Amazon GuardDuty User Guide

Amazon GuardDuty

Amazon GuardDuty: Amazon GuardDuty User Guide

Table of Contents

What is Amazon GuardDuty?

Amazon GuardDuty is a threat detection service that continuously monitors your AWS environment for potential security risks. GuardDuty analyzes and processes Foundational data sources, such as AWS CloudTrail management events, AWS CloudTrail event logs, VPC flow logs (from Amazon EC2 instances), and DNS logs. GuardDuty also offers monitoring logs and events from other AWS services. These sources include Kubernetes audit logs, RDS login activity, S3 logs, EBS volumes, Runtime monitoring, and Lambda network activity logs. GuardDuty consolidates these log and event sources under the term - Features.

GuardDuty uses threat intelligence feeds, such as lists of malicious IP addresses and domains, and machine learning (ML) models to identify unexpected, potentially unauthorized, and malicious activity within your AWS environment. This includes issues such as escalation of privileges, use of exposed credentials, or communication with malicious IP addresses, domains, presence of malware on your Amazon EC2 instances and container workloads, or discovery of unusual patterns of login events on your database.

For example, GuardDuty can detect potentially compromised EC2 instances and container workloads serving malware, or mining bitcoin. It also monitors AWS account access behavior for signs of potential compromise, such as unauthorized infrastructure deployments – instances deployed in a Region that hasn't been used before, or unusual API calls – password policy changed to reduce password strength.

When enabled, GuardDuty provides visibility into the security posture of your AWS environment. When it identifies a potential security risk, it generates a finding and provides further details. You can also set up Amazon EventBridge to receive notifications when GuardDuty generates a finding. GuardDuty also recommends steps to remediate the indicative security issues in your environment.

You can export the generated findings to an Amazon Simple Storage Service (Amazon S3) bucket. GuardDuty also integrates with other AWS security-related services, such as AWS Security Hub and Amazon Detective, that can further help you analyze and investigate the security trends in your environment.

Using GuardDuty

You can use 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. The GuardDuty console provides access to your GuardDuty account, data, and resources.

AWS command line tools

With AWS command line tools, you can issue commands at your system's command line to perform GuardDuty tasks and AWS tasks. The command line tools are useful if you want to build scripts that perform tasks.

For information about installing and using AWS CLI, see AWS Command Line Interface User Guide. To view the available AWS CLI commands for GuardDuty, see CLI command reference.

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.

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.

Pricing for GuardDuty

When using GuardDuty for the first time, there is a 30-day free trial for each AWS account per AWS Region. For more information, see Pricing.

Supported AWS Regions

For information about AWS Regions where you can enable GuardDuty, see Regions and endpoints.

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 threat detection service that monitors Foundational data sources such as AWS CloudTrail event logs, AWS CloudTrail management events, Amazon VPC Flow Logs, and DNS logs. GuardDuty also analyzes features associated with its protection types only if you enable them separately. Features include Kubernetes audit logs, RDS login activity, S3 logs, EBS volumes, Runtime monitoring, and Lambda network activity logs. Using these data sources and features (if enabled), GuardDuty generates security findings for your account.

After you enable GuardDuty, it starts monitoring your environment. You can disable GuardDuty for any account in any Region, at any time. This will stop GuardDuty from processing the foundational data sources and any features that were enabled separately.

You do not need to enable any of the Foundational data sources explicitly. Amazon GuardDuty pulls independent streams of data directly from those services. For a new GuardDuty account, all the available protection types that are supported in an AWS Region are enabled and included in the 30-day free trial period by default. You can opt out of any or all of them. If you're an existing GuardDuty customer, you can choose to enable any or all of the protection plans that are available in your AWS Region. For more information, see Features associated with each protection type in GuardDuty.

When enabling GuardDuty, consider the following items:

- 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 where GuardDuty is available, see Regions and endpoints.
- 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 role, user, or group to manage GuardDuty specifically. For information about the permissions required to enable GuardDuty see Permissions required to enable GuardDuty.
- When you enable GuardDuty for the first time in any AWS Region, by default, it also enables all the available protection types that are supported in that Region, including Malware Protection. GuardDuty 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 the Foundational data sources to generate security findings. Malware Protection creates another service–linked role for your account called `AWSServiceRoleForAmazonGuardDutyMalwareProtection`. This role includes the permissions and trust policies that allow Malware Protection perform agentless scans to detect malware in your GuardDuty account. It allows GuardDuty to create an EBS volume snapshot in your account, and share that snapshot with the GuardDuty service account. For more information, see Service-linked role permissions for GuardDuty. 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 want 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 GuardDuty administrator account.

To designate a delegated GuardDuty administrator account

1. Open the AWS Organizations console at https://console.aws.amazon.com/organizations/, using the 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.

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 dataset 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.

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 **Summary** to view the insights about the findings generated in your AWS environment. For more information about the components of the Summary dashboard, see Summary dashboard.
4. In the navigation pane, choose **Findings**. The sample findings are displayed on the **Current findings** page with the prefix **[SAMPLE]**.
5. 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 about the available fields across all finding types see Finding details. 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 Management Console for actionable resources.
        - Select the + or - looking glass icons to create an inclusive or exclusive filter for that detail. For more information about finding filters see Filtering findings.
6. Archive all your sample findings

    a. Select all findings by selection the check box at the top of the list.

    b. Deselect any findings that 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 exporting GuardDuty findings to an Amazon S3 bucket

GuardDuty recommends configuring settings to export findings because it allows you to export your findings to an S3 bucket for indefinite storage beyond the GuardDuty 90-day retention period. This allows you to keep records of findings or track issues within your AWS 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.

To configure S3 export findings option

1. To encrypt the findings, you'll need a KMS key with a policy that allows GuardDuty to use that key for encryption. The following steps will help you create a new KMS key. If you are using a KMS key from another account, you need to apply the key policy by logging in to the AWS account that owns the key. The Region of your KMS key and S3 bucket must be the same. However, you can use this same bucket and key pair for each Region from where you want to export findings.

    a. Open the AWS KMS console at https://console.aws.amazon.com/kms.

    b. To change the AWS Region, use the Region selector in the upper-right corner of the page.

    c. In the navigation pane, choose **Customer managed keys**.

    d. Choose **Create key**.

    e. Choose **Symmetric** under **Key type**, and then choose **Next**.

Note

For detailed steps about creating your KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

f. Provide an **Alias** for your key, and then choose **Next**.

g. Choose **Next**, and then again choose **Next** to accept the default administration and usage permissions.

h. After you **Review** the configuration, choose **Finish** to create the key.

i. On the **Customer managed keys** page, choose your key alias.

j. In the **Key policy** tab, choose **Switch to policy view**.

