNS topic can be added to an CloudWatch Events Event rule during or after the creation of the Event Rule.

Email setup

Creating an SNS topic

1. Sign in to the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. Select **Topics** from the navigation pane and then **Create Topic**.
3. In the Create topic section, select **Standard**. Next, enter a Topic name, for example **`GuardDuty_to_Email`**. Other details are optional.
4. Choose **Create Topic**. The Topic details for your new topic will open.
5. In the Subscriptions section select **Create Subscription**
6. a. From the **Protocol** menu, select **Email**.

b. In the **Endpoint** field add the email address you would like to receive notifications at.

Note
You will be required to confirm your subscription through your email client after creating it.

c. Choose **Create subscription**

7. Check for a subscription message in your inbox and choose **Confirm Subscription**

Slack setup

Creating an SNS topic

1. Sign in to the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. Select **Topics** from the navigation pane and then **Create Topic**.
3. In the Create topic section, select **Standard**. Next, enter a Topic name, for example **`GuardDuty_to_Slack`**. Other details are optional. Choose **Create topic** to finalize.

Configuring an AWS Chatbot client

1. Navigate to the AWS Chatbot console
2. From the **Configured clients** panel, select **Configure new client**.
3. Choose Slack and confirm with "Configure".

Note
When choosing Slack you must confirm permissions for AWS Chatbot to access your channel by selecting "allow".

4. Select **Configure new channel** to open the configuration details pane.

a. Enter a name for the channel.

b. For Slack channel, choose the channel that you want to use. To use private Slack channel with AWS Chatbot, choose Private channel.

c. In Slack, copy the Channel ID of the private channel by right-clicking on the channel name and selecting Copy Link.

d. On the AWS Management Console, in the AWS Chatbot window, paste the ID you copied from slack into the Private channel ID field.

e. In **Permissions**, chose to create an IAM role using a template, if you do not have a role already.

f. For **Policy** templates, choose Notification permissions. This is the IAM policy template for AWS Chatbot. It provides the necessary read and list permissions for CloudWatch alarms, events and logs, and for Amazon SNS topics.

g. Choose the Region you previously created your SNS topic in, and then select the Amazon SNS topic you created to send notifications to the Slack channel.

5. Select **Configure**.

Chime setup

Creating an SNS topic

1. Sign in to the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. Select **Topics** from the navigation pane and then **Create Topic**.
3. In the Create topic section, select **Standard**. Next, enter a Topic name, for example **`GuardDuty_to_Chime`**. Other details are optional. Choose **Create topic** to finalize.

Configuring a AWS Chatbot client

1. Navigate to the AWS Chatbot console
2. From the **Configured clients** panel, select **Configure new client**.
3. Choose Chime and confirm with "Configure".
4. From the **Configuration details** pane, enter a name for the channel.
5. In Chime open the desired chat room
   a. Choose the gear icon in the upper-right corner and choose **Manage webhooks and bots**.
   b. Select **Copy URL** to copy the webhook URL to your clipboard.
6. On the AWS Management Console, in the AWS Chatbot window, paste the URL you copied into the **Webhook URL** field.
7. In **Permissions**, chose to create an IAM role using a template, if you do not have a role already.
8. For **Policy** templates, choose Notification permissions. This is the IAM policy template for AWS Chatbot. It provides the necessary read and list permissions for CloudWatch alarms, events and logs, and for Amazon SNS topics.
9. Choose the Region you previously created your SNS topic in, and then select the Amazon SNS topic you created to send notifications to the Chime room.
10. Select **Configure**.

Setup a CloudWatch event for GuardDuty findings

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Select **Rules** from the navigation pane and then **Create Rule**.
3. From the **Service Name** menu, choose **GuardDuty**.
4. From the **Event Type** menu, choose **GuardDuty Finding**.
5. In **Event Pattern Preview** choose **Edit**.
6. Paste the below JSON code into **Event Pattern Preview** and choose **Save**

```
{
  "source": [
    "aws.guardduty"
  ],
  "detail-type": [
    "GuardDuty Finding"
  ],
  "detail": {
    "severity": [
      4,
      4.0,
      4.1,
      4.2,
      4.3,
      4.4,
      4.5,
      4.6,
      4.7,
      4.8,
      4.9,
      5,
      5.0,
      5.1,
      5.2,
      5.3,
      5.4,
```

```
      5.5,
      5.6,
      5.7,
      5.8,
      5.9,
      6,
      6.0,
      6.1,
      6.2,
      6.3,
      6.4,
      6.5,
      6.6,
      6.7,
      6.8,
      6.9,
      7,
      7.0,
      7.1,
      7.2,
      7.3,
      7.4,
      7.5,
      7.6,
      7.7,
      7.8,
      7.9,
      8,
      8.0,
      8.1,
      8.2,
      8.3,
      8.4,
      8.5,
      8.6,
      8.7,
      8.8,
      8.9
    ]
  }
}
```

Note
The above code will alert for any Medium to High finding.

7. In the **Targets** section click **Add Target**.
8. From the **Select Targets** menu, choose **SNS Topic**.
9. For **Select Topic** select the name of the SNS Topic you created in Step 1.
10. Configure the input for the event.

    - If you are setting up notifications for Chime or Slack skip to Step 11, the input type defaults to **Matched event**.
    - If you are setting up notifications for email via SNS follow the steps below to customize the message sent to your inbox using the following steps:

    a. Expand **Configure input** and then choose **Input Transformer**.
    b. Copy the following code and paste it into the **Input Path** field.

```
{
    "severity": "$.detail.severity",
```

```
    "Account_ID": "$.detail.accountId",
    "Finding_ID": "$.detail.id",
    "Finding_Type": "$.detail.type",
    "region": "$.region",
    "Finding_description": "$.detail.description"
}
```

c. Copy the following code and paste it into the **Input Template** field to format the email.

```
"AWS <Account_ID> has a severity <severity> GuardDuty finding type <Finding_Type>
 in the <region> region."
"Finding Description:"
"<Finding_description>. "
"For more details open the GuardDuty console at https://console.aws.amazon.com/
guardduty/home?region=<region>#/findings?search=id=<Finding_ID>"
```

11. Click **Configure Details**.
12. In the **Configure rule details** page, enter a **Name** and **Description** for the rule, and then choose **Create Rule**.

Creating a CloudWatch Events rule and target for GuardDuty (CLI)

The following procedure shows how to use AWS CLI commands to create a CloudWatch Events rule and target for GuardDuty. Specifically, the procedure shows you how to create a rule that enables CloudWatch to send events for all findings that GuardDuty generates and add an AWS Lambda function as a target for the rule.

> **Note**
> In addition to Lambda functions, GuardDuty and CloudWatch support the following target types: Amazon EC2 instances, Amazon Kinesis streams, Amazon ECS tasks, AWS Step Functions state machines, the `run` command, and built-in targets.

You can also create a CloudWatch Events rule and target for GuardDuty through the CloudWatch Events console. For more information and detailed steps, see Creating a CloudWatch Events rule that triggers on an event. In the **Event Source** section, select **`GuardDuty`** for **Service name** and **`GuardDuty Finding`** for **Event Type**.

To create a rule and target

1. To create a rule that enables CloudWatch to send events for all findings that GuardDuty generates, run the following CloudWatch CLI command.

   ```
   AWS events put-rule --name Test --event-pattern "{\"source\":
   [\"aws.guardduty\"]}"
   ```

   > **Important**
   > You can further customize your rule so that it instructs CloudWatch to send events only for a subset of the GuardDuty-generated findings. This subset is based on the finding attribute or attributes that are specified in the rule. For example, use the following CLI command to create a rule that enables CloudWatch to only send events for the GuardDuty findings with the severity of either 5 or 8:
   > ```
   > AWS events put-rule --name Test --event-pattern "{\"source\":
   > [\"aws.guardduty\"],\"detail-type\":[\"GuardDuty Finding\"],\"detail
   > \":{\"severity\":[5,8]}}"
   > ```

For this purpose, you can use any of the property values that are available in the JSON for GuardDuty findings.

2. To attach a Lambda function as a target for the rule that you created in step 1, run the following CloudWatch CLI command.

   ```
   AWS events put-targets --rule Test --targets Id=1,Arn=arn:aws:lambda:us-
   east-1:111122223333:function:<your_function>
   ```

   Note
   Make sure to replace <your_function> in the command above with your actual Lambda function for the GuardDuty events.

3. To add the permissions required to invoke the target, run the following Lambda CLI command.

   ```
   AWS lambda add-permission --function-name <your_function> --statement-id 1
   --action 'lambda:InvokeFunction' --principal events.amazonaws.com
   ```

   Note
   Make sure to replace <your_function> in the command above with your actual Lambda function for the GuardDuty events.

   Note
   In the procedure above, we're using a Lambda function as the target for the rule that triggers CloudWatch Events. You can also configure other AWS resources as targets to trigger CloudWatch Events. For more information, see PutTargets.

CloudWatch Events for GuardDuty multi-account environments

As a GuardDuty administrator CloudWatch Event rules in your account will trigger based on applicable findings from your member accounts . This means that if you set up a finding notifications through CloudWatch Events in your administrator account, as detailed in the preceding section, you will be notified of high and medium severity findings generated by your member accounts in addition to your own.

You can identify the member account the GuardDuty finding originated from with the `accountId` field of the finding's JSON details.

To start writing a custom event rule for a specific member account in your environment in the console, create a new rule and paste the following template into Event Pattern Preview, adding the account ID of the member account you want to trigger the event.

```
{
  "source": [
    "aws.guardduty"
  ],
  "detail-type": [
    "GuardDuty Finding"
  ],
  "detail": {
    "accountId": [
      "123456789012"
    ]
  }
}
```

Note
This example will trigger on any findings for the listed account ID. Multiple IDs can be added, separated by a comma following JSON syntax.

Remediating security issues discovered by GuardDuty

Amazon GuardDuty generates findings (p. 13) that indicate potential security issues. In this release of GuardDuty, the potential security issues indicate either a compromised EC2 instance or a set of compromised credentials in your AWS environment. The following sections describe the recommended remediation steps for these scenarios. If there are alternative remediation scenarios they will be described in the entry for that specific finding type. You can access the full information about a finding type by selecting it from the Active findings types table (p. 26).

Topics

Remediating a compromised EC2 instance

Follow these recommended steps to remediate a compromised EC2 instance in your AWS environment:

1. Investigate the potentially compromised instance for malware and remove any discovered malware. You can also refer to the AWS Marketplace for partner products that might help to identify and remove malware.
2. If you are unable to identify and stop unauthorized activity on your EC2 instance, we recommend that you terminate the compromised EC2 instance and replace it with a new instance as needed. The following are additional resources for securing your EC2 instances:
   - "Security and Network" section in Best practices for amazon EC2.
   - Amazon EC2 security groups for Linux instances and Amazon EC2 security groups for Windows instances.
   - Tips for securing your EC2 instances (Linux).
   - AWS security best practices
   - Infrastructure Domain Incidents on AWS
3. Browse for further assistance on the AWS developer forums: https://forums.aws.amazon.com/index.jspa
4. If you are a Premium Support package subscriber, you can submit a technical support request.

Remediating a compromised S3 Bucket

Follow these recommended steps to remediate a compromised S3 bucket in your AWS environment:

1. **Identify the affected S3 resource.**

A GuardDuty finding for S3 will list an S3 bucket, the bucket's Amazon Resource number (ARN) and a bucket owner in the finding details.

2. **Identify the source of the suspicious activity and the API call used.**

   The API call used will be listed as `API` in the finding details. The source will be an IAM principal (either an IAM user, role, or account) and identifying details will be listed in the finding. Depending on the source type, Remote IP or source domain info will be available and can help you evaluate whether the source was authorized. If the finding involved credentials from an EC2 instance the details for that resource will also be included.

3. **Determine whether the call source was authorized to access the identified resource.**

   For example consider the following:

   - If an IAM user was involved is it possible their credentials have been compromised? See the next section on Remediating Compromised AWS Credentials.
   - If an API was invoked from a principal that has no prior history of invoking this type of API, does this source need access permissions for this operation? Can the bucket permissions be further restricted?
   - If the access was seen from the **user name** `ANONYMOUS_PRINCIPAL` with **user type** of `AWSAccount` this indicates the bucket is public and was accessed. Should this bucket be public? If not, review the security recommendations below for alternative solutions to sharing S3 resources.
   - If the access was though a successful `PreflightRequest` call seen from the **user name** `ANONYMOUS_PRINCIPAL` with **user type** of `AWSAccount` this indicates the bucket has a cross-origin resource sharing (CORS) policy set. Should this bucket have a CORS policy? If not, ensure the bucket is not inadvertently public and review the security recommendations below for alternative solutions to sharing S3 resources. For more information on CORS see Using cross-origin resource sharing (CORS) in the S3 user guide.

If the access was authorized, you can ignore the finding. If you determine that your S3 data has been exposed or accessed by an unauthorized party review the following S3 security recommendations to tighten permissions and restrict access. Appropriate remediation solutions will depend on the needs of your specific environment.

These are some recommendations based on specific S3 access needs:

- For a centralized way to limit public access to your S3 data use S3 block public access. Block public access settings can be enabled for access points, buckets, and AWS Accounts through four different settings to control granularity of access. For more information see S3 Block Public Access Settings.
- AWS Access policies can be used to control how IAM users can access your resources or how your buckets can be accessed. For more information see Using Bucket Policies and User Policies.

  Additionally you can use Virtual Private Cloud (VPC) endpoints with S3 bucket policies to restrict access to specific VPC endpoints. For more information see Example Bucket Policies for VPC Endpoints for Amazon S3
- To temporarily allow access to your S3 objects to trusted entities outside your account you can create a Presigned URL through S3. This access is created using your account credentials and depending on the credentials used can last 6 hours to 7 days. For more information see Generating presigned URLs with S3.
- For use cases that require that sharing of S3 objects between different sources you can use S3 Access Points to create permission sets that restrict access to only those within your private network. For more information see Managing data access with Amazon S3 access points.
- To securely grant access to your S3 resources to other AWS Accounts you can use an access control list (ACL), for more information see Managing S3 Access with ACLs.

For a full overview of S3 security options see S3 Security best practices.

Remediating compromised AWS credentials

Follow these recommended steps to remediate compromised credentials in your AWS environment:

1. **Identify the affected IAM entity and the API call used.**

   The API call used will be listed as `API` in the finding details. The IAM entity (either an IAM user or role) and it's identifying information will be listed in the **Resource** section of a finding's details. The type of IAM entity involved can be determined by the **User Type** field, the name of the IAM entity will be in the **User name** field. The type of IAM entity involved in the finding can also be determined by the **Access key ID** used.

   For keys beginning with `AKIA`:

   This type of key is a long term customer-managed credential associated with an IAM user or AWS account root user. For information on managing access keys for IAM users see Managing access keys for IAM users.

   For keys beginning with `ASIA`:

   This type of key is a short term temporary credential generated by AWS Security Token Service. These keys exists for only a short time and cannot be viewed or managed in the AWS Management Console. IAM roles will always use AWS STS credentials, but they can also be generated for IAM Users, for more information on AWS STS see IAM: Temporary security credentials.

   If a role was used the **User name** field will indicate the name of the role used. You can determine how the key was requested with AWS CloudTrail by examining the `sessionIssuer` element of the CloudTrail log entry, for more information see IAM and AWS STS information in CloudTrail.
2. **Review permissions for the IAM entity.**

   Open the IAM console, depending on the type of entity used, choose the **Users** or **Roles** tab, and locate the affected entity by typing the identified name into the search field. Use the **Permission** and **Access Advisor** tabs to review effective permissions for that entity.
3. **Determine whether the IAM entity credentials were used legitimately.**

   Contact the user of the credentials to determine if the activity was intentional.

   For example, find out if the user did the following:
   - Invoked the API operation that was listed in the GuardDuty finding
   - Invoked the API operation at the time that is listed in the GuardDuty finding
   - Invoked the API operation from the IP address that is listed in the GuardDuty finding

If you confirm that the activity is a legitimate use of the AWS credentials, you can ignore the GuardDuty finding. If not, this activity could be the result of a compromise to that particular access key, the IAM user's user ID and password, or possibly the entire AWS account. If you suspect your credentials have been compromised, review the information in the My AWS account may be compromised article to remediate the issue.

Remediating Kubernetes security issues discovered by GuardDuty

Amazon GuardDuty generates findings (p. 13) that indicate potential Kubernetes security issues when Kubernetes protection is enabled for your account. For more information, see Kubernetes protection in

Amazon GuardDuty (p. 137). The following sections describe the recommended remediation steps for these scenarios. Specific remediation actions are described in the entry for that specific finding type. You can access the full information about a finding type by selecting it from the Active findings types table (p. 26).