k. Choose **Edit** and add the following key policy to your KMS key, granting GuardDuty access to your key. This statement allows GuardDuty to use only the key to which you add this policy. When editing the key policy, ensure that the 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:Region1:444455556666:key/KMSKeyId",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "111122223333",
            "aws:SourceArn":
 "arn:aws:guardduty:Region2:111122223333:detector/SourceDetectorID"
        }
    }
}
```

Replace *Region1* with the Region of your KMS key. Replace *444455556666* with the AWS account that owns the KMS key. Replace *KMSKeyId* with the key ID of the KMS key that you chose for encryption. To identify all these values – Region, AWS account, and key ID, view the **ARN** of your KMS key. To locate the key ARN, see Finding the key ID and ARN.

Similarly, replace *111122223333* with the AWS account of the GuardDuty account. Replace *Region2* with the Region of the GuardDuty account. Replace *SourceDetectorID* with the detector ID of the GuardDuty account for *Region2*.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

l. Choose **Save**.

2. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
3. In the navigation pane, choose **Settings**.
4. Under **Findings export options**, choose **Configure now**.
5. Choose **New bucket**. Provide a unique name for your S3 bucket.
6. (Optional) you can test your new export settings by generating sample findings. In the navigation pane, choose **Settings**.
7. Under the **Sample findings** section, choose **Generate sample findings**. The new sample findings will appear as entries in the S3 bucket created by GuardDuty in up to five minutes.

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 initiate 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 Amazon Chime, and also modify the types of findings alerts are sent for, see Setup an Amazon SNS topic and endpoint.

To create an SNS topic for your findings alerts

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the navigation pane, choose **Topics**.
3. Choose **Create Topic**.

4. For **Type**, select **Standard**.
5. For **Name**, enter **`GuardDuty`**.
6. Choose **Create Topic**. The topic details for your new topic will open.
7. In the **Subscriptions** section, choose **Create subscription**.
8. For **Protocol**, choose **Email**.
9. For **Endpoint**, enter the email address to send notifications to.
10. Choose **Create subscription**.

    After you create your subscription, you must confirm the subscription through email.
11. To check for a subscription message, go to your email inbox, and in the subscription message, choose **Confirm subscription**.

    > **Note**
    >
    > To check the email confirmation status, go to the SNS console and choose **Subscriptions**.

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. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.

    A rule can't have the same name as another rule in the same Region and on the same event bus.
5. For **Event bus**, choose **default**.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS events**.
9. For **Event pattern**, choose **Event pattern form**.
10. For **Event source**, choose **AWS services**.
11. For **AWS service**, choose **GuardDuty**.

12. For **Event Type**, choose **GuardDuty Finding**.
13. Choose **Next**.
14. For **Target types**, choose **AWS service**.
15. For **Select a target**, choose **SNS topic**, and for **Topic**, choose the name of the SNS topic you created earlier.
16. In the **Additional settings** section, for **Configure target input**, choose **Input transformer**.

    Adding an input transformer formats the JSON finding data sent from GuardDuty into a human-readable message.
17. Choose **Configure input transformer**.
18. In the **Target input transformer** section, for **Input path**, paste the following code:

    ```
    {
      "severity": "$.detail.severity",
      "Finding_ID": "$.detail.id",
      "Finding_Type": "$.detail.type",
      "region": "$.region",
      "Finding_description": "$.detail.description"
    }
    ```
19. To format the email, for **Template**, paste the following code and make sure to replace the text in red with the values appropriate to your Region:

    ```
    "You have a severity severity GuardDuty finding type Finding_Type in
     the Region_Name 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%3DFinding_ID"
    ```
20. Choose **Confirm**.
21. Choose **Next**.
22. (Optional) Enter one or more tags for the rule. For more information, see [Amazon EventBridge tags]() in the *Amazon EventBridge User Guide*.
23. Choose **Next**.

24. Review the details of the rule and choose **Create rule**.
25. (Optional) 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 Finding types.

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.
- If you are receiving findings for expected behavior in your environment, you can automatically archive findings based on the criteria you define with suppression rules.
- 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.

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 account** 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 10,000 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 account 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.

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 AWS Region in which GuardDuty operates. When you enable GuardDuty in a Region, a new detector with a unique 32 alphanumeric detectorId is generated in that Region. The format of a detectorId is `12abc34d567e8fa901bc2d34e56789f0`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

> **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 an unauthorized or unexpected activity in your AWS environment. GuardDuty analyzes and processes data from AWS CloudTrail event logs, AWS CloudTrail management events, AWS CloudTrail data events for S3, VPC flow logs, DNS logs, EKS audit logs, RDS login activity monitoring, and EBS volumes. For more information, see Foundational data sources.

Feature

A feature object configured for your GuardDuty protection plan helps to detect an unauthorized or unexpected activity in your AWS environment. Each GuardDuty protection plan configures the corresponding feature object to analyze and processe data. Some of the feature objects include EKS audit logs, RDS login activity monitoring, and EBS volumes. For more information, see Features activation in GuardDuty.

Finding

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

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.

Scan options

When GuardDuty Malware Protection is enabled, it allows you to specify which Amazon EC2 instances and Amazon Elastic Block Store(EBS) volumes to scan or skip. This feature lets you add the existing tags that are associated with your EC2 instances and EBS volume to either an inclusion tags list or exclusion tags list. The resources associated to the tags that you add to an inclusion tags list, are scanned for malware, and those added to an exclusion tags list are not scanned. For more information, see Scan options with user-defined tags.

Snapshots retention

When GuardDuty Malware Protection is enabled, it provides an option to retain the snapshots of your EBS volumes in your AWS account. GuardDuty generates the replica EBS volumes based on the snapshots of your EBS volumes. You can retain the snapshots of your EBS volumes only if the Malware Protection scan detects malware in the replica EBS volumes. If no malware is detected in the replica EBS volumes, GuardDuty automatically deletes the snapshots of your EBS volumes, irrespective of the snapshots retention setting. For more information, see Snapshots retention.

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.

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.

Threat IP list

A list of known malicious IP addresses. In addition to generating findings because of a potentially suspicious activity, GuardDuty also generates findings based on these threat lists. For more information, see Working with trusted IP lists and threat lists.

Features activation in GuardDuty

When you enable Amazon GuardDuty for the first time or enable a protection type within GuardDuty, GuardDuty starts processing the corresponding Foundational data sources within your AWS environment. GuardDuty uses these data sources to process a stream of events, such as VPC flow logs, DNS logs, and AWS CloudTrail event and management logs. It then analyzes these events to identify potential security threats and generates findings in your account.

In addition to log data sources, GuardDuty can use additional data from other AWS services in your AWS environment to monitor and analyze for potential security threats.

Feature activation

When you add additional GuardDuty protections, for example, S3 Protection, Runtime Monitoring, or EKS Protection, you can configure the GuardDuty feature corresponding to the protection type. Historically, GuardDuty protections were called `dataSources` in the APIs. However, after March 2023, new GuardDuty protection types are now configured as `features` and not `dataSources`. GuardDuty still supports configuring protection types launched before March 2023, as `dataSources` through the API, but new protection types are only available as `features`.

If you manage GuardDuty configuration and protection types through the console, you are not directly impacted by this change and don't need to take any action. Feature activation affects the behavior of the APIs that are invoked to enable GuardDuty or protection types within GuardDuty. For more information, see GuardDuty API changes.

GuardDuty API changes in March 2023

The GuardDuty APIs configure protection features that don't belong to the list of Foundational data sources. A feature object contains feature details, such as feature name and status, and may contain additional configuration for some of the features. This migration affects the following APIs in the *Amazon GuardDuty API Reference*:

- CreateDetector
- GetDetector
- UpdateDetector
- GetMemberDetectors

- UpdateMemberDetectors
- DescribeOrganizationConfiguration
- UpdateOrganizationConfiguration
- GetRemainingFreeTrialDays
- GetUsageStatistics

Features activation compared to data sources

Historically, all GuardDuty features were passed through a `dataSources` object in the API. From March 2023, GuardDuty prefers `features` object instead of the `dataSources` object in the API. All earlier data sources have corresponding features, but newer features may not have corresponding data sources.

The following list shows the comparison between `dataSources` and `features` object when passed through an API:

- The `dataSources` object contains objects for each protection type and its status. The `features` object is a list of available features that correspond to each protection type within GuardDuty.

  Starting March 2023, feature activation will be the only way to configure new GuardDuty features in your AWS environment.
- The `dataSources` schema in the API request or response is the same in each AWS Region where GuardDuty is available. However, every feature may not be available in each Region. Therefore, the available feature names may differ based on the Region.

Understanding how feature activation works