Different types of attacks and configuration issues can trigger GuardDuty Kubernetes findings. This guide helps you identify the root causes of GuardDuty findings against your cluster and outlines appropriate remediation guidance. The following are the primary root causes that lead to GuardDuty Kubernetes findings:

- Configuration issues (p. 119)
- Compromised users (p. 119)
- Compromised pods (p. 121)
- Compromised nodes (p. 122)
- Compromised container images (p. 122)

> **Note**
> Before Kubernetes version 1.14, the `system:unauthenticated` group was associated to `system:discovery` and `system:basic-user` **ClusterRoles** by default. This may allow unintended access from anonymous users. Cluster updates do not revoke these permissions, which means that even if you have updated your cluster to version 1.14 or later, these permissions may still be in place. We recommend that you disassociate these permissions from the `system:unauthenticated` group. For instructions on removing these permissions, see Review and revoke unnecessary anonymous access.

Configuration issues

If a finding indicates a configuration issue, see the remediation section of that finding for guidance on resolving that particular issue. For more information, see the following finding types that indicate configuration issues:

- Policy:Kubernetes/AnonymousAccessGranted (p. 74)
- Policy:Kubernetes/ExposedDashboard (p. 75)
- Policy:Kubernetes/AdminAccessToDefaultServiceAccount (p. 74)
- Policy:Kubernetes/KubeflowDashboardExposed (p. 75)
- Any finding that ends in SuccessfulAnonymousAccess.

Remediating compromised Kubernetes users

A GuardDuty finding can indicate a compromised Kubernetes user when a user identified in the finding has performed an unexpected API action. You can identify the user in the **Kubernetes user details** section of a finding details in the console, or in the `resources.eksClusterDetails.kubernetesDetails.kubernetesUserDetails` of the findings JSON. These user details include `user name`, `uid`, and the Kubernetes groups that the user belongs to.

If the user was accessing the workload using an IAM entity, you can use the `Access Key details` section to identify the details of an IAM user or role. See the following user types and their remediation guidance.

> **Note**
> You can use Amazon Detective to further investigate the IAM user or role identified in the finding. While viewing the finding details in GuardDuty console, click on **Investigate in Detective**. Then click on the AWS user or role from the listed items to investigate it in Detective.

Built-in Kubernetes admin – The default user assigned by Amazon EKS to the IAM identity that created the cluster. This user type is identified by the user name `kubernetes-admin`.

To revoke access of a built-in Kubernetes admin:

- Identify the `userType` from the `Access Key details` section.
  - If the `userType` is **Role** and the role belongs to an EC2 instance role:
    - Identify that instance then follow the instructions in Remediating a compromised EC2 instance (p. 116).
  - If the `userType` is **User**, or is a **Role** that was assumed by a user:
    1. Rotate the access key of that user.
    2. Rotate any secrets that user had access to.
    3. Review the information in My AWS account may be compromised for further details.

OIDC authenticated user – A user granted access through an OIDC provider. Typically an OIDC user has an email address as a user name. You can check if your cluster uses OIDC with the following command: `aws eks list-identity-provider-configs --cluster-name` *`your-cluster-name`*

To revoke access of an OIDC authenticated user:

1. Rotate the credentials of that user in the OIDC provider.
2. Rotate any secrets that user had access to.

AWS-Auth ConfigMap defined user – An IAM user that was granted access through an AWS-auth ConfigMap. For more information see Managing users or IAM roles for your cluster in the &EKS; user guide. You can review their permissions using the following command: `kubectl edit configmaps aws-auth --namespace kube-system`

To revoke access of an AWS ConfigMap user:

1. Use the following command to open the ConfigMap.

```
kubectl edit configmaps aws-auth --namespace kube-system
```

2. Identify the role or user entry under the **mapRoles** or **mapUsers** section with the same user name as the one reported in the Kubernetes user details section of your GuardDuty finding. See the following example, where the admin user has been identified in a finding.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::444455556666:role/eksctl-my-cluster-nodegroup-standard-
wo-NodeInstanceRole-1WP3NUE3O6UCF
      user name: system:node:EC2_PrivateDNSName
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::123456789012:user/admin
      username: admin
      groups:
        - system:masters
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

3. Remove that user from the ConfigMap. See the following example where the admin user has been removed.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/eksctl-my-cluster-nodegroup-standard-
wo-NodeInstanceRole-1WP3NUE3O6UCF
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

4. If the `userType` is **User**, or is a **Role** that was assumed by a user:
   a. Rotate the access key of that user.
   b. Rotate any secrets that user had access to.
   c. Review the information in My AWS account may be compromised for further details.

If the finding does not have a `resource.accessKeyDetails` section, the user is a Kubernetes service account.

Service account – The service account provides an identity for pods and can be identified by a user name with the following format: `system:serviceaccount:`*`namespace`*`:`*`service_account_name`*.

To revoke access to a service account:

1. Rotate the service account credentials.
2. Review the guidance for pod compromise in the following section.

Remediating compromised Kubernetes pods

When GuardDuty specifies details of a pod or workload resource inside the `resource.kubernetesDetails.kubernetesWorkloadDetails` section, that pod or workload resource is likely compromised. A GuardDuty finding can indicate a single pod has been compromised or that multiple pods have been compromised through a higher-level resource. See the following compromise scenarios for guidance on how to identify the pod or pods that have been compromised.

Single pod compromise

If the `type` field inside the `resource.kubernetesDetails.kubernetesWorkloadDetails` section is **Pod**, the finding identifies a single pod. The name field is the `name` of the pod and `namespace` field is its namespace. Use the instructions in Identify the offending Pod and worker node to identify the worker node running the pod.

Pods compromised through workload resource

If the `type` field inside the `resource.kubernetesDetails.kubernetesWorkloadDetails` section identifies a **Workload Resource**, such as a `Deployment`, it is likely that all of the pods within that workload resource have been compromised. Use the instructions in Identify the offending Pods and worker nodes using workload name to identify all the pods of the Workload Resource and the nodes they are running on.

Pods compromised through service account

If a GuardDuty finding identifies a Service Account in the `resource.kubernetesDetails.kubernetesUserDetails` section, it is

likely that pods using the identified service account are compromised. The user name reported by a finding is a service account if it has the following format: `system:serviceaccount:`*`namespace`*`:`*`service_account_name`*. Use the instructions in Identify the offending Pods and worker nodes using service account name to identify all the pods using the service account and nodes they are running on.

After you have identified all the compromised pods and the nodes they are running on, use the following instructions in the EKS best practices guide to isolate the pod, rotate its credentials, and gather data for forensic analysis.

To remediate a compromised pod:

1. Identify the vulnerability that compromised the pods.
2. Implement the fix for that vulnerability and start new replacement pods.
3. Delete the vulnerable pods. For more information see Redeploy compromised Pod or Workload Resource.

Remediating compromised container images

When a GuardDuty finding indicates a pod compromise, the image used to launch the pod could be malicious or compromised. GuardDuty findings identify the container image within the `resource.kubernetesDetails.kubernetesWorkloadDetails.containers.image` field. You can determine if the image is malicious by scanning it for malware.

To remediate a compromised container image:

1. Stop using the image immediately and remove it from your image repository.
2. Identify all pods using the image. For more information see Identify pods with vulnerable or compromised container images and worker nodes.
3. Isolate the compromised pods, rotate credentials and gather data for analysis. For more information see the following instructions in the EKS best practices guide.
4. Delete all pods using the compromised image.

Remediating compromised Kubernetes nodes

A GuardDuty finding can indicate a node compromise if the user identified in the finding represents a node identity, or if the finding indicates the use of a privileged container.

The user identity is a worker node if the **username** field has following format: `system:node:node name`. For example, `system:node:ip-192-168-3-201.ec2.internal`. This indicates that the adversary has gained access to the node and it is using the node's credentials to talk to the Kubernetes API endpoint.

A finding indicates the use of a privileged container if one or more of the containers listed in the finding has the `resource.kubernetesDetails.kubernetesWorkloadDetails.containers.securityContext.privilege` finding field set to `True`.

To remediate a compromised node:

1. Use the instructions in the EKS best practices guide to isolate the pod, rotate its credentials, and gather data for forensic analysis.
2. Identify the service accounts used by all of the pods running on the node. Review their permissions and rotate the service accounts if needed.

3. Terminate the node.

Managing multiple accounts in Amazon GuardDuty

To manage multiple accounts in Amazon GuardDuty, you must choose a single AWS account to be the administrator account for GuardDuty. You can then associate other AWS accounts with the administrator account as member accounts. There are two ways to associate accounts with a GuardDuty administrator account: either through an AWS Organizations organization that both accounts are members of, or by sending an invitation through GuardDuty.

GuardDuty recommends using the AWS Organizations method. For more information about setting up an organization, see Creating an organization in the *AWS Organizations User Guide*.

Managing multiple accounts with AWS Organizations

If the account that you want to specify as the GuardDuty administrator account is part of an organization in AWS Organizations, then you can specify that account as the organization's delegated administrator for GuardDuty. The account that is registered as the delegated administrator automatically becomes the GuardDuty administrator account.

You can use this administrator account to enable and manage GuardDuty for any account in the organization when you add that account as a member account.

If you already have a GuardDuty administrator account with associated member accounts by invitation, you can register that account as the GuardDuty delegated administrator for the organization. When you do, all currently associated member accounts remain members, allowing you to take full advantage of the added functionality of managing your GuardDuty accounts with AWS Organizations.

For more information about supporting multiple accounts in GuardDuty through an organization see Managing GuardDuty accounts with AWS Organizations (p. 126).

Managing multiple accounts by invitation

If the accounts you want to associate are not part of your AWS Organizations organization, you can specify a administrator account in GuardDuty and then use the administrator account to invite other AWS accounts to become member accounts. When the invited account accepts the invitation, that account becomes a GuardDuty member account associated with the administrator account.

For more information about supporting multiple accounts by Invitation in GuardDuty see Managing GuardDuty accounts by invitation (p. 132).

Understanding the relationship between GuardDuty administrator and member accounts

When you use GuardDuty in a multiple-account environment, the administrator account can manage certain aspects of GuardDuty on behalf of the member accounts. The primary functions the administrator account can perform are the following:

- Add and remove associated member accounts. The process by which this is done differs based on whether the accounts are associated through organizations or by invitation.
- Manage the status of GuardDuty within associated member accounts, including enabling and suspending GuardDuty.

  Note
  Delegated administrator accounts managed with AWS Organizations automatically enable GuardDuty in accounts added as members.
- Customize findings within the GuardDuty network through the creation and management of suppression rules, trusted IP lists, and threat lists. Member accounts lose access to these features in a multiple-account environment.

The following table details the relationship between GuardDuty administrator and member accounts.

Account designations listed as *Self* can take the listed action only in their own accounts. A designation of *Any* indicates that account can perform the described action for any associated account, and *All* denotes actions that are applied to all associated accounts when taken by the designated account. Table cells with dashes (–) indicate that an account of that designation cannot perform the listed action.

Action	Designation		
	administrator (organizations)	**administrator** (by invitation)	**Member**
View all AWS Organizations member accounts regardless of GuardDuty status	Any	–	–
Automatically Enable S3 Protection for New Accounts	All		
Enable GuardDuty	Any	Self	–
View GuardDuty findings	Any	Any	Self
Archive findings	Any	Any	–
Apply suppression rules	All	All	–
Generate sample findings	Self	Self	Self
Create trusted IP or threat lists	All	All	–
Update trusted IP or threat lists	All	All	–
Delete trusted IP or threat lists	All	All	–
Set CloudWatch Events notification frequency	All	All	–
Set Amazon S3 location for exporting findings	All	All	–
Suspend GuardDuty	Any*	Any*	–

* Indicates this action must be taken for all associated accounts before being taken in the designated account.

Managing GuardDuty accounts with AWS Organizations

When you use GuardDuty with an AWS Organizations organization, you can designate any account within the organization to be the GuardDuty delegated administrator. Only the organization management account can designate GuardDuty delegated administrators.

An account that is designated as a delegated administrator becomes a GuardDuty administrator account, has GuardDuty automatically enabled in the designated Region, and is granted permission to enable and manage GuardDuty for all accounts in the organization within that Region. The other accounts in the organization can be viewed and added as GuardDuty member accounts associated with the delegated administrator account.

If you have already set up a GuardDuty administrator with associated member accounts by invitation, and the member accounts are part of the same organization, their **Type** changes from **by Invitation** to **via Organizations** when you set a GuardDuty delegated administrator for your organization. If the new delegated administrator previously added members by invitation that are not part of the same organization, their **Type** is **by Invitation**. In both cases, these previously added accounts are member accounts to the organization's GuardDuty delegated administrator.

You can continue to add accounts as members even if they are outside of your organization. To learn more, see Designating administrator and member accounts through invitation (console) (p. 132) and Designating GuardDuty administrator and member accounts through invitation (API) (p. 133).

Important considerations for GuardDuty delegated administrators

Take note of the following factors that define how the delegated administrator operates in GuardDuty:

A delegated administrator can manage a maximum of 5000 members.

There is a limit of 5000 member accounts per GuardDuty delegated administrator. However, there could be more than 5000 accounts in your organization. The number of **All** accounts in your organization is displayed on the **Accounts** page of the GuardDuty console.

If you exceed 5000 member accounts you will receive a notification through CloudWatch, AWS Health Dashboard, and in an email to the delegated administrator account.

A delegated administrator is Regional.

Unlike AWS Organizations, GuardDuty is a Regional service. This means that GuardDuty delegated administrators, and their member accounts must be added in each desired Region for account management through AWS Organizations to be active in every Region. In other words, if the organization management account designates a delegated administrator for GuardDuty in only US East (N. Virginia) that delegated administrator will only manage member accounts added in that Region. For more information on Regions in GuardDuty see Regions and endpoints (p. 191).

An organization can have only one delegated administrator.

You can only have one delegated administrator per account. If you have designated an account as a delegated administrator in one region, that account must be your delegated administrator in all other regions. To change your delegated administrator after one has already been set, see the procedure for de-registering a delegated administrator.

Setting your Organization management account as the delegated administrator is not recommended.

The Organization management account can be the delegated administrator, but this is not recommended based on AWS Security best practices following the principle of least privilege.

Changing a delegated administrator does not disable GuardDuty for member accounts.

If you remove the delegated administrator, all associated member accounts are removed as GuardDuty members, but GuardDuty is not disabled in those accounts.

Permissions required to designate a delegated administrator

When delegating a GuardDuty delegated administrator you must have permissions to enable GuardDuty as well as certain AWS Organizations API actions listed in the following policy statement.

You can add the following statement to the end of an IAM policy to grant these permissions:

```
{
    "Sid": "PermissionsForGuardDutyAdmin",
    "Effect": "Allow",
    "Action": [
        "guardduty:EnableOrganizationAdminAccount",
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization"
    ],
    "Resource": "*"
}
```

Additionally, if you wish to designate your AWS Organizations management account as the GuardDuty delegated administrator that entity will need `CreateServiceLinkedRole` permissions to initialize GuardDuty. This can be added to an IAM policy using the following statement, replacing the account ID with the ID of your organization management account:

```
{
 "Sid": "PermissionsToEnableGuardDuty"
 "Effect": "Allow",
 "Action": [
  "iam:CreateServiceLinkedRole"
 ],
 "Resource": "arn:aws:iam::123456789012:role/aws-service-role/guardduty.amazonaws.com/
AWSServiceRoleForAmazonGuardDuty",
 "Condition": {
  "StringLike": {
   "iam:AWSServiceName": "guardduty.amazonaws.com"
  }
 }
}
```

Note
If you're using GuardDuty in an manually-enabled Region, replace the value for the "Service" with the Regional endpoint for the Region. For example, if you're using GuardDuty in the Middle East (Bahrain) (me-south-1) Region, replace `"Service": "guardduty.amazonaws.com"` with `"Service": "guardduty.me-south-1.amazonaws.com"`.

Designating a GuardDuty delegated administrator

The following procedures show you how to designate a delegated administrator for your AWS organization and add member accounts. Select Console or API and follow the provided steps.

Console

Step 1 — register a GuardDuty delegated administrator for your organization

1. Log in to the AWS Management Console using the management account for your AWS Organizations organization.
2. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Is GuardDuty already enabled in your account?

   - If GuardDuty is not already enabled, you can select **Get Started** and then designate a GuardDuty delegated administrator on the **Welcome to GuardDuty** page.

     Note
     The management account must have the GuardDuty service-linked role in order for the delegated administrator to be able to enable and manage GuardDuty in that account. You can enable GuardDuty in any region of the management account to create this role automatically.
   - If GuardDuty is enabled, you can designate a GuardDuty delegated administrator on the **Settings** page.
3. Enter the twelve-digit AWS account ID of the account that you want to designate as the GuardDuty delegated administrator for the organization.
4. Choose **Delegate**. If GuardDuty is not already enabled, designating a delegated administrator will enable GuardDuty for that account in your current Region.
5. (Recommended) Repeat the previous steps in each AWS Region.

After you designate the delegated administrator, you only need to use the organization management account to change or remove the delegated administrator account.

Important
When you add an account as a member, GuardDuty is automatically enabled in that account in the current Region. This behaviour differs from the invitation method, in which GuardDuty must be enabled prior to the account being added as a member.

You must add your organization members in each Region to enable GuardDuty for those Regions.

Step 2 - add existing organization accounts as members

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation panel, choose **Settings**, and then choose **Accounts**.

   The accounts table displays all of the accounts in the organization. The **Type** for these accounts is **via organizations**. The status of accounts that are not member accounts associated with the organization's GuardDuty delegated administrator is **Not a member**.
3. Select the account or accounts that you want to add as members by checking the box next to the account ID.

   Note
   You can enable GuardDuty in the current Region for all organization accounts by selecting **enable** in the banner at the top of the page. This action also triggers the *Auto-Enable* feature that will enable GuardDuty in any future accounts added to your organization.

Alternately, you can use the **filter** field to filter by **Relationship status: Not a member**, and then select every account that does not have GuardDuty enabled in the current Region.

4. Choose **Actions** then choose **Add member**.
5. Confirm that you want to add as members the number of accounts selected. The **Status** for the accounts will change to **Enabled**.
6. (Recommended) Repeat these steps in each AWS Region to ensure that your delegated administrator can manage findings for member accounts in all Regions.

Step 3 - automate the addition of new organization accounts as members

1. Log in to the https://console.aws.amazon.com/guardduty/ console using the delegated administrator account.
2. In the navigation pane, under **Settings**, choose **Accounts**. Then select Auto-enable.
3. Select the first toggle icon to turn auto-enable on, if you wish to enable additional detection features for your new members in addition to enabling GuardDuty select from the **S3 Protection** or the **Kubernetes Audit Logs Monitoring** options. For more information about these features see Configuring S3 protection in multiple-account environments (p. 142) or Kubernetes protection in Amazon GuardDuty (p. 137) . When you have made updates choose **Update Settings** to finalize your changes.
4. (Recommended) Repeat these steps in each AWS Region to ensure that GuardDuty is automatically enabled on any new account, in every Region.

The auto-enable feature enables GuardDuty for all future members of your organization. This allows your GuardDuty delegated administrator to manage any new members that are created within or added to the organization. When the number of member accounts reaches the limit of 5000, the Auto-enable feature is automatically turned off. If an account is removed and the total number of members decreases to fewer than 5000, the Auto-enable feature turns back on.

API

Designate a delegated administrator and add member accounts (API)

1. Run the enableOrganizationAdminAccount API operation using the credentials of the AWS account of the Organizations management account.

   You can also use the AWS Command Line to do this by running the following CLI command. Make sure to specify the account ID of the account you want to make a GuardDuty delegated administrator.

   ```
   AWS  guardduty enable-organization-admin-account --admin-account-id 11111111111
   ```

   This command sets the delegated administrator for your current Region only. If GuardDuty is not already enabled for that account in the current Region, it will be automatically enabled.

   To set the delegated administrator for other Regions, you must specify the Region you want your delegated administrator to manage. For more information, see GuardDuty endpoints and quotas. The following example demonstrates how to enable a delegated administrator in US West (Oregon).

   ```
   AWS  guardduty enable-organization-admin-account --admin-account-id 11111111111 --
   region us-west-2
   ```

2. Run the CreateMembers API operation using the credentials of the AWS account you designated as the delegated administrator for GuardDuty in the previous step.

   You must specify the regional detector ID of the delegated administrator AWS account and the account details, including the account IDs and email addresses, of the accounts that you want to become GuardDuty members. You can create one or more members with this API operation.

   Important
   Accounts added as members will have GuardDuty enabled in that region, with the exception of the organization management account, which must first enable GuardDuty before it can be added as a member account.

   You can also do this using AWS Command Line Tools by running the following CLI command. Make sure to use your own valid detector ID, account ID, and email.

```
 AWS  guardduty create-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-details AccountId=123456789012,Email=guarddutymember@amazon.com
```

   You can view a list of all organization members using the ListAccounts API operation or by running the following CLI command.

```
 AWS  organizations list-accounts
```

3. Run the updateOrganizationConfiguration API operation using the credentials of the GuardDuty delegated administrator account to automatically enable GuardDuty in that Region for new member accounts.

   You must specify the detector ID of the delegated administrator AWS account.

   You can also do this using AWS Command Line Tools by running the following CLI command. Make sure to use your own valid detector ID.

```
 AWS  guardduty update-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable
```

   You can confirm that you have turned on the auto enable GuardDuty feature in a Region by running the describeOrganizationConfiguration API operation or by running the following CLI command using the detector ID of the delegated administrator in the desired Region.

```
 AWS  guardduty describe-organization-configuration —detector-
id 12abc34d567e8fa901bc2d34e56789f0
```

4. (Recommended) Repeat these steps in each Region using your unique detector ID for that Region to enable GuardDuty monitoring coverage for all members in all AWS Regions.

Consolidating GuardDuty administrator accounts under a single organization delegated administrator

GuardDuty recommends using association through AWS Organizations to manage member accounts under a delegated administrator account. You can use the example process outlined below to consolidate

administrator and member associated by invitation in an organization under a single GuardDuty delegated administrator.

Note
Accounts already being managed by a GuardDuty delegated administrator or delegated administrator accounts with active members cannot be added to a different GuardDuty delegated administrator account. Each organization can have only one GuardDuty delegated administrator account per region, and each member account can have only one delegated administrator.

1. Ensure all accounts you wish to manage GuardDuty for are part of your organization. For information on adding an account to your organization see Inviting an AWS account to join your organization.
2. Disassociate all member accounts from pre-existing administrator accounts, except those under the account you wish to designate as the GuardDuty delegated administrator for the organization.
3. Designate a GuardDuty delegated administrator for the organization from the **Settings** page.
4. Log in to the designated delegated admin account.
5. Proceed to add members from the organization.

Important
Remember that GuardDuty is a regional service. It is recommended that you designate your delegated administrator account and add all your members in every region to maximize the effectiveness of GuardDuty.

De-registering a GuardDuty delegated administrator

Note
Only the Organizations management account can de-register a delegated administrator.

Select Console or API and follow the provided steps to de-register your delegated administrator. Once de-registration is complete you can designate a new delegated administrator.

Console

When you de-register a delegated administrator from the console, if your account is also the Organizations management account you must repeat this process in each Region your account was designated as delegated administrator in.

Important
If you are the Organizations management account and have designated a different account as delegated administrator they will be de-registered in every Region.

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Select **Settings**.
3. From the Settings page, under **Delegated Administrator** choose **Remove**.
4. Confirm the change by selecting **Remove Administrator**.

API

When you de-register a delegated administrator from the API you must do so in every region before you can designate a new delegated administrator.

1. Run the DisableOrganizationAdminAccount API operation using the credentials of the Organizations management account.

```
aws guardduty disable-organization-admin-account ##admin-account-id "123456789012"
```

2. Repeat in each Region managed by that delegated administrator.

Managing GuardDuty accounts by invitation

To manage accounts outside of your organization, you can use the legacy invitation method. When you use this method, your account is designated as a administrator account when another account accepts your invitation to become a member account.

If your account is not a administrator account, you can accept an invitation from another account. When you accept, your account becomes a member account. An AWS account cannot be a GuardDuty administrator and member account at the same time.

Accounts associated by invitation have the same overall administrator-to-member relationship as accounts associated by AWS Organizations, as described in Understanding the relationship between GuardDuty administrator and member accounts (p. 124). However, invitation administrator account users cannot enable GuardDuty on behalf of associated member accounts or view other non-member accounts within their AWS Organizations organization.

Important
Cross-Regional data transfer may occur when GuardDuty creates member accounts using this method. In order to verify member accounts' email addresses, GuardDuty uses an email verification service that operates only in the US East (N. Virginia) Region.

Designating administrator and member accounts through invitation (console)

Use the following procedures to add an account, invite an account, or accept an invitation from another account.

Step 1 - add an account

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Accounts**.
3. Choose **Add accounts by invitation** in the top panel.
4. On the **Add member accounts** page, under **Enter accounts**, enter the AWS account ID and email address of the account that you want to add. Then choose **Add**.

   Important
   The email address that you specify in this step MUST be identical to the email address associated with the AWS account that you want to add as GuardDuty member account.

   You can add more accounts, one at a time, by specifying their IDs and email addresses. You can also choose **Upload list (.csv)** to bulk add accounts. This can be useful if you want to invite some of these accounts to enable GuardDuty right away but want to delay for others.

   Important
   The first line of your CSV file must contain the following header, as shown in the following example: **`Account ID,Email`**. Each subsequent line must contain a single valid account ID and a single valid email address for the account that you want to add. Accounts must appear one per line, and the account ID and email address must be separated by a comma.

   ```
   Account ID,Email
   111111111111,user@example.com
   ```

5. When you are finished adding accounts, choose **Next**.

The added accounts appear in a list on the **Accounts** page. Each added account in this list has an **Invite** link in the **Status** column.

Step 2 - invite an account

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Accounts**.
3. For the account that you want to invite to enable GuardDuty, choose the **Invite** link that appears in the **Status** column of the added accounts list.
4. In the **Invitation to GuardDuty** dialog box, enter an invitation message (optional), and then choose **Send notification**.

   Note
   If the invited account does not have access to email, select **Also send an email notification to the root user on the invitee's AWS account and generate an alert in the invitee's Personal Health Dashboard** before sending the invitation.

   The value in the **Status** column for the invited account changes to **Pending**.

Step 3 - accept an invitation

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Do one of the following:

   - If you don't have GuardDuty enabled, on the **Enable GuardDuty** page, choose **Enable GuardDuty**. Then use the **Accept** widget and the **Accept invitation** button to accept the membership invitation.

     Important
     You must enable GuardDuty before you can accept a membership invitation.
   - If you already have GuardDuty enabled, use the **Accept** widget and the **Accept invitation** button to accept the membership invitation.

   After you accept the invitation, your account becomes a GuardDuty member account. The account whose user sent the invitation becomes the GuardDuty administrator account. The administrator account user can see that the value in the **Status** column for your member account changes to **Monitored**. The administrator account user can now view and manage GuardDuty findings for your member account.

Designating GuardDuty administrator and member accounts through invitation (API)

You can designate administrator and member GuardDuty accounts by invitation through the API operations. Run the following GuardDuty API operations in order to designate administrator and member accounts in GuardDuty.

Complete the following procedure using the credentials of the AWS account that you want to designate as the GuardDuty administrator account.

1. Run the CreateMembers API operation using the credentials of the AWS account that has GuardDuty enabled. This is the account that you want to be the administrator GuardDuty account.

You must specify the detector ID of the current AWS account and the account ID and email address of the accounts that you want to become GuardDuty members. You can create one or more members with this API operation.

You can also use AWS Command Line Tools to designate a administrator account by running the following CLI command. Make sure to use your own valid detector ID, account ID, and email.

```
 AWS  guardduty create-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-details AccountId=123456789012,Email=guarddutymember@amazon.com
```

2. Run the InviteMembers API operation using the credentials of the AWS account that has GuardDuty enabled. This is the account that you want to be the administrator GuardDuty account.

   You must specify the detector ID of the current AWS account and the account IDs of the accounts that you want to become GuardDuty members. You can invite one or more members with this API operation.

   Note
   You can also specify an optional invitation message using the `message` request parameter.

   You can also use AWS Command Line Tools to designate member accounts by running the following CLI command. Make sure to use your own valid detector ID and valid account IDs for the accounts you want to invite.

```
 AWS  guardduty invite-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-ids 123456789012
```

Complete the following procedure using the credentials of each AWS account that you want to designate as a GuardDuty member account.

1. Run the CreateDetector API operation for each AWS account that was invited to become a GuardDuty member account and that you want to accept an invitation.

   You must specify if the detector resource is to be enabled using the GuardDuty service. A detector must be created and enabled in order for GuardDuty to become operational. You must first enable GuardDuty before accepting an invitation.

   You can also do this by using AWS Command Line Tools using the following CLI command.

```
 AWS  guardduty create-detector --enable
```

2. Run the AcceptInvitation API operation for each AWS account that you want to accept the membership invitation, using that account's credentials.

   You must specify the detector ID of this AWS account for the member account, the account ID of the administrator account that sent the invitation, and the invitation ID of the invitation that you are accepting. You can find the account ID of the administrator account in the invitation email or by using the `ListInvitations` operation of the API.

   You can also accept an invitation using AWS Command Line Tools by running the following CLI command. Make sure to use a valid detector ID, administrator account ID, and invitation ID.

```
 AWS  guardduty accept-invitation --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
master-id 012345678901 --invitation-id 84b097800250d17d1872b34c4daadcf5
```

Enable GuardDuty in multiple accounts simultaneously

Use the following method to enable GuardDuty in multiple accounts at the same time.

Use Python scripts to enable GuardDuty in multiple accounts simultaneously

You can automate the enabling or disabling of GuardDuty on multiple accounts using the scripts from the sample repository on GitHub at https://github.com/aws-samples/amazon-guardduty-multiaccount-scripts. Use the process in this section to enable GuardDuty for a list of member accounts using Amazon EC2. For information about using the disable script or setting up the script locally refer, to the GitHub instructions.

The enableguardduty.py script enables GuardDuty, sends invitations from the administrator account, and accepts invitations in all member accounts. The result is a administrator GuardDuty account that contains all security findings for all member accounts. Because GuardDuty is isolated by Region, findings for each member account roll up to the corresponding Region in the administrator account. For example, the us-east-1 Region in your GuardDuty administrator account contains the security findings for all us-east-1 findings from all associated member accounts.

These scripts have a dependency on a shared IAM role with the managed policy **AmazonGuardDutyFullAccess**. This policy provides entities access to GuardDuty and must be present on the administrator account and in each account for which you want to enable GuardDuty.

The following process enables GuardDuty in all available Regions by default. You can enable GuardDuty in specified Regions only by using the optional `--enabled_regions` argument and providing a comma-separated list of Regions. You can also optionally customize the invitation message that is sent to member accounts by opening the `enableguardduty.py` and editing the `gd_invite_message` string.

1. Create an IAM role in the GuardDuty administrator account and attach the **AmazonGuardDutyFullAccess** managed policy with the following permissions.

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "guardduty:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "guardduty.amazonaws.com"
                }
            }
        }
    ]
}
```

2. Create an IAM role in each member account you want to be managed by your GuardDuty administrator account. This role must have the same name as the role created in step 1,

it should allow the administrator account as a trusted entity, and it should have the same **AmazonGuardDutyFullAccess** managed policy described previously.

3. Launch a new Amazon Linux instance with an attached role that has the following trust relationship that allows the instance to assume a service role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Log in to the new instance and run the following commands to set it up.

```
sudo yum install git python
sudo yum install python-pip
pip install boto3
 AWS  configure
git clone https://github.com/aws-samples/amazon-guardduty-multiaccount-scripts.git
cd amazon-guardduty-multiaccount-scripts
sudo chmod +x disableguardduty.py enableguardduty.py
```

5. Create a CSV file containing a list of account IDs and emails of the member accounts that you added a role to in step 2. Accounts must appear one per line, and the account ID and email address must be separated by a comma as in the following example.

```
111111111111,user@example.com
```

Note
The CSV file must be in the same location as your `enableguardduty.py` script. You can copy an existing CSV file from Amazon S3 to your current directory with the following method.

```
AWS  s3 cp s3://my-bucket/my_key_name example.csv
```

6. Run the Python script. Make sure to supply your GuardDuty administrator account ID, the name of the role created in the first steps, and the name of your CSV file as arguments.