The GuardDuty APIs will continue to return a `dataSources` object as applicable, and they will also return a `features` object containing the same information in a different format. GuardDuty features launched before March 2023 will be available through `dataSources` object and `features` object. GuardDuty launched features since March 2023 will only be available through the `features` object. You can't create or update a detector, or describe your AWS Organizations using both `dataSources` and `features` object notation in the same API request. To enable GuardDuty protection types, you will need to migrate your existing data sources to the `features` object by using the same APIs that now include the `features` object too.

> **Note**
>
> GuardDuty will not add new data source after this modification.

GuardDuty has deprecated the use of data sources. However, it still supports the Foundational data sources. The GuardDuty best practices recommend using features activation for any protection types that are already enabled for your account. The best practices also require using features activation when you enable a new protection type for your account.

Incorporating features activation changes

- If you manage GuardDuty configurations through APIs, SDKs, or AWS CloudFormation template, and want to enable potential new GuardDuty features, you will need to modify your code and template, respectively. For more information, see the updated APIs in the *Amazon GuardDuty API Reference*.
- For GuardDuty features configured prior to this upgrade, you can continue using the APIs, SDKs, or AWS CloudFormation template. However, we recommend that you switch to using `feature` object.

  All the data sources have an equivalent feature object. For more information, see Mapping `dataSources` to `features`.
- Presently, `additionalConfiguration` in the `features` object is only available for certain protection types.
  - For such protection types, if your feature's `AdditionalConfiguration` `status` is set to `ENABLED` but your feature's configuration `status` is not set to `ENABLED`, GuardDuty will not take any action in this case.
  - The following APIs get impacted by this:
    - UpdateDetector
    - UpdateMemberDetectors
    - UpdateOrganizationConfiguration

Mapping `dataSources` to `features`

The following table shows the mapping of protection types, `dataSources`, and `features`.

GuardDuty protection type	Data source name*	Feature name
VPC Flow Logs	`flowLogs` (read only; can't be modified)	`FLOW_LOGS` (read only; can't be modified)
DNS logs	`dnsLogs` (read only; can't be modified)	`DNS_LOGS` (read only; can't be modified)
CloudTrail events	`cloudLogs` (read only; can't be modified)	`CLOUD_LOGS` (read only; can't be modified)
S3	`s3Logs`	`S3_DATA_EVENTS`
EKS Audit Log Monitoring	`kubernetes.auditlogs`	`EKS_AUDIT_LOGS`
Malware Protection	`malwareProtection.scanEc2InstanceWithFindings.ebsVolumes`	`EBS_MALWARE_PROTECTION`
RDS Login events	GuardDuty provides only feature activation support for these protection types.	`RDS_LOGIN_EVENTS`
EKS Runtime Monitoring		`EKS_RUNTIME_MONITORING`
Runtime Monitoring		`RUNTIME_MONITORING`
GuardDuty security agent for Amazon EKS clusters		`EKS_RUNTIME_MONITORING.additionalConfiguratio`

GuardDuty protection type	Data source name*	Feature name
		`n.EKS_ADDON_MANAGEMENT` `RUNTIME_MONITORING.additionalConfiguration.EKS_ADDON_MANAGEMENT`
GuardDuty security agent for Amazon ECS-Fargate clusters		`RUNTIME_MONITORING.additionalConfiguration.ECS_FARGATE_AGENT_MANAGEMENT`
GuardDuty security agent for Amazon EC2 instances		`RUNTIME_MONITORING.additionalConfiguration.EC2_AGENT_MANAGEMENT`
Lambda Protection	`LAMBDA_NETWORK_LOGS`	

*GetUsageStatistics uses its own `dataSource` names. For more information, see Estimating GuardDuty cost or GetUsageStatistics.

Foundational data sources

GuardDuty uses the foundational data sources to detect communication with known malicious domains and IP addresses and identify anomalous behavior. While in transit from these sources to GuardDuty, all of the log data is encrypted. GuardDuty extracts various fields from these logs sources for profiling and anomaly detection, and then discards these logs.

The following sections describe how GuardDuty uses each supported data source. When you enable GuardDuty in your AWS account, GuardDuty automatically starts to monitor these log sources.

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 helps you identify which users and accounts invoked AWS APIs for services that support CloudTrail, the source IP address from where the calls were invoked, and the time at which the calls were invoked. For more information, see What is AWS CloudTrail in *AWS CloudTrail User Guide*.

GuardDuty also monitors CloudTrail management events. When you enable GuardDuty, it starts consuming CloudTrail management events directly from CloudTrail through an independent and duplicated stream of events and analyzes your CloudTrail event logs. There is no additional charge when GuardDuty accesses the events recorded in CloudTrail.

GuardDuty does not manage your CloudTrail events or affect your existing CloudTrail configurations. Similarly, your CloudTrail configurations don't affect how GuardDuty consumes and processes the event logs. To manage access and retention of your CloudTrail events, use the CloudTrail service console or API. For more information, see Viewing events with CloudTrail event history in *AWS CloudTrail User Guide*.

How GuardDuty handles AWS CloudTrail global events

For most AWS services, CloudTrail events are recorded in the AWS Region where they are created. For global services such as AWS Identity and Access Management (IAM), AWS Security Token Service (AWS STS), Amazon Simple Storage Service (Amazon S3), Amazon CloudFront, and Amazon Route 53 (Route 53), events are only generated in the Region where they occur but they have a global significance.

When GuardDuty consumes CloudTrail Global service events with security value such as network configurations or user permissions, it replicates those events and processes them in each Region where you have enabled GuardDuty. This behavior helps GuardDuty maintain user and role profiles in each Region, which is vital to detecting anomalous events.

We highly recommend that you enable GuardDuty in all AWS Regions which are enabled for your AWS account. This helps GuardDuty generate findings about unauthorized or unusual activity even in those Regions that you may not be using actively.

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)

VPC Flow Logs

The VPC Flow Logs feature of Amazon VPC captures information about the IP traffic going to and from network interfaces attached to the Amazon Elastic Compute Cloud (Amazon EC2) instances within your AWS environment.

When you enable GuardDuty, it immediately starts analyzing your VPC flow logs from Amazon EC2 instances within your account. 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 of your existing flow logs configuration.

GuardDuty Lambda Protection

Lambda Protection is an optional enhancement to Amazon GuardDuty. Presently, Lambda Network Activity Monitoring includes Amazon VPC flow logs from all Lambda functions for your account, even those logs that don't use VPC networking. To protect your Lambda function from potential security threats, you will need to configure Lambda Protection in your GuardDuty account. For more information, see GuardDuty Lambda Protection.

GuardDuty Runtime Monitoring

When you manage the security agent (either manually or through GuardDuty) in EKS Runtime Monitoring or Runtime Monitoring for EC2 instances, and GuardDuty is presently deployed on an Amazon EC2 instance and receives the Collected runtime event types from this instance, GuardDuty will not charge your AWS account for the analysis of VPC flow logs from this Amazon EC2 instance. This helps GuardDuty avoid double usage cost in the account.

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.

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 affect GuardDuty analysis.

> **Note**
>
> GuardDuty doesn't support monitoring DNS logs for Amazon EC2 instances that are launched on AWS Outposts because the Amazon Route 53 Resolver query logging feature is not available in that environment.

EKS Protection in Amazon GuardDuty

EKS Audit Log Monitoring helps you detect potentially suspicious activities in EKS clusters within Amazon Elastic Kubernetes Service (Amazon EKS). EKS Audit Log Monitoring uses Kubernetes audit logs to capture chronological activities from users, applications using the Kubernetes API, and the control plane. For more information, see Kubernetes audit logs.

> **Note**
>
> EKS Runtime Monitoring is managed as a part of Runtime Monitoring. For more information, see GuardDuty Runtime Monitoring.

Features in EKS Protection

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.

For more information, 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. 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 User Guide.

For information about configuring EKS Audit Log Monitoring, see EKS Audit Log Monitoring.

EKS Audit Log Monitoring

EKS Audit Log Monitoring helps you detect potentially suspicious activities in your EKS clusters within Amazon Elastic Kubernetes Service. When you enable EKS Audit Log Monitoring, GuardDuty immediately begins to monitor Kubernetes audit logs from your Amazon EKS clusters and analyze them for potentially malicious and suspicious activity. It consumes Kubernetes audit log events directly from the Amazon EKS control plane logging feature through an independent and duplicative stream of audit logs. This process does not require any additional set up or affect any existing Amazon EKS control plane logging configurations that you might have.

When you disable EKS Audit Log Monitoring, GuardDuty immediately stops monitoring and analyzing the Kubernetes audit logs for your Amazon EKS resources.

EKS Audit Log Monitoring may not be available in all the AWS Regions where GuardDuty is available. For more information, see Region-specific feature availability.

How 30-day free trial period affects GuardDuty accounts

- When you enable GuardDuty for the first time, EKS Audit Log Monitoring within EKS Protection is already included in the 30-day free trial period.
- The existing GuardDuty accounts can enable EKS Audit Log Monitoring for the first time with a 30-day free trial period.

Configuring EKS Audit Log Monitoring for a standalone account

Choose your preferred access method to enable or disable EKS Audit Log Monitoring for a standalone account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose EKS Protection.
3. Under the **Configuration** tab, you can view the current configuration status of EKS Audit Log Monitoring. In the **EKS Audit Log Monitoring** section, choose **Enable** to enable or **Disable** to disable the EKS Audit Log Monitoring feature.
4. Choose **Save**.

API/CLI

- Run the updateDetector API operation using the regional detector ID of the delegated GuardDuty administrator account and passing the `features` object name as `EKS_AUDIT_LOGS` and status as `ENABLED` or `DISABLED`.

Alternatively, You can also enable or disable EKS Audit Log Monitoring running the a AWS CLI command. The following example code enables GuardDuty EKS Audit Log Monitoring. To disable it, replace `ENABLED` with `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
features [{"Name" : "EKS_AUDIT_LOGS", "Status" : "ENABLED"}]'
```

Configuring EKS Audit Log Monitoring in multiple-account environments

In a multiple-account environment, only the delegated GuardDuty administrator account has the option to enable or disable the EKS Audit Log Monitoring; feature for the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account manages their member accounts using AWS Organizations. This delegated GuardDuty administrator account can choose to auto-enable EKS Audit Log Monitoring for all the new accounts as they join the organization. For more information about multiple-account environments, see Managing multiple accounts in Amazon GuardDuty.

Configuring EKS Audit Log Monitoring for delegated GuardDuty administrator account

Choose your preferred access method to configure EKS Audit Log Monitoring for the delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the management account credentials.

2. In the navigation pane, choose EKS Protection.

3. Under the **Configuration** tab, you can view the current configuration status of EKS Audit Log Monitoring in the respective section. To update the configuration for delegated GuardDuty administrator account, choose **Edit** in the **EKS Audit Log Monitoring** pane.

4. Do one of the following:

Using Enable for all accounts

- Choose **Enable for all accounts**. This will enable the protection plan for all the active GuardDuty accounts in your AWS organization, including the new accounts that join the organization.
- Choose **Save**.

Using Configure accounts manually

- To enable the protection plan only for the delegated GuardDuty administrator account account, choose **Configure accounts manually**.
- Choose **Enable** under the **delegated GuardDuty administrator account (this account)** section.
- Choose **Save**.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `EKS_AUDIT_LOGS` and `status` as `ENABLED` or `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

You can enable or disable EKS Audit Log Monitoring by running the following AWS CLI command. Make sure to use delegated GuardDuty administrator account's valid *`detector ID`*.

> **Note**
>
> The following example code enables EKS Audit Log Monitoring. Make sure to replace *`12abc34d567e8fa901bc2d34e56789f0`* with the `detector-id` of the delegated GuardDuty administrator account and *`555555555555`* with the AWS account of the delegated GuardDuty administrator account.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --accountids 555555555555 --features '[{"Name": "EKS_AUDIT_LOGS", "Status":
 "ENABLED"}]'
```

To disable EKS Audit Log Monitoring, replace `ENABLED` with `DISABLED`.

Auto-enable EKS Audit Log Monitoring for all member accounts

Choose your preferred access method to enable the EKS Audit Log Monitoring for existing member accounts in your organization.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   Using the EKS Protection page

   1. In the navigation pane, choose **EKS Protection**.
   2. Under the **Configuration** tab, you can view the current status of EKS Audit Log Monitoring for active member accounts in your organization.

      To update the EKS Audit Log Monitoring configuration, choose **Edit**.
   3. Choose **Enable for all accounts**. This action automatically enables EKS Audit Log Monitoring for both the existing and new accounts in the organization.
   4. Choose **Save**.

   > **Note**
   >
   > It may take up to 24 hours to update the configuration for the member accounts.

   Using the Accounts page

   1. In the navigation pane, choose **Accounts**.

2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **EKS Audit Log Monitoring**.
4. Choose **Save**.

If you can't use the **Enable for all accounts** option and want to customize EKS Audit Log Monitoring configuration for specific accounts in your organization, see Selectively enable or disable EKS Audit Log Monitoring for member accounts.

API/CLI

- To selectively enable or disable EKS Audit Log Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable EKS Audit Log Monitoring for a single member account. To disable it, replace ENABLED with DISABLED.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"name": "EKS_AUDIT_LOGS", "status": "ENABLED"}]'
  ```

  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.

Enable EKS Audit Log Monitoring for all existing active member accounts

Choose your preferred access method to enable EKS Audit Log Monitoring for all existing active member accounts in the organization.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.

2. In the navigation pane, choose **EKS Protection**.
3. On the **EKS Protection** page, you can view the current status of the **GuardDuty-initiated malware scan** configuration. Under the **Active member accounts** section, choose **Actions**.
4. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
5. Choose **Save**.

API/CLI

- To selectively enable or disable EKS Audit Log Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable EKS Audit Log Monitoring for a single member account. To disable it, replace ENABLED with DISABLED.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"name": "EKS_AUDIT_LOGS", "status": "ENABLED"}]'
  ```

  > **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.

Auto-enable EKS Audit Log Monitoring for new member accounts

The newly added member accounts must **Enable** GuardDuty before selecting configuring GuardDuty-initiated malware scan. The member accounts managed by invitation can configure GuardDuty-initiated malware scan manually for their accounts. For more information, see Step 3 - Accept an invitation.

Choose your preferred access method to enable EKS Audit Log Monitoring for new accounts that join your organization.

Console

The delegated GuardDuty administrator account can enable EKS Audit Log Monitoring for new member accounts in an organization, using either the **EKS Audit Log Monitoring** or **Accounts** page.

To auto-enable EKS Audit Log Monitoring for new member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.
2. Do one of the following:
   - Using the **EKS Protection** page:
     1. In the navigation pane, choose **EKS Protection**.
     2. On the **EKS Protection** page, choose **Edit** in the **EKS Audit Log Monitoring**.
     3. Choose **Configure accounts manually**.
     4. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, EKS Audit Log Monitoring will be automatically enabled for their account. Only the organization delegated GuardDuty administrator account can modify this configuration.
     5. Choose **Save**.
   - Using the **Accounts** page:
     1. In the navigation pane, choose **Accounts**.
     2. On the **Accounts** page, choose **Auto-enable** preferences.
     3. In the **Manage auto-enable preferences** window, select **Enable for new accounts** under **EKS Audit Log Monitoring**.

4. Choose **Save**.

API/CLI

- To selectively enable or disable EKS Audit Log Monitoring for your new accounts, run the UpdateOrganizationConfiguration API operation using your own *detector ID*.
- The following example shows how you can enable EKS Audit Log Monitoring for the new members that join your organization. You can also pass a list of account IDs separated by a space.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-organization-configuration --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable --features '[{"Name":
   "EKS_AUDIT_LOGS", "AutoEnable": "NEW"}]'
  ```

Selectively enable or disable EKS Audit Log Monitoring for member accounts

Choose your preferred access method to enable or disable EKS Audit Log Monitoring for selective member accounts in your organization.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. In the navigation pane, choose **Accounts**.

   On the **Accounts** page, review the **EKS Audit Log Monitoring** column for the status of your member account.

3. **To enable or disable EKS Audit Log Monitoring**

   Select an account that you want to configure for EKS Audit Log Monitoring. You can select multiple accounts at a time. Under the **Edit Protection Plans** dropdown, choose **EKS Audit Log Monitoring**, and then choose the appropriate option.

API/CLI

To selectively enable or disable EKS Audit Log Monitoring for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.

The following example shows how you can enable EKS Audit Log Monitoring for a single member account. To disable it, replace ENABLED with DISABLED. You can also pass a list of account IDs separated by a space.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --accountids 111122223333 --features '[{"Name": "EKS_AUDIT_LOGS", "Status":
 "ENABLED"}]'
```

Lambda Protection in Amazon GuardDuty

Lambda Protection helps you identify potential security threats when an AWS Lambda function gets invoked in your AWS environment. When you enable Lambda Protection, GuardDuty starts monitoring Lambda network activity logs, starting with VPC Flow Logs from all Lambda functions for account, including those logs that don't use VPC networking, and are generated when the Lambda function gets invoked. If GuardDuty identifies suspicious network traffic that is indicative of the presence of a potentially malicious piece of code in your Lambda function, GuardDuty will generate a finding.

> **Note**
>
> Lambda Network Activity Monitoring doesn't include the logs for Lambda@Edge functions.

You can configure Lambda Protection for any account or available AWS Regions, at any time. By default, an existing GuardDuty account can enable Lambda Protection with a 30-day trial period. For a new GuardDuty account, Lambda Protection is already enabled and included in the 30-day trial period. For information about usage statistics, see Estimating cost.

GuardDuty monitors network activity logs generated by invoking the Lambda functions. Presently, Lambda Network Activity Monitoring includes Amazon VPC flow logs from all Lambda functions for your account, including those logs that don't use VPC networking, and are subject to change, including expansion to other network activity such as DNS query data generated by invoking the Lambda functions. The expansion into other forms of network activity monitoring will increase the volume of data that GuardDuty will process for Lambda Protection. This will directly impact the usage cost of Lambda Protection. Whenever GuardDuty starts monitoring an additional network activity log, it will provide a notice to the accounts that have turned on Lambda Protection, at least 30 days prior to the release.

Feature in Lambda Protection

Lambda Network Activity Monitoring

When you enable Lambda Protection, GuardDuty monitors Lambda network activity logs generated when a Lambda function associated to your account gets invoked. This helps you detect potential security threats to the Lambda function. GuardDuty monitors VPC flow logs from all your Lambda functions, including those that don't use VPC networking. For Lambda functions that are configured to use VPC networking, you don't need to enable VPC flow logs for the elastic network interfaces (ENI) created by Lambda for GuardDuty. GuardDuty only charges for the amount of Lambda network activity logs data processed (in GB) to generate a finding. GuardDuty optimizes cost by applying smart filters and analyzing a subset of Lambda network activity logs that are relevant to threat detection. For information about pricing, see Amazon GuardDuty pricing.

GuardDuty doesn't manage your Lambda network activity logs (including VPC and non-VPC flow logs) or make them accessible in your account.

Configuring Lambda Protection

Configuring Lambda Protection for a standalone account

For accounts associated with AWS Organizations, you can automate this process through GuardDuty console or API instructions, as described in the next section.

Choose your preferred access method to enable or disable Lambda Protection for a standalone account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Settings**, choose **Lambda Protection**.
3. The Lambda Protection page shows the current status for your account. You may enable or disable the feature at any time by selecting **Enable** or **Disable**.
4. Choose **Save**.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `LAMBDA_NETWORK_LOGS` and `status` as `ENABLED` or `DISABLED`.

You can also enable or disable Lambda Network Activity Monitoring by running the following AWS CLI command. Make sure to use your own valid *`detector ID`*.

> **Note**
>
> The following example code enables Lambda Network Activity Monitoring. To disable it, replace ENABLED with DISABLED.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
features [{"Name" : "LAMBDA_NETWORK_LOGS", "Status" : "ENABLED"}]'
```