```
python enableguardduty.py --master_account 111111111111 --assume_role roleName
 accountID.csv
```

Kubernetes protection in Amazon GuardDuty

Kubernetes protection enables Amazon GuardDuty to detect suspicious activities and potential compromises of your Kubernetes clusters within Amazon Elastic Kubernetes Service (Amazon EKS).

Kubernetes protection is an optional enhancement that enables GuardDuty to consume Kubernetes data sources. The process for enabling or disabling Kubernetes protection is covered in this topic.

We strongly recommend that you do not disable Kubernetes protection in GuardDuty. If the feature is not enabled, the ability of GuardDuty to monitor or generate findings for suspicious activity within your Amazon EKS environment is limited.

Understanding how GuardDuty uses Kubernetes data sources

When Kubernetes protection is enabled, GuardDuty uses optional data sources to detect threats against Kubernetes API. Currently the following data sources can be ingested with Kubernetes protection enabled:

Kubernetes audit logs

Kubernetes audit logs are a feature of all Kubernetes clusters that capture chronological API activity from users, applications, and the control plane. When Kubernetes protection is enabled, GuardDuty ingests these logs from Amazon EKS to produce Kubernetes findings for your Amazon EKS resources without requiring you to turn on or store these logs. For more information see Kubernetes audit logs (p. 11)

When Kubernetes protection is enabled, GuardDuty immediately begins to analyze Kubernetes data sources from your Amazon EKS clusters and monitor them for malicious and suspicious activity. For more information, see How Amazon GuardDuty uses its data sources (p. 10).

If you disable Kubernetes protection, GuardDuty immediately stops consuming this data source and stops monitoring your EKS clusters.

Configuring Kubernetes protection for a standalone account

You can disable or enable Kubernetes protection through the console or the `UpdateDetector` API operation.

To configure Kubernetes protection for your account, see the following configuration options.

To enable or disable Kubernetes protection

Choose one of the following access methods for instructions on enabling or disabling Kubernetes protection for a standalone account.

Console

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. In the navigation pane, under **Settings**, choose Kubernetes protection.
3. The Kubernetes protection pane lists the current status of Kubernetes protection for your account. You may enable or disable it at any time by selecting **Enable** or **Disable** respectively, then confirming your selection.

API

- Run the updateDetector API operation using your own Regional detector ID and passing the `dataSources` object with `[["Kubernetes Logs":"enable"]]` set to true or false to enable or disable.

  You can also enable or disable Kubernetes protection using AWS command line tools by running the following AWS CLI command. Make sure to use your own valid detector ID.

  Note
  The following example code enables Kubernetes protection. To disable it, replace `true` with `false`.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
data-sources '{"Kubernetes":{"AuditLogs":{"Enable":true}}}'
```

Configuring Kubernetes protection in multiple-account environments

In a multi-account environment, only GuardDuty delegated administrator accounts can configure Kubernetes protection. GuardDuty delegated administrators can enable or disable Kubernetes protection for their member accounts. GuardDuty member accounts cannot enable or disable this data source.

GuardDuty administrator accounts that manage their member accounts with AWS Organizations can choose to have Kubernetes protection automatically enabled on all new accounts in the organization. For more information, see Managing GuardDuty accounts with AWS Organizations (p. 126).

Automatically enabling Kubernetes protection for Organization member accounts

Note
This functionality is only available to GuardDuty delegated administrators with members incorporated through AWS Organizations.

1. Log in to the https://console.aws.amazon.com/guardduty/ console using the administrator account.
2. In the navigation pane, under **Settings**, choose **Accounts**.
3. Ensure that **Auto-enable** for GuardDuty is turned on. If it is off, you can enable it by choosing **Enable** from the banner or by choosing **Auto-enable is OFF**. This feature automatically enables GuardDuty for new member accounts when they join your organization and must be turned on to auto-enable Kubernetes protection.
4. Once Auto-enable for GuardDuty is on, you can enable Kubernetes protection for your new members by selecting the Kubernetes protection toggle icon. Choose **Update Settings** to confirm.

To manually enable or disable Kubernetes protection in member accounts

Choose your access method below for instructions on enabling or disabling Kubernetes protection for member accounts.

Console

To enable Kubernetes protection for all accounts

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. If you want to enable Kubernetes protection for all accounts at once, choose Kubernetes protection from the navigation pane.
3. You will see a statement reflecting the number of accounts you manage that have Kubernetes protection enabled. Choose **Enable all** to enable Kubernetes protection for all accounts.

   Note
   If you manage accounts within an organization, this action also enables the **Auto-enable** feature to automatically enable Kubernetes protection for future member accounts within your organization.

To manually enable or disable Kubernetes protection in member accounts

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. In the navigation pane, under **Settings**, choose **Accounts**.

   Note
   From the **Accounts** table, review the Kubernetes protection column. A green checkmark icon indicates that Kubernetes protection is enabled, and a blue dash icon indicates that it is disabled. If this column is blank, the account is not eligible for Kubernetes protection. You can also filter by **ENABLED** or **DISABLED**.
3. Select the account for which you want to configure Kubernetes protection. From the **Actions** menu choose **Enable Kubernetes protection** or **Disable Kubernetes protection**. Then confirm your selection to change the settings for the selected account. The table updates automatically to show your changes.

API

To selectively enable or disable Kubernetes protection for your member accounts, run the updateMemberDetectors API operation using your own detector ID. The following example shows how you can enable Kubernetes protection for a single member account. To disable it, replace `true` with `false`.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-ids 123456789012 --data-sources '{"Kubernetes":{"AuditLogs":{"Enable":true}}}'
```

Note
You can also pass a list of account IDs separated by a space.

When the code has successfully run, it returns an empty list of `UnprocessedAccounts`. If there were any problems changing the detector settings for an account, that account ID is listed along with a summary of the issue.

Note
If you use scripts to onboard new accounts and want to disable Kubernetes protection in your new accounts, you can modify the createDetector API operation with the optional `dataSources` object as described in this topic.

Automatically disabling Kubernetes protection for new GuardDuty accounts

Important
By default, Kubernetes protection is enabled automatically for all GuardDuty accounts.

If you are a GuardDuty administrator enabling GuardDuty for the first time on a new account, and you do not want Kubernetes protection enabled by default, you can disable it by modifying the createDetector API operation with the optional `dataSources` object. The following example uses the AWS CLI to enable a new GuardDuty detector with the Kubernetes protection disabled.

```
aws guardduty create-detector --enable --data-sources '{"Kubernetes":{"AuditLogs":
{"Enable":false}}}'
```

Amazon S3 protection in Amazon GuardDuty

S3 protection enables Amazon GuardDuty to monitor object-level API operations to identify potential security risks for data within your S3 buckets.

GuardDuty monitors threats against your Amazon S3 resources by analyzing AWS CloudTrail management events and CloudTrail S3 data events. These data sources monitor different kinds of activity, for example, CloudTrail management events for S3 include operations that list or configure S3 buckets, such as `ListBuckets`, `DeleteBuckets`, and `PutBucketReplication`. Examples of data events for S3 include object-level API operations, such as `GetObject`, `ListObjects`, `DeleteObject`, and `PutObject`.

GuardDuty monitoring of CloudTrail management events is on by default for all accounts that have enabled GuardDuty and is not configurable. CloudTrail S3 data event logs are a configurable data source in GuardDuty. By default, S3 protection is enabled for new detectors, for accounts created before the addition of S3 protection this data source must be enabled manually. The processes for enabling or disabling S3 data event monitoring is covered in this topic.

We strongly recommend that you enable S3 protection in GuardDuty. If the feature is disabled, GuardDuty is unable to fully monitor or generate findings for suspicious access to data stored in your S3 buckets.

Understanding how GuardDuty uses S3 data events

The S3 protection feature in GuardDuty refers to whether S3 data events are enabled as a data source for GuardDuty. When S3 data event monitoring is enabled GuardDuty immediately begins to analyze S3 data events from all of your S3 buckets and monitor them for malicious and suspicious activity. For more information, see How Amazon GuardDuty uses its data sources (p. 10).

GuardDuty does not process requests to objects that you have made publicly accessible, but it does alert you when a bucket is made publicly accessible. When GuardDuty detects a threat based on S3 data event monitoring, it generates a security finding. For information about the types of findings GuardDuty can generate for Amazon S3 see GuardDuty S3 finding types (p. 44).

If you disable S3 protection, GuardDuty immediately stops consuming this data source and stops monitoring access to data stored in your S3 buckets.

Configuring S3 protection for a standalone account

For accounts associated by AWS Organizations, this process can be automated through console settings as described in the next section.

Accounts that were using GuardDuty before the addition of S3 protection can enable the new data source by configuring GuardDuty through the console or the `UpdateDetector` API operation.

To configure Amazon S3 data events as a data source for your account, see the following configuration options.

To enable or disable S3 protection

Choose your access method below for instructions on enabling or disabling S3 protection for a standalone account.

Console

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. In the navigation pane, under **Settings**, choose **S3 Protection**.
3. The **S3 Protection** pane lists the current status of S3 protection for your account. You may enable or disable it at any time by selecting **Enable** or **Disable** respectively, then confirming your selection.

API

- Run the updateDetector API operation using your own regional detector ID and passing the `dataSources` object with `"S3 Logs":"enable"` set to true or false, respectively.

  You can also enable or disable S3 protection using AWS command line tools by running the following AWS CLI command. Make sure to use your own valid detector ID.

  Note
  The following example code enables S3 protection. To disable it, replace `true` with `false`.

```
 AWS  guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
data-sources '{"S3Logs":{"Enable":true}}'
```

Configuring S3 protection in multiple-account environments

In a multi-account environment, only GuardDuty administrator accounts can configure S3 protection. GuardDuty administrator accounts can enable or disable S3 protection for their member accounts. GuardDuty member accounts cannot enable or disable this data source.

GuardDuty administrator accounts that manage their member accounts with AWS Organizations support can choose to have S3 protection automatically enabled on all new accounts in the organization. For more information, see Managing GuardDuty accounts with AWS Organizations (p. 126).

Automatically enabling S3 protection for Organization member accounts

Note
This functionality is only available to administrators of GuardDuty members incorporated through AWS Organizations.

1. Log in to the https://console.aws.amazon.com/guardduty/ console using the administrator account.
2. In the navigation pane, under **Settings**, choose **Accounts**.

3. Ensure **Auto-enable** for GuardDuty is turned on. If it is off you can enable by selecting **Enable** from the banner or by selecting **Auto-enable is OFF**. This feature will automatically enable GuardDuty for new member accounts within your organization and must be enabled in order to auto-enable S3 protection.
4. Once Auto-enable for GuardDuty is on, you can enable S3 protection for your new members in addition to enabling GuardDuty by selecting the **S3 Protection** toggle icon. Choose **Update Settings** to confirm.

To selectively enable or disable S3 protection in member accounts

Choose your access method below for instructions on enabling or disabling S3 protection for member accounts.

Console

To enable S3 protection for all accounts

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. If you want to enable S3 protection for all accounts at once, choose **S3 Protection** from the navigation pane.
3. You will see a statement reflecting the number of accounts you manage that have S3 protection enabled. Choose **Enable all** to enable S3 protection for all accounts.

   Note
   If you manage accounts within an organization, this action also enables the **Auto-enable** feature to automatically enable S3 protection for future member accounts within your organization.

To selectively enable or disable S3 protection in member accounts

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. In the navigation pane, under **Settings**, choose **Accounts**.

   Note
   From the Accounts table, review the **S3 Protection** column. A green checkmark icon indicates that S3 protection is enabled, and a blue dash icon indicates that it is disabled. If this column is blank, the account is not eligible for S3 protection. You can also filter by **ENABLED** or **DISABLED**.
3. Select the account for which you want to configure S3 protection. From the **Actions** menu choose **Enable S3 Protection** or **Disable S3 Protection**, then confirm your selection to change the settings for the selected account. The table will update automatically to show your changes.

API

To selectively enable or disable S3 protection for your member accounts, run the updateMemberDetectors API operation using your own detector ID. The following example shows how you can enable S3 protection for a single member account. To disable it, replace `true` with `false`.

```
AWS  guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
--account-ids 123456789012 --data-sources '{"S3Logs":{"Enable":true}}'
```

Note
You can also pass a list of account IDs separated by a space.

When the code has successfully executed, it returns an empty list of `UnprocessedAccounts`. If there were any problems changing the detector settings for an account, that account ID is listed along with a summary of the issue.

Note
If you use scripts to on-board new accounts and wish to disable S3 protection in your new accounts, you can modify the createDetector API operation with the optional `dataSources` object as described in this topic.

Automatically disabling S3 protection for new GuardDuty accounts

Important
By default, S3 protection is enabled automatically for new detectors.

If you are a GuardDuty administrator enabling GuardDuty for the first time on a new account, and you do not want S3 protection enabled by default, you can disable it by modifying the createDetector API operation with the optional `dataSources` object. The following example uses the AWS CLI to enable a new GuardDuty detector with the S3 protection disabled.

```
AWS  guardduty create-detector --enable --data-sources '{"S3Logs":{"Enable":false}}'
```

Estimating GuardDuty costs

You can use the GuardDuty console and API operations to estimate how much GuardDuty will cost you per month. During the 30-day free trial period, the cost estimation projects what your estimated costs will be after the free trial period. If you are operating in a multi-account environment, your GuardDuty administrator account can monitor cost metrics for all of your member accounts.

You can view cost estimation based on the following metrics:

- **Account ID** - Lists the estimated cost for your account, or for your member accounts if you are operating as a GuardDuty administrator account.
- **Data source** - Lists the estimated cost on the specified data source for the following GuardDuty data source types: VPC Flow Logs, CloudTrail Management logs, S3 Data Event logs, or DNS logs.
- **S3 buckets** - Lists the estimated cost for S3 data events on a specified bucket or the most expensive buckets for accounts in your environment.

  Note
  S3 bucket statistics are only available if S3 Protection is enabled for the account. For more information see Amazon S3 protection in Amazon GuardDuty (p. 141).

Understanding how usage costs are calculated

When you use the cost monitoring feature of GuardDuty, it is important to understand how the estimates are calculated. The estimates displayed in GuardDuty may differ slightly from those in your Billing and Cost Management console. Note the following about how GuardDuty cost estimates are calculated.

- The GuardDuty usage estimate is for the current Region only.
- The GuardDuty usage estimate is an average daily cost based on the past 7 to 30 days of usage.

  Note
  For newly enabled detectors or data sources with fewer than seven days of usage, the cost is listed as **Pending**.
- The free trial estimate reflects only those data sources currently in a free trial period.
- The GuardDuty usage estimate includes GuardDuty volume pricing discounts per Region, as detailed on the Amazon GuardDuty Pricing page, but only for individual accounts meeting the volume pricing tiers. Volume pricing discounts are not included in estimates for combined total usage between accounts within an Organization. For information about combined usage volume discount pricing, see AWS Billing: Volume Discounts.

Review GuardDuty usage statistics (Console)

1. Log into the https://console.aws.amazon.com/guardduty/ console using the GuardDuty administrator account.
2. In the navigation pane, choose **Usage**.
3. GuardDuty administrator accounts with members see a list of all managed accounts. Single accounts see a breakdown by data source.

If you have member accounts, you can view statistics for an individual account by selecting that account in the Accounts table. If S3 protection is enabled for the selected account, the top S3 buckets by usage cost are displayed in the **By data source** panel.

Note
A green dot indicates that a free trial period is active.

Review GuardDuty usage statistics (API)

To view cost metrics, run the getUsageStatistics API operation using the credentials of the GuardDuty administrator account. Supply the following information to run the command:

- (Required) Specify the Regional GuardDuty detector ID of the account you want to retrieve statistics for.
- (Required) Specify the type of statistics to retrieve: `SUM_BY_ACCOUNT | SUM_BY_DATA_SOURCE | SUM_BY_RESOURCE | TOP_RESOURCES`.
- (Required) Specify at least one data source to query from the following options: `FLOW_LOGS | CLOUD_TRAIL | DNS_LOGS | S3_LOGS`.
- (Optional) Specify a list of account IDs to retrieve usage statistics for.

You can also use the AWS Command Line. Run the command below, replacing the example detector ID with your own, to get the sum of usage for all data sources. For single accounts, this command returns the cost over the past 30 days for your account only. If you are a GuardDuty administrator with member accounts, you see costs listed by account for all members.