Configuring Lambda Protection in multi-account environments

In a multi-account environment, only the delegated GuardDuty administrator account has the option to enable or disable Lambda Protection for the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account manages member accounts using AWS Organizations. The delegated GuardDuty administrator account can choose to auto-enable Lambda Network Activity Monitoring for all the new accounts as they join the organization. For more information about multi-account environments, see Managing multiple accounts in Amazon GuardDuty.

Configuring Lambda Protection for delegated GuardDuty administrator account

Choose your preferred access method to enable or disable Lambda Network Activity Monitoring for delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the management account credentials.
2. In the navigation pane, under **Settings**, choose **Lambda Protection**.
3. On the **Lambda Protection** page, choose **Edit**.
4. Do one of the following:

Using Enable for all accounts

- Choose **Enable for all accounts**. This will enable the protection plan for all the active GuardDuty accounts in your AWS organization, including the new accounts that join the organization.
- Choose **Save**.

Using Configure accounts manually

- To enable the protection plan only for the delegated GuardDuty administrator account account, choose **Configure accounts manually**.
- Choose **Enable** under the **delegated GuardDuty administrator account (this account)** section.
- Choose **Save**.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `LAMBDA_NETWORK_LOGS` and `status` as `ENABLED` or `DISABLED`.

You can enable or disable Lambda Network Activity Monitoring by running the following AWS CLI command. Make sure to use delegated GuardDuty administrator account's valid *`detector ID`*.

> **Note**
>
> The following example code enables Lambda Network Activity Monitoring. To disable it, replace `ENABLED` with `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-ids 555555555555 --features '[{"Name": "LAMBDA_NETWORK_LOGS", "Status":
 "ENABLED"}]'
```

Auto-enable Lambda Network Activity Monitoring for all member accounts

Choose your preferred access method to enable the Lambda Network Activity Monitoring feature for all member accounts. This includes existing member accounts and the new accounts that join the organization.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   Using the Lambda Protection page

   1. In the navigation pane, choose **Lambda Protection**.
   2. Choose **Enable for all accounts**. This action automatically enables Lambda Network Activity Monitoring for both existing and new accounts in the organization.
   3. Choose **Save**.

   > **Note**
   >
   > It may take up to 24 hours to update the configuration for the member accounts.

   Using the Accounts page

   1. In the navigation pane, choose **Accounts**.
   2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
   3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **Lambda Network Activity Monitoring**.

   > **Note**
   >
   > By default, this action automatically turns on the **Auto-enable GuardDuty for new member accounts** option.

4. Choose **Save**.

   If you can't use the **Enable for all accounts** option, see Selectively enable or disable Lambda Network Activity Monitoring for member accounts.

API/CLI

- To selectively enable or disable Lambda Network Activity Monitoring for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable Lambda Network Activity Monitoring for a single member account. To disable a member account, replace `ENABLED` with `DISABLED`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"Name": "LAMBDA_NETWORK_LOGS", "Status": "ENABLED"}]'
  ```

  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.

Enable Lambda Network Activity Monitoring for all existing active member accounts

Choose your preferred access method to enable Lambda Network Activity Monitoring for all the existing active member accounts in the organization.

Console

To configure Lambda Network Activity Monitoring for all existing active member accounts

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.

2. In the navigation pane, choose **Lambda Protection**.
3. On the **Lambda Protection** page, you can view the current status of the configuration. Under the **Active member accounts** section, choose **Actions**.
4. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
5. Choose **Confirm**.

API/CLI

- To selectively enable or disable Lambda Network Activity Monitoring for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable Lambda Network Activity Monitoring for a single member account. To disable a member account, replace ENABLED with DISABLED.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"Name": "LAMBDA_NETWORK_LOGS", "Status": "ENABLED"}]'
  ```

  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.

Auto-enable Lambda Network Activity Monitoring for new member accounts

Choose your preferred access method to enable Lambda Network Activity Monitoring for new accounts that join your organization.

Console

The delegated GuardDuty administrator account can enable Lambda Network Activity Monitoring for new member accounts in an organization, using either the **Lambda Protection** or **Accounts** page.

To auto-enable Lambda Network Activity Monitoring for new member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   - Using the **Lambda Protection** page:

     1. In the navigation pane, choose **Lambda Protection**.
     2. On the **Lambda Protection** page, choose **Edit**.
     3. Choose **Configure accounts manually**.
     4. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, Lambda Protection will be automatically enabled for their account. Only the organization delegated GuardDuty administrator account can modify this configuration.
     5. Choose **Save**.

   - Using the **Accounts** page:

     1. In the navigation pane, choose **Accounts**.
     2. On the **Accounts** page, choose **Auto-enable** preferences.
     3. In the **Manage auto-enable preferences** window, select **Enable for new accounts** under **Lambda Network Activity Monitoring**.
     4. Choose **Save**.

API/CLI

- To enable or disable Lambda Network Activity Monitoring for new member accounts, invoke the UpdateOrganizationConfiguration API operation using your own *`detector ID`*.
- The following example shows how you can enable Lambda Network Activity Monitoring for a single member account. To disable it, see Selectively enable or disable Lambda Network Activity Monitoring for member accounts. If you don't want to enable it for all the new accounts joining the organization, set `AutoEnable` to `NONE`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable --features '[{"Name":
 "LAMBDA_NETWORK_LOGS", "AutoEnable": "NEW"}]'
```

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.

Selectively enable or disable Lambda Network Activity Monitoring for member accounts

Choose your preferred access method to selectively enable or disable Lambda Network Activity Monitoring for member accounts.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.
2. In the navigation pane, under **Settings**, choose **Accounts**.

   On the **Accounts** page, review the **Lambda Network Activity Monitoring** column. It indicates whether or not Lambda Network Activity Monitoring is enabled.
3. Choose the account for which you want to configure Lambda Protection. You can choose multiple accounts at a time.
4. From the **Edit Protection Plans** dropdown menu, choose **Lambda Network Activity Monitoring**, and then choose an appropriate action.

API/CLI

Invoke the updateMemberDetectors API using your own *`detector ID`*.

The following example shows how you can enable Lambda Network Activity Monitoring for a single member account. To disable it, replace `ENABLED` with `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 111122223333 --features '[{"Name": "LAMBDA_NETWORK_LOGS", "Status":
 "ENABLED"}]'
```

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.

Malware Protection in Amazon GuardDuty

Malware Protection helps you detect the potential presence of malware by scanning the Amazon Elastic Block Store (Amazon EBS) volumes that are attached to the Amazon Elastic Compute Cloud (Amazon EC2) instances and container workloads. Malware Protection provides scan options where you can decide if you want to include or exclude specific Amazon EC2 instances and container workloads at the time of scanning. It also provides an option to retain the snapshots of Amazon EBS volumes attached to the Amazon EC2 instances or container workloads, in your GuardDuty accounts. The snapshots get retained only when malware is found and Malware Protection findings are generated.

Malware Protection offers two types of scans to detect potentially malicious activity in your Amazon EC2 instances and container workloads – GuardDuty-initiated malware scan and On-demand malware scan. The following table shows the comparison between both the scan types.

Factor	**GuardDuty-initiated malware scan**	**On-demand malware scan**
How the scan gets invoked	After you enable GuardDuty-initiated malware scan, whenever GuardDuty generates a finding that indicates the potential presence of malware in an Amazon EC2 instance or a container workload, GuardDuty automatically initiates an agentless malware scan on the Amazon EBS volumes attached to your potentially impacted resource. For more information, see GuardDuty-initiated malware scan.	You can initiate an On-demand malware scan by providing the Amazon Resource Name (ARN) associated with your Amazon EC2 instance or container workload. You can initiate an On-demand malware scan even when no GuardDuty finding is generated for your resource. For more information, see On-demand malware scan.
Configuration needed	To use GuardDuty-initiated malware scan, you must	Your account must have GuardDuty enabled. To use

Factor	GuardDuty-initiated malware scan	On-demand malware scan
	enable it for your account. For more information, see Configuring GuardDuty-initiated malware scan.	On-demand malware scan, there is no configuration required at the feature-level.
Wait time to initiate a new scan	Whenever GuardDuty generates one of the Findings that invoke GuardDuty-initiated malware scan, a malware scan initiates automatically only once every 24 hours.	You can initiate an On-demand malware scan on the same resource any time after 1 hour from the start time of the previous scan.
Availability of the 30-day free trial period	When you enable GuardDuty-initiated malware scan for the first time in your account, you can use a 30-day free trial period*.	There is no free trial period* with On-demand malware scan for new or existing GuardDuty accounts.
Scan options	After you've configured GuardDuty-initiated malware scan, Malware Protection also helps you to select which resources to scan or skip. Malware Protection will not initiate an automatic scan on the resources that you choose to exclude from scanning.	On-demand malware scan supports a global tag – `GuardDutyExcluded` . Scan options with user-defined tags is not applicable to On-demand malware scan because you provide the resource ARN manually.

*You will incur usage cost for creating EBS volume snapshots and retaining snapshots. For more information about configuring your account to retain snapshots, see Snapshots retention.

Malware Protection is an optional enhancement to GuardDuty, and is designed in a way that it won't affect the performance of your resources. For information about how Malware Protection works within GuardDuty, see Feature in Malware Protection. For information about availability of Malware Protection in different AWS Regions, see Regions and endpoints.

> **Note**
>
> GuardDuty Malware Protection doesn't support Fargate with either Amazon EKS or Amazon ECS.

Feature in Malware Protection

Elastic Block Storage (EBS) volume

This section explains how Malware Protection, including both GuardDuty-initiated malware scan and On-demand malware scan, scans the Amazon EBS volumes associated with your Amazon EC2 instances and container workloads. Before proceeding, consider the following customizations:

- Scan options – Malware Protection offers the capability to specify tags to either include or exclude Amazon EC2 instances and Amazon EBS volumes from the scanning process. Only GuardDuty-initiated malware scan supports scan options with user-defined tags. Both GuardDuty-initiated malware scan and On-demand malware scan support the global `GuardDutyExcluded` tag. For more information, see Scan options with user-defined tags.
- Snapshots retention – Malware Protection provides an option to retain the snapshots of your Amazon EBS volumes in your AWS account. By default, this option is turned off. You can opt in for snapshots retention for both GuardDuty initiated and on-demand malware scans. For more information, see Snapshots retention.

When GuardDuty generates a finding that is indicative of potential presence of malware in an Amazon EC2 instance or a container workload and you have enabled the GuardDuty initiated scan type within Malware Protection, a GuardDuty-initiated malware scan may get invoked on the basis of your scan options.

To initiate an On-demand malware scan on the Amazon EBS volumes associated with an Amazon EC2 instance, provide the Amazon Resource Name (ARN) of the Amazon EC2 instance.

As a response to an On-demand malware scan or automatically invoked GuardDuty-initiated malware scan, GuardDuty creates snapshots of the relevant EBS volumes attached to the potentially impacted resource, and shares them with the GuardDuty service account. From these snapshots, GuardDuty creates an encrypted replica EBS volume in the service account.

After the scan completes, GuardDuty deletes the encrypted replica EBS volumes and the snapshots of your EBS volumes. If malware is found and you've turned on the snapshots retention setting, the snapshots of your EBS volumes won't get deleted and are automatically retained in your AWS account. When no malware is found, the snapshots of your EBS volumes will not be retained, regardless of the snapshots retention setting. By default, the snapshots retention setting is turned off. For information about the costs of snapshots and their retention, see Amazon EBS pricing.

GuardDuty will retain each replica EBS volume in the service account for up to 55 hours. If there is a service outage, or failure with a replica EBS volume and its malware scan, GuardDuty will retain such an EBS volume for no more than seven days. The extended volume retention period is to triage and address the outage or failure. GuardDuty Malware Protection will delete the replica EBS volumes from the service account after the outage or failure is addressed, or once the extended retention period lapses.

Supported Amazon EBS volumes for malware scan

In all of the AWS Regions where GuardDuty supports the Malware Protection feature, you can scan the Amazon EBS volumes that are unencrypted or encrypted. You can have Amazon EBS volumes that are encrypted with either AWS managed key or customer managed key. Presently, some of the AWS Regions support both the ways to encrypt your Amazon EBS volumes, while others support only customer managed key.

For more information where this capability is not yet supported, see China Regions

The following list describes the key that GuardDuty uses whether or not your Amazon EBS volumes are encrypted:

- **Amazon EBS volumes that are either unencrypted or encrypted with AWS managed key –** GuardDuty uses its own key to encrypt the replica Amazon EBS volumes.

  When your account belongs to an AWS Region that doesn't support scanning Amazon EBS volumes that are encrypted with the default AWS managed key for EBS, see Modifying default AWS KMS key ID of an Amazon EBS volume.
- **Amazon EBS volumes that are encrypted with customer managed key –** GuardDuty uses the same key to encrypt the replica EBS volume.

Malware Protection doesn't support scanning Amazon EC2 instances with `productCode` as `marketplace`. If a malware scan gets initiated for such an Amazon EC2 instance, the scan will be skipped. For more information, see `UNSUPPORTED_PRODUCT_CODE_TYPE` in Reasons for skipping resource during malware scan.

Modifying default AWS KMS key ID of an Amazon EBS volume

By default, invoking the CreateVolume API with encryption set to `true` and not specifying the KMS key ID, creates an Amazon EBS volume that gets encrypted with the default AWS KMS key for EBS encryption. However, when an encryption key is not provided explicitly, you can modify the default key by invoking the ModifyEbsDefaultKmsKeyId API or by using the corresponding AWS CLI command.

To modify the EBS default key ID, add the following necessary permission to your IAM policy – `ec2:modifyEbsDefaultKmsKeyId`. Any newly-created Amazon EBS volume that you choose to be encrypted but don't specify an associated KMS key ID, will use the default key ID. Use one of the following methods to update the EBS default key ID:

To modify default KMS key ID of an Amazon EBS volume

Do one of the following:

- **Using an API** – You can use the ModifyEbsDefaultKmsKeyId API. For information about how you can view the encryption status of your volume, see Create Amazon EBS volume.
- **Using AWS CLI command** – The following example modifies the default KMS key ID that will encrypt Amazon EBS volumes if you don't provide a KMS key ID. Make sure to replace the Region with the AWS Region of your KM key ID.