```
 AWS  guardduty get-usage-statistics --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --usage-statistic-type SUM_BY_ACCOUNT --usage-criteria '{"DataSources":
["S3_LOGS","CLOUD_TRAIL","DNS_LOGS","FLOW_LOGS"]}'
```

Security in Amazon GuardDuty

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The Shared Responsibility Model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to GuardDuty, see AWS services in scope by compliance program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using GuardDuty. It shows you how to configure GuardDuty to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your GuardDuty resources.

Contents

Data protection in Amazon GuardDuty

The AWS shared responsibility model applies to data protection in Amazon GuardDuty. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.

- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with GuardDuty or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

All GuardDuty customer data is encrypted at rest using AWS encryption solutions.

GuardDuty data, such as findings, is encrypted at rest using AWS Key Management Service (AWS KMS) using AWS owned customer master keys (CMK).

Encryption in transit

GuardDuty analyzes log data from other services. GuardDuty encrypts all data in transit from these services with HTTPS and KMS. Once GuardDuty extracts the information it needs from the logs, they are discarded. For more information on how GuardDuty uses information from other services see GuardDuty data sources.

GuardDuty data is encrypted in transit between services.

Logging Amazon GuardDuty API calls with AWS CloudTrail

Amazon GuardDuty is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in GuardDuty. CloudTrail captures API calls for GuardDuty as events, including calls from the GuardDuty console and from code calls to the GuardDuty APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for GuardDuty. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to GuardDuty, the IP address the request was made from, who made the request, when it was made, and more.

To learn more about CloudTrail, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

GuardDuty information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in GuardDuty, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail event history.

For an ongoing record of events in your AWS account, including events for GuardDuty, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All GuardDuty actions are logged by CloudTrail and are documented in the GuardDuty API reference.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity element.

Example: GuardDuty log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateIPThreatIntelSet` action.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Alice",
        "accountId": "444455556666",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-14T22:54:20Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::444455556666:user/Alice",
                "accountId": "444455556666",
                "userName": "Alice"
            }
        }
```

```
    },
    "eventTime": "2018-06-14T22:57:56Z",
    "eventSource": "guardduty.amazonaws.com",
    "eventName": "CreateThreatIntelSet",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "54.240.230.177",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "detectorId": "5ab04b1110c865eecf516eee2435ede7",
        "name": "Example",
        "format": "TXT",
        "activate": false,
        "location": "https://s3.amazonaws.com/bucket.name/file.txt"
    },
    "responseElements": {
        "threatIntelSetId": "1ab200428351c99d859bf61992460d24"
    },
    "requestID": "5f6bf981-7026-11e8-a9fc-5b37d2684c5c",
    "eventID": "81337b11-e5c8-4f91-b141-deb405625bc9",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "444455556666"
}
```


From this event information, you can determine that the request was made to create a threat list `Example` in GuardDuty. You can also see that the request was made by an IAM user named Alice on June 14, 2018.

Identity and Access Management for AWS GuardDuty

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use GuardDuty resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in GuardDuty.

Service user – If you use the GuardDuty service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more GuardDuty features to do your work, you

might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in GuardDuty, see Troubleshooting AWS GuardDuty identity and access (p. 165).

Service administrator – If you're in charge of GuardDuty resources at your company, you probably have full access to GuardDuty. It's your job to determine which GuardDuty features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with GuardDuty, see How AWS GuardDuty works with IAM (p. 154).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to GuardDuty. To view example GuardDuty identity-based policies that you can use in IAM, see Identity-based policy examples for AWS GuardDuty (p. 159).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS GuardDuty in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS GuardDuty works with IAM

Before you use IAM to manage access to GuardDuty, learn what IAM features are available to use with GuardDuty.

IAM features you can use with AWS GuardDuty

IAM feature	GuardDuty support
Identity-based policies (p. 155)	Yes
Resource-based policies (p. 155)	No
Policy actions (p. 156)	Yes
Policy resources (p. 156)	Yes
Policy condition keys (p. 157)	Yes
ACLs (p. 157)	No

IAM feature	GuardDuty support
ABAC (tags in policies) (p. 157)	Partial
Temporary credentials (p. 158)	Yes
Principal permissions (p. 158)	Yes
Service roles (p. 158)	Yes
Service-linked roles (p. 159)	Yes

To get a high-level view of how GuardDuty and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for GuardDuty

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for GuardDuty

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for AWS GuardDuty (p. 159).

Resource-based policies within GuardDuty

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional

identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for GuardDuty

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of GuardDuty actions, see Actions defined by AWS GuardDuty in the *Service Authorization Reference*.

Policy actions in GuardDuty use the following prefix before the action:

```
guardduty
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "guardduty:action1",
      "guardduty:action2"
         ]
```

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for AWS GuardDuty (p. 159).

Policy resources for GuardDuty

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of GuardDuty resource types and their ARNs, see Resources defined by AWS GuardDuty in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS GuardDuty.

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for AWS GuardDuty (p. 159).

Policy condition keys for GuardDuty

Supports policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of GuardDuty condition keys, see Condition keys for AWS GuardDuty in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS GuardDuty.

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for AWS GuardDuty (p. 159).

Access control lists (ACLs) in GuardDuty

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with GuardDuty

Supports ABAC (tags in policies)	Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with GuardDuty

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for GuardDuty

Supports principal permissions	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS GuardDuty in the *Service Authorization Reference*.

Service roles for GuardDuty

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Warning
Changing the permissions for a service role might break GuardDuty functionality. Edit service roles only when GuardDuty provides guidance to do so.

Service-linked roles for GuardDuty

Supports service-linked roles	Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing GuardDuty service-linked roles, see Using service-linked roles for Amazon GuardDuty (p. 167).

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS GuardDuty

By default, IAM users and roles don't have permission to create or modify GuardDuty resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

Topics

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete GuardDuty resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using GuardDuty quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as

necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.

- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Using the GuardDuty console

To access the Amazon GuardDuty console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the GuardDuty resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

To ensure that users and roles can still use the GuardDuty console, also attach the GuardDuty `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Permissions required to enable GuardDuty

This section describes the permissions that various IAM identities (users, groups, and roles) must have in order to initially enable GuardDuty either through the console or programmatically (using the GuardDuty API or the GuardDuty commands in the AWS CLI).

To grant permissions required to enable GuardDuty, attach the following policy to an IAM user, group, or role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "guardduty:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::123456789012:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "guardduty.amazonaws.com"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::1234567890123:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty"
        }
    ]
}
```

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Custom IAM policy to grant read-only access to GuardDuty

To grant read-only access to GuardDuty you can use the `AmazonGuardDutyReadOnlyAccess` managed policy.

To create a custom policy that grants an IAM user, role, or group read-only access to GuardDuty you can use the following statement:

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "guardduty:ListMembers",
                "guardduty:GetMembers",
                "guardduty:ListInvitations",
                "guardduty:ListDetectors",
                "guardduty:GetDetector",
                "guardduty:ListFindings",
                "guardduty:GetFindings",
                "guardduty:ListIPSets",
                "guardduty:GetIPSet",
                "guardduty:ListThreatIntelSets",
                "guardduty:GetThreatIntelSet",
                "guardduty:GetMasterAccount",
                "guardduty:GetInvitationsCount",
                "guardduty:GetFindingsStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

Deny Access to GuardDuty findings

You can use the following policy to deny an IAM user, role, or group access to GuardDuty findings. Users can't view findings or the details about findings, but they can access all other GuardDuty operations:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "guardduty:CreateDetector",
                "guardduty:DeleteDetector",
                "guardduty:UpdateDetector",
                "guardduty:GetDetector",
                "guardduty:ListDetectors",
                "guardduty:CreateIPSet",
                "guardduty:DeleteIPSet",
                "guardduty:UpdateIPSet",
                "guardduty:GetIPSet",
                "guardduty:ListIPSets",
                "guardduty:CreateThreatIntelSet",
                "guardduty:DeleteThreatIntelSet",
                "guardduty:UpdateThreatIntelSet",
                "guardduty:GetThreatIntelSet",
                "guardduty:ListThreatIntelSets",
                "guardduty:ArchiveFindings",
                "guardduty:UnarchiveFindings",
                "guardduty:CreateSampleFindings",
                "guardduty:CreateMembers",
                "guardduty:InviteMembers",
                "guardduty:GetMembers",
                "guardduty:DeleteMembers",
                "guardduty:DisassociateMembers",
                "guardduty:StartMonitoringMembers",
                "guardduty:StopMonitoringMembers",
                "guardduty:ListMembers",
                "guardduty:GetMasterAccount",
                "guardduty:DisassociateFromMasterAccount",
                "guardduty:AcceptInvitation",
```

```
                "guardduty:ListInvitations",
                "guardduty:GetInvitationsCount",
                "guardduty:DeclineInvitations",
                "guardduty:DeleteInvitations"
            ],
            "Resource": "*"
        },
         {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::123456789123:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "guardduty.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::123456789123:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty"
        }
    ]
}
```

Using a custom IAM policy to limit access to GuardDuty resources

To define a user's access to GuardDuty based on the detector ID, you can use all GuardDuty API actions in your custom IAM policies, **except** the following operations:

- `guardduty:CreateDetector`
- `guardduty:DeclineInvitations`
- `guardduty:DeleteInvitations`
- `guardduty:GetInvitationsCount`
- `guardduty:ListDetectors`
- `guardduty:ListInvitations`

Use the following operations in an IAM policy to define a user's access to GuardDuty based on the IPSet ID and ThreatIntelSet ID:

- `guardduty:DeleteIPSet`
- `guardduty:DeleteThreatIntelSet`
- `guardduty:GetIPSet`
- `guardduty:GetThreatIntelSet`
- `guardduty:UpdateIPSet`
- `guardduty:UpdateThreatIntelSet`

The following examples show how to create policies using some of the preceding operations:

- This policy allows a user to run the `guardduty:UpdateDetector` operation, using the detector ID of 1234567 in the us-east-1 Region:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateDetector",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:012345678910:detector/1234567"
        }
    ]
}
```

- This policy allows a user to run the `guardduty:UpdateIPSet` operation, using the detector ID of 1234567 and the IPSet ID of 000000 in the us-east-1 Region:

  Note
  Make sure that the user has the permissions required to access trusted IP lists and threat lists in GuardDuty. For more information, see Permissions required to upload trusted IP lists and threat lists (p. 98).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:012345678910:detector/1234567/
ipset/000000"
        }
    ]
}
```

- This policy allows a user to run the `guardduty:UpdateIPSet` operation, using any detector ID and the IPSet ID of 000000 in the us-east-1 Region:

  Note
  Make sure that the user has the permissions required to access trusted IP lists and threat lists in GuardDuty. For more information, see Permissions required to upload trusted IP lists and threat lists (p. 98).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:012345678910:detector/*/
ipset/000000"
        }
    ]
}
```

- This policy allows a user to run the `guardduty:UpdateIPSet` operation, using the detector ID of 1234567 and any IPSet ID in the us-east-1 Region:

Note
Make sure that the user has the permissions required to access trusted IP lists and threat lists in GuardDuty. For more information, see Permissions required to upload trusted IP lists and threat lists (p. 98).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:012345678910:detector/1234567/
ipset/*"
        }
    ]
}
```

Troubleshooting AWS GuardDuty identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with GuardDuty and IAM.

Topics

I am not authorized to perform an action in GuardDuty

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `guardduty:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 guardduty:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `guardduty:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to GuardDuty.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in GuardDuty. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access GuardDuty

To allow others to access GuardDuty, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in GuardDuty.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my GuardDuty resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether GuardDuty supports these features, see How AWS GuardDuty works with IAM (p. 154).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.

- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Using service-linked roles for Amazon GuardDuty

Amazon GuardDuty uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to GuardDuty. Service-linked roles are predefined by GuardDuty and include all the permissions that GuardDuty requires to call other AWS services on your behalf.

A service-linked role makes setting up GuardDuty easier because you don't have to manually add the necessary permissions. GuardDuty defines the permissions of its service-linked role, and unless the permissions are defined otherwise, only GuardDuty can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

GuardDuty supports using service-linked roles in all of the Regions where GuardDuty is available. For more information, see Regions and endpoints (p. 191).

You can delete the GuardDuty service-linked role only after first disabling GuardDuty in all Regions where it is enabled. This protects your GuardDuty resources because you can't inadvertently remove permission to access them.

For information about other services that support service-linked roles, see AWS services that work with IAM in the *IAM User Guide* and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for GuardDuty

GuardDuty uses the service-linked role named `AWSServiceRoleForAmazonGuardDuty`. This service-linked role allows GuardDuty to retrieve metadata for the EC2 instances in your AWS environment that are involved in potentially suspicious activity. It also allows GuardDuty to include the retrieved EC2 instance metadata in the findings that GuardDuty generates about potentially suspicious activity. The `AWSServiceRoleForAmazonGuardDuty` service-linked role trusts the `guardduty.amazonaws.com` service to assume the role.

The permissions policy for the role allows GuardDuty to perform tasks such as:

- Use Amazon EC2 actions to retrieve information about your EC2 instances and images.
- Use Amazon EC2 actions to retrieve information about your EC2 networking components such as VPCs, subnets, and transit gateways.
- Use Amazon S3 actions to retrieve information about S3 buckets and objects.
- Use AWS Organizations actions to describe associated accounts.

The role is configured with the following AWS managed policy, named `AWSServiceRoleForAmazonGuardDuty`.

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

```
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeImages",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcPeeringConnections",
        "ec2:DescribeTransitGatewayAttachments"
        "organizations:ListAccounts",
        "organizations:DescribeAccount",
        "s3:GetBucketPublicAccessBlock",
        "s3:GetEncryptionConfiguration",
        "s3:GetBucketTagging",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAllMyBuckets",
        "s3:GetBucketAcl",
        "s3:GetBucketPolicy",
        "s3:GetBucketPolicyStatus",
      ],
      "Resource": "*"
    }
  ]
}
```

The following is the trust policy that is attached to the `AWSServiceRoleForAmazonGuardDuty` service-linked role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "guardduty.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating a service-linked role for GuardDuty

The `AWSServiceRoleForAmazonGuardDuty` service-linked role is automatically created when you enable GuardDuty for the first time or enable GuardDuty in a supported Region where you previously didn't have it enabled. You can also create the `AWSServiceRoleForAmazonGuardDuty` service-linked role manually using the IAM console, the IAM CLI, or the IAM API.

> **Important**
> The service-linked role that is created for the GuardDuty delegated administrator account doesn't apply to the member GuardDuty accounts.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For the `AWSServiceRoleForAmazonGuardDuty` service-linked role to be successfully created, the IAM identity that you use GuardDuty with must have the required permissions. To grant the required permissions, attach the following policy to this IAM user, group, or role:

> **Note**
> Replace the sample account ID in the example below with your actual AWS account ID.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "guardduty:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::123456789012:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "guardduty.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::1234567890123:role/aws-service-role/
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty"
        }
    ]
}
```

For more information about creating the role manually, see Creating a service-linked role in the *IAM User Guide*.

Editing a service-linked role for GuardDuty

GuardDuty doesn't allow you to edit the `AWSServiceRoleForAmazonGuardDuty` service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for GuardDuty

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that isn't actively monitored or maintained.

> **Important**
> You must first disable GuardDuty in all Regions where it is enabled in order to delete the `AWSServiceRoleForAmazonGuardDuty`.
> If the GuardDuty service isn't disabled when you try to delete the service-linked role, the deletion fails. For more information, see Suspending or disabling GuardDuty (p. 183).

When you disable GuardDuty, the `AWSServiceRoleForAmazonGuardDuty` is NOT automatically deleted. If you then enable GuardDuty again, it'll start using the existing `AWSServiceRoleForAmazonGuardDuty`.

To manually delete the service-linked role using IAM

Use the IAM console, the IAM CLI, or the IAM API to delete the `AWSServiceRoleForAmazonGuardDuty` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

AWS managed policies for Amazon GuardDuty

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AmazonGuardDutyFullAccess

You can attach the `AmazonGuardDutyFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow a user full access to all GuardDuty actions.

Permissions details

This policy includes the following permissions.

- `GuardDuty` – Allows users full access to all GuardDuty actions.
- `IAM` – Allows users to create the GuardDuty service-linked role. This allows a GuardDuty administrator to enable GuardDuty for member accounts.
- `Organizations` – Allows users to designate a delegated administrator and manage members for a GuardDuty organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "guardduty:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
```

```
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "guardduty.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:RegisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policy: AmazonGuardDutyReadOnlyAccess

You can attach the `AmazonGuardDutyReadOnlyAccess` policy to your IAM identities.

This policy grants read-only permissions that allow a user to view GuardDuty findings and details of your GuardDuty organization.

Permissions details

This policy includes the following permissions.

- `GuardDuty` – Allows users to view GuardDuty findings and perform API operations that start with `Get`, `List`, or `Describe`.
- `Organizations` – Allows users to retrieve information about your GuardDuty organization configuration, including details of the delegated administrator account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "guardduty:Describe*",
                "guardduty:Get*",
                "guardduty:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
```

```
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policy: AWSServiceRoleForAmazonGuardDuty

You can't attach `AWSServiceRoleForAmazonGuardDuty` to your IAM entities. This AWS managed policy is attached to a service-linked role that allows GuardDuty to perform actions on your behalf. For more information, see ??? (p.).

GuardDuty updates to AWS managed policies

View details about updates to AWS managed policies for GuardDuty since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the GuardDuty Document history page.

Change	Description	Date
AWSServiceRoleForAmazonGuardDuty (p. 167) – Update to an existing policy	GuardDuty added new permissions to allow GuardDuty to use Amazon EC2 networking actions to improve findings. GuardDuty can now perform the following EC2 actions to gain information about how your EC2 instances are communicating. This information is used to improve finding accuracy. • `ec2:DescribeVpcEndpoints` • `ec2:DescribeSubnets` • `ec2:DescribeVpcPeeringConnections` • `ec2:DescribeTransitGatewayAttachments`	Aug 3, 2021
GuardDuty started tracking changes	GuardDuty started tracking changes for its AWS managed policies.	Aug 3, 2021

Compliance validation for Amazon GuardDuty

Third-party auditors assess the security and compliance of GuardDuty as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using GuardDuty is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and compliance quick start guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA security and compliance whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating resources with rules in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices. GuardDuty integrates with Security Hub to consolidate findings.

Resilience in Amazon GuardDuty

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

Infrastructure security in Amazon GuardDuty

As a managed service, GuardDuty is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

You use AWS published API calls to access GuardDuty through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

AWS Service Integrations with GuardDuty

GuardDuty can be integrated with other AWS security services. These services can ingest data from GuardDuty to allow you to view findings in new ways. Review the following integration options to learn more about how that service is set up to work with GuardDuty.

Integrating GuardDuty with AWS Security Hub

AWS Security Hub collects security data from across your AWS accounts, services, and supported third party partner products to assess the security state of your environment according to industry standards and best practices. In addition to evaluating your security posture, Security Hub creates a central location for findings across all of your integrated AWS services, and AWS Partner products. Enabling Security Hub with GuardDuty will automatically allow GuardDuty findings data to be ingested by Security Hub.

For more information about using Security Hub with GuardDuty see Integration with AWS Security Hub (p. 174).

Integrating GuardDuty with Amazon Detective

Amazon Detective uses log data from across your AWS accounts to create data visualizations for your resources and IP addresses interacting with your environment. Detective's visualizations help you quickly and easily investigate security issues. You can pivot from GuardDuty finding details to information in the Detective console once both services are enabled.

For more information about using Detective with GuardDuty see Integration with Amazon Detective (p. 181).

Integration with AWS Security Hub

AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you to check your environment against security industry standards and best practices. Security Hub collects security data from across AWS accounts, services, and supported third-party partner products and helps you to analyze your security trends and identify the highest priority security issues.

The Amazon GuardDuty integration with Security Hub enables you to send findings from GuardDuty to Security Hub. Security Hub can then include those findings in its analysis of your security posture.

Contents

How Amazon GuardDuty sends findings to AWS Security Hub

In AWS Security Hub, security issues are tracked as findings. Some findings come from issues that are detected by other AWS services or by third-party partners. Security Hub also has a set of rules that it uses to detect security issues and generate findings.

Security Hub provides tools to manage findings from across all of these sources. You can view and filter lists of findings and view details for a finding. See Viewing findings in the *AWS Security Hub User Guide*. You can also track the status of an investigation into a finding. See Taking action on findings in the *AWS Security Hub User Guide*.

All findings in Security Hub use a standard JSON format called the AWS Security Finding Format (ASFF). The ASFF includes details about the source of the issue, the affected resources, and the current status of the finding. See AWS Security Finding Format (ASFF) in the *AWS Security Hub User Guide*.

Amazon GuardDuty is one of the AWS services that sends findings to Security Hub

Types of findings that GuardDuty sends to Security Hub

Once the integration is enabled, GuardDuty sends all of the findings it generates to Security Hub. The findings are sent to Security Hub using the AWS Security Finding Format (ASFF). In ASFF, the `Types` field provides the finding type.

Latency for sending findings

When GuardDuty creates a new finding, it is usually sent to Security Hub within five minutes.

Retrying when Security Hub is not available

If Security Hub is not available, GuardDuty retries sending the findings until they are received.

Updating existing findings in Security Hub

After it sends a finding to Security Hub, GuardDuty sends updates to reflect additional observations of the finding activity to Security Hub. The rate at which aggregated findings are updated is based on the Export update frequency (p. 107) specified.

Archiving or unarchiving a GuardDuty finding will not update the finding in Security Hub. This means that manually unarchived findings that become active in GuardDuty will not be sent to Security Hub

Viewing GuardDuty findings in AWS Security Hub

To view your GuardDuty Findings in Security Hub select **See Findings** under **Amazon GuardDuty** from the summary page. Alternatively you can select **Findings** from the navigation panel and filter the findings to display only GuardDuty findings by selecting the **Product name:** field with a value of `GuardDuty`.

Interpreting GuardDuty finding names in AWS Security Hub

GuardDuty sends the findings to Security Hub using the AWS Security Finding Format (ASFF). In ASFF, the `Types` field provides the finding type. ASFF types use a different naming scheme than GuardDuty

types. The table below details all the GuardDuty finding types with their ASFF counterpart as they appear in Security Hub.

Note
For some GuardDuty finding types Security Hub assigns different ASFF finding names depending on whether the finding detail's **Resource Role** was **ACTOR** or **TARGET**. For more information see Finding details (p. 13).

GuardDuty finding type	ASFF finding type
Backdoor:EC2/C&CActivity.B	TTPs/Command and Control/Backdoor:EC2-C&CActivity.B
Backdoor:EC2/C&CActivity.B!DNS	TTPs/Command and Control/Backdoor:EC2-C&CActivity.B!DNS
Backdoor:EC2/DenialOfService.Dns	TTPs/Command and Control/Backdoor:EC2-DenialOfService.Dns
Backdoor:EC2/DenialOfService.Tcp	TTPs/Command and Control/Backdoor:EC2-DenialOfService.Tcp
Backdoor:EC2/DenialOfService.Udp	TTPs/Command and Control/Backdoor:EC2-DenialOfService.Udp
Backdoor:EC2/DenialOfService.UdpOnTcpPorts	TTPs/Command and Control/Backdoor:EC2-DenialOfService.UdpOnTcpPorts
Backdoor:EC2/DenialOfService.UnusualProtocol	TTPs/Command and Control/Backdoor:EC2-DenialOfService.UnusualProtocol
Backdoor:EC2/Spambot	TTPs/Command and Control/Backdoor:EC2-Spambot
Behavior:EC2/NetworkPortUnusual	Unusual Behaviors/VM/Behavior:EC2-NetworkPortUnusual
Behavior:EC2/TrafficVolumeUnusual	Unusual Behaviors/VM/Behavior:EC2-TrafficVolumeUnusual
CryptoCurrency:EC2/BitcoinTool.B	TTPs/Command and Control/CryptoCurrency:EC2-BitcoinTool.B
CryptoCurrency:EC2/BitcoinTool.B!DNS	TTPs/Command and Control/CryptoCurrency:EC2-BitcoinTool.B!DNS
Discovery:S3/BucketEnumeration.Unusual	TTPs/Discovery:S3-BucketEnumeration.Unusual
Discovery:S3/MaliciousIPCaller.Custom	TTPs/Discovery:S3-MaliciousIPCaller.Custom
Discovery:S3/TorIPCaller	TTPs/Discovery:S3-TorIPCaller
Discovery:S3-MaliciousIPCaller	TTPs/Discovery:S3-MaliciousIPCaller
Exfiltration:S3/ObjectRead.Unusual	TTPs/Exfiltration:S3-ObjectRead.Unusual
Exfiltration:S3-MaliciousIPCaller	TTPs/Exfiltration:S3-MaliciousIPCaller
Impact:EC2/PortSweep	TTPs/Impact/Impact:EC2-PortSweep
Impact:EC2/WinRMBruteForce	TTPs/Impact/Impact:EC2-WinRMBruteForce

GuardDuty finding type	ASFF finding type
Impact:S3/ObjectDelete.Unusual	TTPs/Impact:S3-ObjectDelete.Unusual
Impact:S3/PermissionsModification.Unusual	TTPs/Impact:S3-PermissionsModification.Unusual
Impact:S3-MaliciousIPCaller	TTPs/Impact:S3-MaliciousIPCaller
PenTest:IAMUser/KaliLinux	TTPs/PenTest:IAMUser/KaliLinux
PenTest:IAMUser/ParrotLinux	TTPs/PenTest:IAMUser/ParrotLinux
PenTest:IAMUser/PentooLinux	TTPs/PenTest:IAMUser/PentooLinux
PenTest:S3/KaliLinux	TTPs/PenTest:S3-KaliLinux
PenTest:S3/ParrotLinux	TTPs/PenTest:S3-ParrotLinux
PenTest:S3/PentooLinux	TTPs/PenTest:S3-PentooLinux
Persistence:IAMUser/NetworkPermissions	TTPs/Persistence/Persistence:IAMUser-NetworkPermissions
Persistence:IAMUser/ResourcePermissions	TTPs/Persistence/Persistence:IAMUser-ResourcePermissions
Persistence:IAMUser/UserPermissions	TTPs/Persistence/Persistence:IAMUser-UserPermissions
Policy:IAMUser/RootCredentialUsage	TTPs/Policy:IAMUser-RootCredentialUsage
Policy:S3/AccountBlockPublicAccessDisabled	TTPs/Policy:S3-AccountBlockPublicAccessDisabled
Policy:S3/BucketAnonymousAccessGranted	TTPs/Policy:S3-BucketAnonymousAccessGranted
Policy:S3/BucketBlockPublicAccessDisabled	Effects/Data Exposure/Policy:S3-BucketBlockPublicAccessDisabled
Policy:S3/BucketPublicAccessGranted	TTPs/Policy:S3-BucketPublicAccessGranted
PrivilegeEscalation:IAMUser/AdministrativePermissions	TTPs/Privilege Escalation/PrivilegeEscalation:IAMUser-AdministrativePermissions
Recon:EC2/PortProbeEMRUnprotectedPort	TTPs/Discovery/Recon:EC2-PortProbeEMRUnprotectedPort
Recon:EC2/PortProbeUnprotectedPort	TTPs/Discovery/Recon:EC2-PortProbeUnprotectedPort
Recon:EC2/Portscan	TTPs/Discovery/Recon:EC2-Portscan
Recon:IAMUser/MaliciousIPCaller	TTPs/Discovery/Recon:IAMUser-MaliciousIPCaller
Recon:IAMUser/MaliciousIPCaller.Custom	TTPs/Discovery/Recon:IAMUser-MaliciousIPCaller.Custom
Recon:IAMUser/NetworkPermissions	TTPs/Discovery/Recon:IAMUser-NetworkPermissions
Recon:IAMUser/ResourcePermissions	TTPs/Discovery/Recon:IAMUser-ResourcePermissions

GuardDuty finding type	ASFF finding type
Recon:IAMUser/TorIPCaller	TTPs/Discovery/Recon:IAMUser-TorIPCaller
Recon:IAMUser/UserPermissions	TTPs/Discovery/Recon:IAMUser-UserPermissions
ResourceConsumption:IAMUser/ComputeResources	Unusual Behaviors/User/ResourceConsumption:IAMUser-ComputeResources
Stealth:IAMUser/CloudTrailLoggingDisabled	TTPs/Defense Evasion/Stealth:IAMUser-CloudTrailLoggingDisabled
Stealth:IAMUser/LoggingConfigurationModified	TTPs/Defense Evasion/Stealth:IAMUser-LoggingConfigurationModified
Stealth:IAMUser/PasswordPolicyChange	TTPs/Defense Evasion/Stealth:IAMUser-PasswordPolicyChange
Stealth:S3/ServerAccessLoggingDisabled	TTPs/Defense Evasion/Stealth:S3-ServerAccessLoggingDisabled
Trojan:EC2/BlackholeTraffic	TTPs/Command and Control/Trojan:EC2-BlackholeTraffic
Trojan:EC2/BlackholeTraffic!DNS	TTPs/Command and Control/Trojan:EC2-BlackholeTraffic!DNS
Trojan:EC2/DGADomainRequest.B	TTPs/Command and Control/Trojan:EC2-DGADomainRequest.B
Trojan:EC2/DGADomainRequest.C!DNS	TTPs/Command and Control/Trojan:EC2-DGADomainRequest.C!DNS
Trojan:EC2/DNSDataExfiltration	TTPs/Command and Control/Trojan:EC2-DNSDataExfiltration!DNS
Trojan:EC2/DriveBySourceTraffic!DNS	TTPs/Initial Access/Trojan:EC2-DriveBySourceTraffic!DNS
Trojan:EC2/DropPoint	Effects/Data Exfiltration/Trojan:EC2-DropPoint
Trojan:EC2/DropPoint!DNS	Effects/Data Exfiltration/Trojan:EC2-DropPoint!DNS
Trojan:EC2/PhishingDomainRequest!DNS	TTPs/Command and Control/Trojan:EC2-PhishingDomainRequest!DNS
UnauthorizedAccess:EC2/MaliciousIPCaller.Custom	TTPs/Command and Control/UnauthorizedAccess:EC2-MaliciousIPCaller.Custom
UnauthorizedAccess:EC2/MetadataDNSRebind	TTPs/UnauthorizedAccess:EC2-MetadataDNSRebind
UnauthorizedAccess:EC2/RDPBruteForce	TTPs/Initial Access/UnauthorizedAccess:EC2-RDPBruteForce
UnauthorizedAccess:EC2/SSHBruteForce	TTPs/Initial Access/UnauthorizedAccess:EC2-SSHBruteForce

GuardDuty finding type	ASFF finding type
UnauthorizedAccess:EC2/TorClient	Effects/Resource Consumption/ UnauthorizedAccess:EC2-TorClient
UnauthorizedAccess:EC2/TorRelay	Effects/Resource Consumption/ UnauthorizedAccess:EC2-TorRelay
UnauthorizedAccess:IAMUser/ConsoleLogin	Unusual Behaviors/User/ UnauthorizedAccess:IAMUser-ConsoleLogin
UnauthorizedAccess:IAMUser/ ConsoleLoginSuccess.B	TTPs/UnauthorizedAccess:IAMUser-ConsoleLoginSuccess.B
UnauthorizedAccess:IAMUser/ InstanceCredentialExfiltration	Effects/Data Exfiltration/ UnauthorizedAccess:IAMUser-InstanceCredentialExfiltration
UnauthorizedAccess:IAMUser/MaliciousIPCaller	TTPs/UnauthorizedAccess:IAMUser-MaliciousIPCaller
UnauthorizedAccess:IAMUser/ MaliciousIPCaller.Custom	TTPs/UnauthorizedAccess:IAMUser-MaliciousIPCaller.Custom
UnauthorizedAccess:IAMUser/TorIPCaller	TTPs/Command and Control/ UnauthorizedAccess:IAMUser-TorIPCaller
UnauthorizedAccess:S3/MaliciousIPCaller.Custom	TTPs/UnauthorizedAccess:S3-MaliciousIPCaller.Custom
UnauthorizedAccess:S3/TorIPCaller	TTPs/UnauthorizedAccess:S3-TorIPCaller

Typical finding from GuardDuty

GuardDuty sends findings to Security Hub using the AWS Security Finding Format (ASFF).

Here is an example of a typical finding from GuardDuty.