  ```
  aws ec2 modify-ebs-default-kms-key-id --region us-west-2 --kms-key-
  id AKIAIOSFODNN7EXAMPLE
  ```

  The above command will generate an output similar to the following output:

  ```
  {
    "KmsKeyId": "arn:aws:kms:us-west-2:444455556666:key/AKIAIOSFODNN7EXAMPLE"
  }
  ```

  For more information, see modify-ebs-default-kms-key-id.

Customizations in Malware Protection

This section describes how you can customize the scanning options for your Amazon EC2 instances or container workloads when a malware scan gets invoked, either initiated on-demand or through GuardDuty.

General settings

Snapshots retention

GuardDuty provides you with the option to retain the snapshots of your EBS volumes in your AWS account. By default, the snapshots retention setting is turned off. The snapshots will only be retained if you have this setting turned on before the scan initiates.

As the scan initiates, GuardDuty generates the replica EBS volumes based on the snapshots of your EBS volumes. After the scan completes and the snapshots retention setting in your account was turned on already, the snapshots of your EBS volumes will be retained only when malware is found and Malware Protection finding types get generated. Whether or not you have turned on the snapshots retention setting, when no malware is detected, GuardDuty automatically deletes the snapshots of your EBS volumes.

Snapshots usage cost

During the malware scanning, as GuardDuty creates the snapshots of your Amazon EBS volumes, there is a usage cost associated with this step. If you turn on the snapshots retention setting for your account, when malware is found and the snapshots get retained, you will incur usage cost for the same. For information on cost of snapshots and their retention, see Amazon EBS pricing.

Choose your preferred access method to turn on the snapshots retention setting.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Protection plans**, choose **Malware Protection**.
3. Choose **General settings** in the bottom section of the console. To retain the snapshots, turn on **Snapshots retention**.

API/CLI

1. Run UpdateMalwareScanSettings to update the current configuration for snapshot retention setting.
2. Alternatively, you can run the following AWS CLI command to automatically retain snapshots when GuardDuty Malware Protection generates findings.

   Ensure to replace the *detector-id* with your own valid `detectorId`.
3. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   ```
   aws guardduty update-malware-scan-settings --detector-
   id 60b8777933648562554d637e0e4bb3b2 --ebs-snapshot-preservation
    "RETENTION_WITH_FINDING"
   ```
4. If you want to turn off snapshots retention, replace `RETENTION_WITH_FINDING` with `NO_RETENTION`.

Scan options with user-defined tags

By using GuardDuty-initiated malware scan, you can also specify tags to either include or exclude Amazon EC2 instances and Amazon EBS volumes from the scanning and threat detection process. You can customize each GuardDuty-initiated malware scan by editing tags in either the inclusion or exclusion tags list. Each list can include up to 50 tags.

If you don't already have user-defined tags associated to your EC2 resources, see Tag your Amazon EC2 resources in the *Amazon EC2 User Guide for Linux Instances* or Tag your Amazon EC2 resources in the *Amazon EC2 User Guide for Windows Instances*.

Note

On-demand malware scan doesn't support scan options with user-defined tags. It supports Global `GuardDutyExcluded` tag.

To exclude EC2 instances from malware scan

If you want to exclude any Amazon EC2 instance or Amazon EBS volume during the scanning process, you can set the `GuardDutyExcluded` tag to `true` for any Amazon EC2 instance or Amazon EBS volume, and GuardDuty won't scan it. For more information about `GuardDutyExcluded` tag, see Service-linked role permissions for Malware Protection. You can also add an Amazon EC2 instance tag to an exclusion list. If you add multiple tags to the exclusion tags list, any Amazon EC2 instance that contains at least one of these tags will be excluded from the malware scanning process.

Choose your preferred access method to add a tag associated with an Amazon EC2 instance, to an exclusion list.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Protection plans**, choose **Malware Protection**.
3. Expand **Inclusion/Exclusion tags** section. Choose **Add tags**.
4. Choose **Exclusion tags** and then choose to **Confirm**.
5. Specify the tag's **`Key`** and **`Value`** pair that you want to exclude. It is optional to provide the **`Value`**. After you add all the tags, choose **Save**.

   > ⚠ **Important**
   >
   > Tag keys and values are case-sensitive. For more information, see Tag restrictions in the *Amazon EC2 User Guide for Linux Instances* or Tag restrictions in the *Amazon EC2 User Guide for Windows Instances*.

   If a value for a key is not provided and the EC2 instance is tagged with the specified key, this EC2 instance will be excluded from the GuardDuty-initiated malware scan scanning process, regardless of the tag's assigned value.

API/CLI

- Update the malware scan settings by excluding an EC2 instance or a container workload from the scanning process.

  The following AWS CLI example command adds a new tag to the exclusion tags list. Ensure to replace the example *`detector-id`* with your own valid `detectorId`.

  `MapEquals` is a list of `Key`/`Value` pairs.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-malware-scan-settings --detector-
id 60b8777933648562554d637e0e4bb3b2 --scan-resource-criteria '{"Exclude":
 {"EC2_INSTANCE_TAG" : {"MapEquals": [{ "Key": "TestKeyWithValue", "Value":
 "TestValue" }, {"Key":"TestKeyWithoutValue"} ]}}}' --ebs-snapshot-preservation
 "RETENTION_WITH_FINDING"
```

⚠ **Important**

Tag keys and values are case-sensitive. For more information, see Tag restrictions in the *Amazon EC2 User Guide for Linux Instances* or Tag restrictions in the *Amazon EC2 User Guide for Windows Instances*.

To include EC2 instances in malware scan

If you want to scan an EC2 instance, add its tag to the inclusion list. When you add a tag to an inclusion tags list, an EC2 instance that doesn't contain any of the added tags is skipped from the malware scan. If you add multiple tags to the inclusion tags list, an EC2 instance that contains at least one of those tags is included in the malware scan. Sometimes, an EC2 instance may be skipped during the scanning process. For more information, see Reasons for skipping resource during malware scan.

Choose your preferred access method to add a tag associated with an EC2 instance, to an inclusion list.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Protection plans**, choose **Malware Protection**.
3. Expand **Inclusion/Exclusion tags** section. Choose **Add tags**.
4. Choose **Inclusion tags** and then choose **Confirm**.
5. Choose **Add new inclusion tag** and specify the tag's **`Key`** and **`Value`** pair that you want to include. It is optional to provide the **`Value`**.

After you have added all the inclusion tags, choose **Save**.

If a value for a key is not provided an EC2 instance is tagged with the specified key, the EC2 instance will be included in the Malware Protection scanning process, regardless of the tag's assigned value.

API/CLI

- Update the malware scan settings to include an EC2 instance or a container workload in the scanning process.

  The following AWS CLI example command adds a new tag to the inclusion tags list. Ensure that you replace the example *detector-id* with your own valid detectorId. Replace the example *TestKey* and *TestValue* with the Key and Value pair of the tag associated with your EC2 resource.

  MapEquals is a list of Key/Value pairs.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-malware-scan-settings --detector-
  id 60b8777933648562554d637e0e4bb3b2 --scan-resource-criteria '{"Include":
   {"EC2_INSTANCE_TAG" : {"MapEquals": [{ "Key": "TestKeyWithValue", "Value":
   "TestValue" }, {"Key":"TestKeyWithoutValue"} ]}}}' --ebs-snapshot-preservation
   "RETENTION_WITH_FINDING"
  ```

  ⚠ **Important**

  Tag keys and values are case-sensitive. For more information, see Tag restrictions in the *Amazon EC2 User Guide for Linux Instances* or Tag restrictions in the *Amazon EC2 User Guide for Windows Instances*.

ⓘ **Note**

It may take up to 5 minutes for GuardDuty to detect a new tag.

At any time, you can either choose **Inclusion tags** or **Exclusion tags** but not both. If you want to switch between the tags, choose that tag from the dropdown menu when you add new tags, and **Confirm** your selection. This action clears all your current tags.