```
  {
  "SchemaVersion": "2018-10-08",
  "Id": "arn:aws::guardduty:us-east-1:193043430472:detector/
d4b040365221be2b54a6264dc9a4bc64/finding/46ba0ac2845071e23ccdeb2ae03bfdea",
  "ProductArn": "arn:aws::securityhub:us-east-1::product/aws/guardduty",
  "GeneratorId": "arn:aws::guardduty:us-east-1:193043430472:detector/
d4b040365221be2b54a6264dc9a4bc64",
  "AwsAccountId": "193043430472",
  "Types": [
    "TTPs/Initial Access/UnauthorizedAccess:EC2-SSHBruteForce"
  ],
  "FirstObservedAt": "2020-08-22T09:15:57Z",
  "LastObservedAt": "2020-09-30T11:56:49Z",
  "CreatedAt": "2020-08-22T09:34:34.146Z",
  "UpdatedAt": "2020-09-30T12:14:00.206Z",
  "Severity": {
    "Product": 2,
    "Label": "MEDIUM",
    "Normalized": 40
  },
  "Title": "199.241.229.197 is performing SSH brute force attacks against
 i-0c10c2c7863d1a356.",
```

```
  "Description": "199.241.229.197 is performing SSH brute force attacks against
 i-0c10c2c7863d1a356. Brute force attacks are used to gain unauthorized access to your
 instance by guessing the SSH password.",
  "SourceUrl": "https://us-east-1.console.aws.amazon.com/guardduty/home?region=us-east-1#/
findings?macros=current&fId=46ba0ac2845071e23ccdeb2ae03bfdea",
  "ProductFields": {
    "aws/guardduty/service/action/networkConnectionAction/remotePortDetails/portName":
 "Unknown",
    "aws/guardduty/service/archived": "false",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/organization/
asnOrg": "CENTURYLINK-US-LEGACY-QWEST",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/geoLocation/lat":
 "42.5122",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/ipAddressV4":
 "199.241.229.197",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/geoLocation/lon":
 "-90.7384",
    "aws/guardduty/service/action/networkConnectionAction/blocked": "false",
    "aws/guardduty/service/action/networkConnectionAction/remotePortDetails/port": "46717",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/country/
countryName": "United States",
    "aws/guardduty/service/serviceName": "guardduty",
    "aws/guardduty/service/evidence": "",
    "aws/guardduty/service/action/networkConnectionAction/localIpDetails/ipAddressV4":
 "172.31.43.6",
    "aws/guardduty/service/detectorId": "d4b040365221be2b54a6264dc9a4bc64",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/organization/
org": "CenturyLink",
    "aws/guardduty/service/action/networkConnectionAction/connectionDirection": "INBOUND",
    "aws/guardduty/service/eventFirstSeen": "2020-08-22T09:15:57Z",
    "aws/guardduty/service/eventLastSeen": "2020-09-30T11:56:49Z",
    "aws/guardduty/service/action/networkConnectionAction/localPortDetails/portName":
 "SSH",
    "aws/guardduty/service/action/actionType": "NETWORK_CONNECTION",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/city/cityName":
 "Dubuque",
    "aws/guardduty/service/additionalInfo": "",
    "aws/guardduty/service/resourceRole": "TARGET",
    "aws/guardduty/service/action/networkConnectionAction/localPortDetails/port": "22",
    "aws/guardduty/service/action/networkConnectionAction/protocol": "TCP",
    "aws/guardduty/service/count": "74",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/organization/
asn": "209",
    "aws/guardduty/service/action/networkConnectionAction/remoteIpDetails/organization/
isp": "CenturyLink",
    "aws/securityhub/FindingId": "arn:aws::securityhub:us-east-1::product/aws/guardduty/
arn:aws::guardduty:us-east-1:193043430472:detector/d4b040365221be2b54a6264dc9a4bc64/
finding/46ba0ac2845071e23ccdeb2ae03bfdea",
    "aws/securityhub/ProductName": "GuardDuty",
    "aws/securityhub/CompanyName": "Amazon"
  },
  "Resources": [
    {
      "Type": "AwsEc2Instance",
      "Id": "arn:aws::ec2:us-east-1:193043430472:instance/i-0c10c2c7863d1a356",
      "Partition": "aws",
      "Region": "us-east-1",
      "Tags": {
        "Name": "kubectl"
      },
      "Details": {
        "AwsEc2Instance": {
          "Type": "t2.micro",
          "ImageId": "ami-02354e95b39ca8dec",
          "IpV4Addresses": [
            "18.234.130.16",
```

```
            "172.31.43.6"
          ],
          "VpcId": "vpc-a0c2d7c7",
          "SubnetId": "subnet-4975b475",
          "LaunchedAt": "2020-08-03T23:21:57Z"
        }
      }
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ACTIVE"
}
```


Enabling and configuring the integration

To use the integration with AWS Security Hub, you must enable Security Hub. For information on how to enable Security Hub, see Setting up Security Hub in the *AWS Security Hub User Guide*.

When you enable both GuardDuty and Security Hub, the integration is enabled automatically. GuardDuty immediately begins to send findings to Security Hub.

Stopping the publication of findings to Security Hub

To stop sending findings to Security Hub, you can use either the Security Hub console or the API.

See Disabling and enabling the flow of findings from an integration (console) or Disabling the flow of findings from an integration (Security Hub API, AWS CLI) in the *AWS Security Hub User Guide*.

Integration with Amazon Detective

Amazon Detective helps you quickly analyze and investigate security events across one or more AWS accounts by generating data visualizations that represent the ways your resources behave and interact over time. Detective creates visualizations of GuardDuty findings for supported finding types. For a list of the supported finding types see Supported finding types.

Even if a GuardDuty finding type is not supported in Detective, you can still use Detective's visualizations to investigate different entities that are involved with the finding. An entity can be an AWS Account, an AWS resource within an account, or an external IP Address that has interacted with your resources. The GuardDuty console supports pivoting to Amazon Detective from the following entities, depending on finding type: AWS account, IAM user, IAM role or role session, user agent, federated user, Amazon EC2 instance, or IP address.

Contents

Enabling the integration

To use Amazon Detective with GuardDuty you must first enable Amazon Detective. For information on how to enable Detective, see Setting up Amazon Detective in the *Amazon Detective Administration Guide*.

When you enable both GuardDuty and Detective, the integration is enabled automatically. Once enabled, Detective will immediately ingest your GuardDuty findings data.

Note
GuardDuty sends findings to Detective based on the GuardDuty findings export frequency. By default, the export frequency for updates to existing findings is 6 hours. To ensure Detective receives the most recent updates to your findings it is recommended that you change the export freqeuncy to 15 minutes in each region in which you use Detective with GuardDuty. For more information see Setting the frequency for exporting updated active findings (p. 107).

Pivoting to Amazon Detective from a GuardDuty finding

1. Open GuardDuty at https://console.aws.amazon.com/guardduty
2. Select a single finding from your findings table.
3. Select **Investigate with Detective** from the finding details pane.
4. Choose an aspect of the finding to investigate with Amazon Detective. This opens the Detective console for that finding or entity.

If the pivot does not behave as expected, refer to Troubleshooting the pivot in the *Amazon Detective User Guide*.

Note
If you archive a GuardDuty finding in the Detective console that finding will be archived in the GuardDuty console as well.

Using the integration with a GuardDuty multi-account environment

If you are managing a multi-account environment in GuardDuty, you must add your member accounts to Amazon Detective in order to see Detective data visualizations for findings and entities in those accounts.

It is recommended that you use the same GuardDuty Administrator account as the administrator account for Detective. For more information on adding member accounts in Detective see Inviting member accounts.

Note
Detective is a regional service, meaning you must enable Detective and add your member accounts in each region in which you want to use the integration.

Suspending or disabling GuardDuty

You can use the GuardDuty console to suspend or disable GuardDuty. You are not charged for using GuardDuty when the service is suspended.

- All optional data sources must be disabled from all detectors in all regions before you can disable or suspend GuardDuty.
- All member accounts must be disassociated or deleted before you can disable or suspend GuardDuty.
- If you suspend GuardDuty, it no longer monitors the security of your AWS environment or generates new findings. Your existing findings remain intact and are not affected by the GuardDuty suspension. You can choose to re-enable GuardDuty later.
- If you disable GuardDuty, your existing findings and the GuardDuty configuration are lost and can't be recovered. If you want to save your existing findings, you must export them before you disable GuardDuty.

To suspend or disable GuardDuty

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, Choose **Settings**.
3. Within the **Suspend GuardDuty** section, choose **Suspend GuardDuty** or **Disable GuardDuty**, then confirm your action in the next panel.

Subscribing to GuardDuty announcements SNS topic

This section provides information on subscribing to the GuardDuty Announcements SNS topic to receive notifications about newly released finding types, updates to the existing finding types, and other functionality changes. Notifications are available in all formats that Amazon SNS supports.

The GuardDuty SNS topic provides information on updates to the GuardDuty service across AWS to any subscribed account. To receiving notifications about findings within your account, see Creating custom responses to GuardDuty findings with Amazon CloudWatch Events (p. 107)

Note
Your user account must have sns::subscribe IAM permissions to subscribe to an SNS topic.

You can subscribe an Amazon SQS queue to this notification topic, but you must use a topic ARN that is in the same Region. For more information, see Tutorial: Subscribing an Amazon SQS queue to an Amazon SNS topic in the Amazon Simple Queue Service Developer Guide.

You can also use an AWS Lambda function to trigger events when notifications are received. For more information, see Invoking Lambda functions using Amazon SNS notifications in the Amazon Simple Notification Service Developer Guide.

The Amazon SNS topic ARNs for each Region are shown below.

AWS Region	Amazon SNS topic ARN
us-east-1	arn:aws:sns:us-east-1:242987662583:GuardDutyAnnouncements
us-east-2	arn:aws:sns:us-east-2:118283430703:GuardDutyAnnouncements
us-west-1	arn:aws:sns:us-west-1:144182107116:GuardDutyAnnouncements
us-west-2	arn:aws:sns:us-west-2:934957504740:GuardDutyAnnouncements
ca-central-1	arn:aws:sns:ca-central-1:107430051933:GuardDutyAnnouncements
eu-north-1	arn:aws:sns:eu-north-1:973841112453:GuardDutyAnnouncements
eu-west-1	arn:aws:sns:eu-west-1:965013871422:GuardDutyAnnouncements
eu-west-2	arn:aws:sns:eu-west-2:506403581195:GuardDutyAnnouncements
eu-west-3	arn:aws:sns:eu-west-3:436163563069:GuardDutyAnnouncements
eu-central-1	arn:aws:sns:eu-central-1:378365507264:GuardDutyAnnouncements

AWS Region	Amazon SNS topic ARN
ap-east-1	arn:aws:sns:ap-east-1:646602203151:GuardDutyAnnouncements
ap-northeast-1	arn:aws:sns:ap-northeast-1:741172661024:GuardDutyAnnouncements
ap-northeast-2	arn:aws:sns:ap-northeast-2:464168911255:GuardDutyAnnouncements
ap-southeast-1	arn:aws:sns:ap-southeast-1:476419727788:GuardDutyAnnouncements
ap-southeast-2	arn:aws:sns:ap-southeast-2:457615622431:GuardDutyAnnouncements
ap-south-1	arn:aws:sns:ap-south-1:926826061926:GuardDutyAnnouncements
sa-east-1	arn:aws:sns:sa-east-1:955633302743:GuardDutyAnnouncements
us-gov-west-1	arn:aws-us-gov:sns:us-gov-west-1:430639793359:GuardDutyAnnouncements
cn-north-1	arn:aws-cn:sns:cn-north-1:002991280229:GuardDutyAnnouncements
cn-northwest-1	arn:aws-cn:sns:cn-northwest-1:003033775354:GuardDutyAnnouncements

To subscribe to the GuardDuty update notification email in the AWS Management Console

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the Region list, choose the same Region as the topic ARN to which to subscribe. This example uses the `us-west-2` Region.
3. In the left navigation pane, choose **Subscriptions**, **Create subscription**.
4. In the **Create Subscription** dialog box, for **Topic ARN**, paste the topic ARN: `arn:aws:sns:us-west-2:934957504740:GuardDutyAnnouncements`.
5. For **Protocol**, choose **Email**. For **Endpoint**, type an email address that you can use to receive the notification.
6. Choose **Create subscription**.
7. In your email application, open the message from AWS Notifications and open the link to confirm your subscription.

   Your web browser displays a confirmation response from Amazon SNS.

To subscribe to the GuardDuty update notification email with the AWS CLI

1. Run the following command with the AWS CLI:

```
 AWS  sns --region us-west-2 subscribe --topic-arn arn:aws:sns:us-
west-2:934957504740:GuardDutyAnnouncements --protocol email --notification-
endpoint your_email@your_domain.com
```

2. In your email application, open the message from AWS Notifications and open the link to confirm your subscription.

   Your web browser displays a confirmation response from Amazon SNS.

Amazon SNS message format

An example GuardDuty update notification message about new findings is shown below:


```
{
    "Type" : "Notification",
    "MessageId" : "9101dc6b-726f-4df0-8646-ec2f94e674bc",
    "TopicArn" : "arn:aws:sns:us-west-2:934957504740:GuardDutyAnnouncements",
    "Message" : "{\"version\":\"1\",\"type\":\"NEW_FINDINGS\",\"findingDetails\":[{\"link
\":\"https://docs.aws.amazon.com//guardduty/latest/ug/guardduty_unauthorized.html\",
\"findingType\":\"UnauthorizedAccess:EC2/TorClient\",\"findingDescription\":\"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 is acting as a client on a Tor network. A common use for
 a Tor client is to circumvent network monitoring and filter for access to unauthorized
 or illicit content. Tor clients can also generate nefarious Internet traffic, including
 attacking SSH servers. This activity can indicate that your EC2 instance is compromised.
\"}]}",
    "Timestamp" : "2018-03-09T00:25:43.483Z",
    "SignatureVersion" : "1",
    "Signature" : "XWox8GDGLRiCgDOXlo/
fG9Lu/88P8S0FL6M6oQYOmUFzkucuhoblsdea3BjqdCHcWR7qdhMPQnLpN7y9iBrWVUqdAGJrukAI8athvAS+4AQD/
V/QjrhsEnlj+GaiW
+ozAu006X6GopOzFGnCtPMROjCMrMonjz7Hpv/8KRuMZR3pyQYm5d4wWB7xBPYhUMuLoZ1V8YFs55FMtgQV/
YLhSYuEu0BP1GMtLQauxDkscOtPP/vjhGQLFx1Q9LTadcQiRHtNIBxWL87PSI+BVvkin6AL7PhksvdQ7FAgHfXsit
+6p8GyOvKCqaeBG7HZhR1AbpyVka7JSNRO/6ssyrlj1g==",
    "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-433026a4050d206028891664da859041.pem",
    "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:934957504740:GuardDutyAnnouncements:9225ed2b-7228-4665-8a01-c8a5db6859f4"
}
```


The parsed Message value (with escaped quotes removed) is shown below:


```
{
    "version": "1",
    "type": "NEW_FINDINGS",
    "findingDetails": [{
        "link": "https://docs.aws.amazon.com//guardduty/latest/ug/
guardduty_unauthorized.html",
        "findingType": "UnauthorizedAccess:EC2/TorClient",
        "findingDescription": "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 is acting as a
 client on a Tor network. A common use for a Tor client is to circumvent network monitoring
 and filter for access to unauthorized or illicit content. Tor clients can also generate
 nefarious Internet traffic, including attacking SSH servers. This activity can indicate
 that your EC2 instance is compromised."
    }]
}
```


An example GuardDuty update notification message about GuardDuty functionality updates is shown below:

```
{
    "Type" : "Notification",
    "MessageId" : "9101dc6b-726f-4df0-8646-ec2f94e674bc",
    "TopicArn" : "arn:aws:sns:us-west-2:934957504740:GuardDutyAnnouncements",
    "Message" : "{\"version\":\"1\",\"type\":\"NEW_FEATURES\",\"featureDetails\":
[{\"featureDescription\":\"Customers with high-volumes of global CloudTrail events
 should see a net positive impact on their GuardDuty costs.\",\"featureLink\":\"https://
docs.aws.amazon.com//guardduty/latest/ug/guardduty_data-sources.html#guardduty_cloudtrail
\"}]}",
    "Timestamp" : "2018-03-09T00:25:43.483Z",
    "SignatureVersion" : "1",
    "Signature" : "XWox8GDGLRiCgDOXlo/
fG9Lu/88P8S0FL6M6oQYOmUFzkucuhoblsdea3BjqdCHcWR7qdhMPQnLpN7y9iBrWVUqdAGJrukAI8athvAS+4AQD/
V/QjrhsEnlj+GaiW
+ozAu006X6GopOzFGnCtPMROjCMrMonjz7Hpv/8KRuMZR3pyQYm5d4wWB7xBPYhUMuLoZ1V8YFs55FMtgQV/
YLhSYuEu0BP1GMtLQauxDkscOtPP/vjhGQLFx1Q9LTadcQiRHtNIBxWL87PSI+BVvkin6AL7PhksvdQ7FAgHfXsit
+6p8GyOvKCqaeBG7HZhR1AbpyVka7JSNRO/6ssyrlj1g==",
    "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-433026a4050d206028891664da859041.pem",
    "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:934957504740:GuardDutyAnnouncements:9225ed2b-7228-4665-8a01-c8a5db6859f4"
}
```


The parsed Message value (with escaped quotes removed) is shown below:


```
{
    "version": "1",
    "type": "NEW_FEATURES",
    "featureDetails": [{
        "featureDescription": "Customers with high-volumes of global CloudTrail events
 should see a net positive impact on their GuardDuty costs.",
        "featureLink": "https://docs.aws.amazon.com//guardduty/latest/ug/guardduty_data-
sources.html#guardduty_cloudtrail"
    }]
}
```


An example GuardDuty update notification message about updated findings is shown below:


```
{
    "Type": "Notification",
    "MessageId": "9101dc6b-726f-4df0-8646-ec2f94e674bc",
    "TopicArn": "arn:aws:sns:us-west-2:934957504740:GuardDutyAnnouncements",
    "Message": "{\"version\":\"1\",\"type\":\"UPDATED_FINDINGS\",\"findingDetails\":
[{\"link\":\"https://docs.aws.amazon.com//guardduty/latest/ug/guardduty_unauthorized.html
\",\"findingType\":\"UnauthorizedAccess:EC2/TorClient\",\"description\":\"Increased
 severity value from 5 to 8.\"}]}",
    "Timestamp": "2018-03-09T00:25:43.483Z",
    "SignatureVersion": "1",
    "Signature": "XWox8GDGLRiCgDOXlo/
fG9Lu/88P8S0FL6M6oQYOmUFzkucuhoblsdea3BjqdCHcWR7qdhMPQnLpN7y9iBrWVUqdAGJrukAI8athvAS+4AQD/
V/QjrhsEnlj+GaiW
+ozAu006X6GopOzFGnCtPMROjCMrMonjz7Hpv/8KRuMZR3pyQYm5d4wWB7xBPYhUMuLoZ1V8YFs55FMtgQV/
YLhSYuEu0BP1GMtLQauxDkscOtPP/vjhGQLFx1Q9LTadcQiRHtNIBxWL87PSI+BVvkin6AL7PhksvdQ7FAgHfXsit
+6p8GyOvKCqaeBG7HZhR1AbpyVka7JSNRO/6ssyrlj1g==",
    "SigningCertURL": "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-433026a4050d206028891664da859041.pem",
    "UnsubscribeURL": "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:934957504740:GuardDutyAnnouncements:9225ed2b-7228-4665-8a01-c8a5db6859f4"
}
```

The parsed Message value (with escaped quotes removed) is shown below:

```
{
    "version": "1",
    "type": "UPDATED_FINDINGS",
    "findingDetails": [{
        "link": "https://docs.aws.amazon.com//guardduty/latest/ug/
guardduty_unauthorized.html",
        "findingType": "UnauthorizedAccess:EC2/TorClient",
        "description": "Increased severity value from 5 to 8."
    }]
}
```

Quotas for Amazon GuardDuty

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for GuardDuty, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **Amazon GuardDuty**.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*.

Your AWS account has the following quotas for Amazon GuardDuty per Region.

Resource	Default	Comments
Detectors	1	The maximum number of detector resources that you can create per AWS account per Region. You cannot request a quota increase.
Filters	100	The maximum number of saved filters per AWS account per Region.
Finding retention period	90 days	The maximum number of days a finding is retained. You cannot request a quota increase.
IP addresses and CIDR ranges per Trusted IP List	2,000	The maximum number of IP addresses and CIDR ranges that you can include in a single Trusted IP List. You cannot request a quota increase.
IP addresses and CIDR ranges per Threat List	250,000	The maximum number of IP address and CIDR ranges that you can include in a Threat List. You cannot request a quota increase.
Maximum file size	35 MB	The maximum file size for the file used to upload a list of IP

Resource	Default	Comments
		addresses or CIDR ranges to include in a Trusted IP List or a Threat List. You cannot request a quota increase.
Member accounts	5000	The maximum number of member accounts associated with a administrator account. You can have one administrator account per detector.
Threat intel sets	6	The maximum number of Threat intel sets that you can add per AWS account per Region.
Trusted IP sets	1	The maximum number of trusted IP sets that can be uploaded and activated per AWS account per Region. You cannot request a quota increase.

Regions and endpoints

To view the Regions where Amazon GuardDuty is available, see Amazon GuardDuty endpoints in the *Amazon Web Services General Reference*.

We highly recommend that you enable GuardDuty in all supported AWS Regions. This enables GuardDuty to generate findings about unauthorized or unusual activity even in Regions that you are not actively using. This also allows GuardDuty to monitor AWS CloudTrail events for global AWS services such as AWS Identity and Access Management (IAM). If GuardDuty is not enabled in all supported Regions, its ability to detect activity that involves global services is reduced.

If you are using GuardDuty in AWS GovCloud (US), there are differences. For more information, see Amazon GuardDuty in the *AWS GovCloud (US) User Guide*.

Document history for Amazon GuardDuty

update-history-change	update-history-description	update-history-date
Added Kubernetes protection content for GuardDuty	GuardDuty can now generate findings for your Amazon EKS resources through the monitoring of Kubernetes audit logs. To learn how to configure this feature see Kubernetes protection in Amazon GuardDuty. For a list of findings GuardDuty can generate for Amazon EKS resources, see Kubernetes findings. New remediation guidance has been added to support remediating these findings in the Kubernetes finding remediation guide.	January 25, 2022
Added 1 new finding	A new finding UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS has been added. This finding informs you when your instance credentials are accessed by an AWS account outside your AWS environment.	January 20, 2022
Updated the finding types to help identify issues related to log4j (p. 26)	Amazon GuardDuty has updated the following finding types to help identify and prioritize issues related to CVE-2021-44228 and CVE-2021-45046: Backdoor:EC2/C&CActivity.B; Backdoor:EC2/C&CActivity.B!DNS; Behavior:EC2/NetworkPortUnusual.	December 22, 2021
Finding Changes	UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration has been changed to UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS. This improved version of the finding learns the typical locations your credentials are used from to reduce findings from traffic routed through on premise networks. UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS	September 7, 2021

Update to GuardDuty SLR	The GuardDuty SLR has been updated with new actions to improve finding accuracy.	August 3, 2021
Added data source information for each finding type.	Finding descriptions now contain information on which data source GuardDuty uses to generate that finding.	May 10, 2021
Retired 13 finding types.	13 findings have been retired to be replaced with new AnomalousBehavoir findings. Persistence:IAMUser/NetworkPermissions, Persistence:IAMUser/ResourcePermissions, Persistence:IAMUser/UserPermissions, PrivilegeEscalation:IAMUser/AdministrativePermissions, Recon:IAMUser/NetworkPermissions, Recon:IAMUser/ResourcePermissions, Recon:IAMUser/UserPermissions, ResourceConsumption:IAMUser/ComputeResources, Stealth:IAMUser/LoggingConfigurationModified, Discovery:S3/BucketEnumeration.Unusual, Impact:S3/ObjectDelete.Unusual, Impact:S3/PermissionsModification.Unusual.	March 12, 2021
Added 8 new finding types for anomalous behavior.	Added 8 new IAMUser finding types based on anomalous behavior for IAM principals. CredentialAccess:IAMUser/AnomalousBehavior, DefenseEvasion:IAMUser/AnomalousBehavior, Discovery:IAMUser/AnomalousBehavior, Exfiltration:IAMUser/AnomalousBehavior, Impact:IAMUser/AnomalousBehavior, InitialAccess:IAMUser/AnomalousBehavior, Persistence:IAMUser/AnomalousBehavior, PrivilegeEscalation:IAMUser/AnomalousBehavior.	March 12, 2021

Added EC2 findings based on domain reputation.	Added 4 new Impact finding types based on domain reputation. Impact:EC2/AbusedDomainRequest.Reputation , Impact:EC2/BitcoinDomainRequest.Reputation , Impact:EC2/MaliciousDomainRequest.Reputation . Also added a new EC2 finding for C&CActivity. Impact:EC2/SuspiciousDomainRequest.Reputation	January 27, 2021
Added 4 new finding types.	Added 3 new S3 MaliciousIPCaller findings. Discovery:S3/MaliciousIPCaller , Exfiltration:S3/MaliciousIPCaller , Impact:S3/MaliciousIPCaller . Also added a new EC2 finding for C&CActivity. Backdoor:EC2/C&CActivity.B	December 21, 2020
Retired the UnauthorizedAccess:EC2/TorIPCaller finding type.	The UnauthorizedAccess:EC2/TorIPCaller finding type is now retired from GuardDuty. Learn more.	October 1, 2020
Added the Impact:EC2/WinRmBruteForce finding type.	Added a new Impact finding, Impact:EC2/WinRmBruteForce. Learn more.	September 17, 2020
Added the Impact:EC2/PortSweep finding type.	Added a new Impact finding, Impact:EC2/PortSweep. Learn more.	September 17, 2020
GuardDuty is now available in the Africa (Cape Town) and Europe (Milan) Regions.	Added Africa (Cape Town) and Europe (Milan) to the list of AWS Regions in which GuardDuty is available. Learn more	July 31, 2020
Added new usage details for monitoring GuardDuty costs.	You can now use new metrics to query GuardDuty usage cost data for your account and accounts you manage. A new overview of usage costs is available in the console at https://console.aws.amazon.com/guardduty/. More detailed information can be accessed through the API.	July 31, 2020

Added content covering S3 protection through S3 data event monitoring in GuardDuty.	GuardDuty S3 Protection is now available through the monitoring of S3 data plane events as a new data source. New accounts will have this feature enabled automatically. If you are already using GuardDuty you can enable the new data source for yourself or your member accounts.	July 31, 2020
Added 14 new S3 Findings.	14 new S3 finding types have been added for S3 control plane and data plane sources.	July 31, 2020
Added additional support for S3 findings and changed 2 existing finding types names.	GuardDuty findings now include more details for findings involving S3 buckets. Existing finding types that were related to S3 activity have been renamed: Policy:IAMUser/S3BlockPublicAccessDisabled has been changed to Policy:S3/BucketBlockPublicAccessDisabled. Stealth:IAMUser/S3ServerAccessLoggingDisabled has been changed to Stealth:S3/ServerAccessLoggingDisabled.	May 28, 2020
Added content for AWS Organizations integration.	GuardDuty now integrates with AWS Organizations delegated administrators to allow you to manage GuardDuty accounts within your organization. When you set a delegated administrator as your GuardDuty administrator you can automatically enable GuardDuty for any organization member to be managed by the delegated administrator account. You can also automatically enable GuardDuty in new AWS Organizations member accounts. Learn more.	April 20, 2020
Added content for the export findings feature.	Added content that describes the **Export Findings** feature of GuardDuty.	November 14, 2019
Added the UnauthorizedAccess:EC2/MetadataDNSRebind finding type.	Added a new Unauthorized finding, UnauthorizedAccess:EC2/MetadataDNSRebind. Learn more.	October 10, 2019

Added the Stealth:IAMUser/S3ServerAccessLoggingDisabled finding type.	Added a new Stealth finding, Stealth:IAMUser/S3ServerAccessLoggingDisabled. Learn more.	October 10, 2019
Added the Policy:IAMUser/S3BlockPublicAccessDisabled finding type.	Added a new Policy finding, Policy:IAMUser/S3BlockPublicAccessDisabled. Learn more.	October 10, 2019
Retired the Backdoor:EC2/XORDDOS finding type.	The Backdoor:EC2/XORDDOS finding type is now retired from GuardDuty.Learn more	June 12, 2019
Added the PrivilegeEscalation finding type.	The PrivilegeEscalation finding type detects when users attempt to assign escalated, more permissive privileges to their accounts. Learn more	May 14, 2019
GuardDuty is now available in the Europe (Stockholm) Region.	Added Europe (Stockholm) to the list of AWS Regions in which GuardDuty is available. Learn more	May 9, 2019
Added a new finding type, Recon:EC2/PortProbeEMRUnprotectedPort.	This finding informs you that an EMR-related sensitive port on an EC2 Instance is not blocked and is being actively probed. Learn more	May 8, 2019
Added 5 new finding types that detect if your EC2 instances are potentially being used for denial of service (DoS) attacks.	These findings inform you of EC2 instances in your environment that are behaving in a manner that may indicate they is being used to perform Denial of Service (DoS) attacks. Learn more	March 8, 2019
Added a new finding type: Policy:IAMUser/RootCredentialUsage	Policy:IAMUser/RootCredentialUsage finding type informs you that the root credentials of your AWS account are being used to make programmatic requests to AWS services. Learn more	January 24, 2019
UnauthorizedAccess:IAMUser/UnusualASNCaller finding type has been retired	The UnauthorizedAccess:IAMUser/UnusualASNCaller finding type has been retired. You will now be notified about activity invoked from unusual networks via other active GuardDuty finding types. The generated finding type will be based on the category of the API that was invoked from an unusual network. Learn more	December 21, 2018

Added two new finding types: PenTest:IAMUser/ParrotLinux and PenTest:IAMUser/ PentooLinux	PenTest:IAMUser/ParrotLinux finding type informs you that a computer running Parrot Security Linux is making API calls using credentials that belong to your AWS account. PenTest:IAMUser/PentooLinux finding type informs you that a machine running Pentoo Linux is making API calls using credentials that belong to your AWS account. Learn more	December 21, 2018
Added support for the Amazon GuardDuty announcements SNS topic	You can now subscribe to the GuardDuty Announcements SNS topic to receive notifications about newly released finding types, updates to the existing finding types, and other functionality changes. Notifications are available in all formats that Amazon SNS supports. Learn more	November 21, 2018
Added two new finding types: UnauthorizedAccess:EC2/ TorClient and UnauthorizedAccess:EC2/ TorRelay	UnauthorizedAccess:EC2/ TorClient finding type informs you that an EC2 instance in your AWS environment is making connections to a Tor Guard or an Authority node. UnauthorizedAccess:EC2/ TorRelay finding type 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. Learn more	November 16, 2018
Added a new finding type: CryptoCurrency:EC2/ BitcoinTool.B	This finding informs you that an EC2 instance in your AWS environment is querying a domain name that is associated with Bitcoin, or other cryptocurrency-related activity. Learn more	November 9, 2018
Added support for updating the frequency of notifications sent to CloudWatch events	You can now update the frequency of notifications sent to CloudWatch Events for the subsequent occurrences of existing findings. Possible values are 15 minutes, 1 hour, or the default 6 hours. Learn more	October 9, 2018
Added Region support	Added Region support for AWS GovCloud (US-West) Learn more	July 25, 2018

Added support for AWS CloudFormation StackSets in GuardDuty	You can use the Enable Amazon GuardDuty template to enable GuardDuty simultaneously in multiple accounts. Learn more	June 25, 2018
Added support for GuardDuty auto-archive rules	Customers can now build granular auto-archive rules for suppression of findings. For findings that match an auto-archive rule, GuardDuty automatically marks them as archived. This enables customers to further tune GuardDuty to keep only relevant findings in the current findings table. Learn more	May 4, 2018
GuardDuty is available in the Europe (Paris) Region	GuardDuty is now available in Europe (Paris), allowing you to extend continuous security monitoring and threat detection in this Region. Learn more	March 29, 2018
Creating GuardDuty administrator and member accounts through AWS CloudFormation is now supported.	For more information, see `AWS::GuardDuty::master` and `AWS::GuardDuty::member`.	March 6, 2018
Added nine new CloudTrail-based anomaly detections.	These new finding types are automatically enabled in GuardDuty in all supported Regions. Learn more	February 28, 2018
Added three new threat intelligence detections (finding types).	These new finding types are automatically enabled in GuardDuty in all supported Regions. Learn more	February 5, 2018
Limit increase for GuardDuty member accounts.	With this release, you can have up to 1000 GuardDuty member accounts added per AWS account (GuardDuty administrator account). Learn more	January 25, 2018
Changes in upload and further management of trusted IP lists and threat lists for GuardDuty administrator and member accounts.	With this release, Users from administrator GuardDuty accounts can upload and manage trusted IP lists and threat lists. Users from member GuardDuty accounts CANNOT upload and manage lists. Trusted IP lists and threat lists that are uploaded by the administrator account are imposed on GuardDuty functionality in its member accounts. Learn more	January 25, 2018

The following table describes important changes in each release of the *GuardDuty* User Guide.

Earlier updates

Change	Description	Date
Initial publication	Initial publication of the Amazon GuardDuty User Guide.	November 28, 2017