Global `GuardDutyExcluded` tag

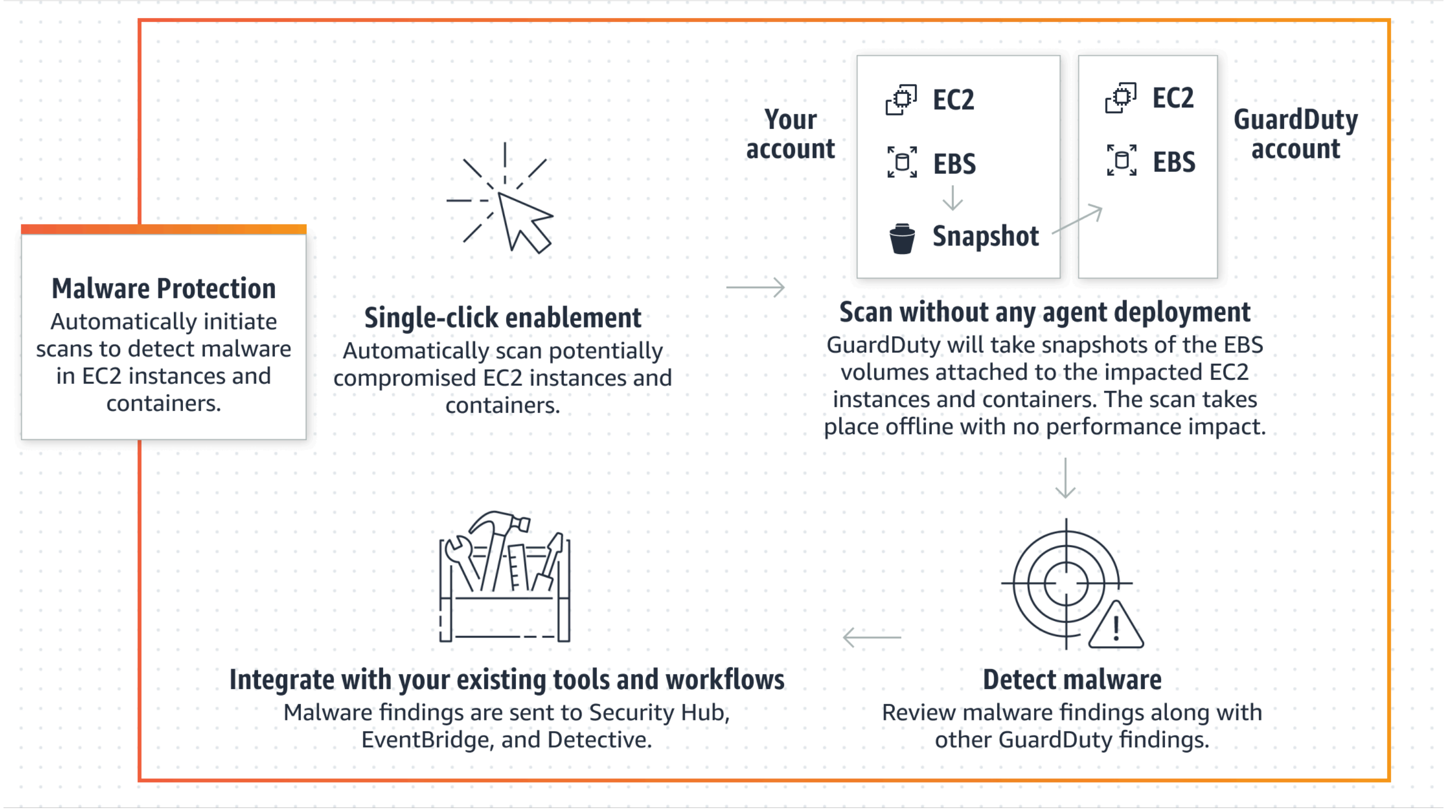
By default, the snapshots of your EBS volumes get created with a `GuardDutyScanId` tag. Do not remove this tag because doing so will prevent GuardDuty from accessing the snapshots. Both scan types in Malware Protection do not scan the Amazon EC2 instances or Amazon EBS volumes that have the `GuardDutyExcluded` tag set to `true`. If a Malware Protection scan on such a resource, a scan ID will be generated but the scan will be skipped with an `EXCLUDED_BY_SCAN_SETTINGS` reason. For more information, see Reasons for skipping resource during malware scan.

GuardDuty-initiated malware scan

With GuardDuty-initiated malware scan enabled, whenever GuardDuty detects malicious activity that indicates the potential presence of malware in your Amazon EC2 instance or container workload and GuardDuty generates Findings that invoke GuardDuty-initiated malware scan, GuardDuty automatically initiates an agentless scan on the Amazon Elastic Block Store (Amazon EBS) volumes attached to the potentially impacted Amazon EC2 instance or container workload to detect the presence of malware. With scan options, you can add inclusion tags associated with the resources that you want to scan or add exclusion tags associated with the resources that you want to skip from the scanning process. An automatic scan initiation will always consider your scan options. You can also choose to turn on the snapshots retention setting to retain the snapshots of your EBS volumes only if Malware Protection detects the presence of malware. For more information, see Customizations in Malware Protection.

For each Amazon EC2 instance and container workload for which GuardDuty generates findings, an automatic GuardDuty-initiated malware scan gets invoked once every 24 hours. For information about how the Amazon EBS volumes attached to your Amazon EC2 instance or container workload are scanned, see Feature in Malware Protection.

The following image describes how GuardDuty-initiated malware scan works.


Malware Protection
Automatically initiate scans to detect malware in EC2 instances and containers.
Single-click enablement
Automatically scan potentially compromised EC2 instances and containers.
Your account
EC2
EBS
Snapshot
EC2
EBS
GuardDuty account
Scan without any agent deployment
GuardDuty will take snapshots of the EBS volumes attached to the impacted EC2 instances and containers. The scan takes place offline with no performance impact.
Detect malware
Review malware findings along with other GuardDuty findings.
Integrate with your existing tools and workflows
Malware findings are sent to Security Hub, EventBridge, and Detective.

When malware is found, GuardDuty generates Malware Protection finding types. If GuardDuty doesn't generate a finding indicative of malware on the same resource, no GuardDuty-initiated malware scan will be invoked. You can also initiate an On-demand malware scan on the same resource. For more information, see On-demand malware scan.

How 30-day free trial period affects GuardDuty accounts

You can choose to turn on or turn off the GuardDuty-initiated malware scan functionality for any account or available AWS Regions, at any time.

- When you activate GuardDuty for the first time (new GuardDuty account), GuardDuty-initiated malware scan is already turned on and included in the 30-day free trial period.
- The existing GuardDuty accounts can turn on GuardDuty-initiated malware scan for the first time with a 30-day free trial period.
- If you've an existing GuardDuty account that has been using Malware Protection before On-demand malware scan was generally available and this GuardDuty account already uses the pricing model for its AWS Region, no action is needed to continue using GuardDuty-initiated malware scan.

Note

If you're on a 30-day free trial period, the usage cost for creating the Amazon EBS volume snapshots and their retention will still apply. For more information, see Amazon EBS pricing.

For information about enabling GuardDuty-initiated malware scan, see Configuring GuardDuty-initiated malware scan.

Configuring GuardDuty-initiated malware scan

Configuring GuardDuty-initiated malware scan for a standalone account

For accounts associated with AWS Organizations, you can automate this process through console settings, as described in the next section.

To enable or disable GuardDuty-initiated malware scan

Choose your preferred access method to configure GuardDuty-initiated malware scan for a standalone account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, under **Protection plans**, choose **Malware Protection**.
3. The Malware Protection pane lists the current status of GuardDuty-initiated malware scan for your account. You may enable or disable it at any time by selecting **Enable** or **Disable** respectively.
4. Choose **Save**.

API/CLI

- Run the updateDetector API operation using your own regional detector ID and passing the `dataSources` object with `EbsVolumes` set to `true` or `false`.

  You can also enable or disable GuardDuty-initiated malware scan 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 GuardDuty-initiated malware scan. To disable it, replace `true` with `false`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
 aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
features [{"Name" : "EBS_MALWARE_PROTECTION", "Status" : "ENABLED"}]'
```

Configuring GuardDuty-initiated malware scan in multiple-account environments

In a multi-account environment, only GuardDuty administrator account accounts can configure GuardDuty-initiated malware scan. GuardDuty administrator account accounts can enable or disable the use of GuardDuty-initiated malware scan for their member accounts. Once the administrator account configures GuardDuty-initiated malware scan for a member account, the member account will follow the administrator account account settings and be unable to modify these settings through the console. GuardDuty administrator account accounts that manage their member accounts with AWS Organizations support can choose to have GuardDuty-initiated malware scan enabled automatically on all the existing and new accounts in the organization. For more information, see Managing GuardDuty accounts with AWS Organizations.

Establishing trusted access to enable GuardDuty-initiated malware scan

If the GuardDuty delegated administrator account is not the same as the management account in your organization, the management account must enable GuardDuty-initiated malware scan for their organization. This way, the delegated administrator account can create the Service-linked role permissions for Malware Protection in member accounts that are managed through AWS Organizations.

> **Note**
>
> Before you designate a delegated GuardDuty administrator account, see Considerations and recommendations.

Choose your preferred access method to allow the delegated GuardDuty administrator account to enable GuardDuty-initiated malware scan for member accounts in the organization.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   To log in, use the management account for your AWS Organizations organization.

2. a. If you have not designated a delegated GuardDuty administrator account, then:

      On the **Settings** page, under **delegated GuardDuty administrator account**, enter the 12-digit `account ID` that you want to designate to administer the GuardDuty policy in your organization. Choose **Delegate**.

   b. i. If you've already designated a delegated GuardDuty administrator account that is different from the management account, then:

         On the **Settings** page, under **Delegated Administrator**, turn on the **Permissions** setting. This action will allow the delegated GuardDuty administrator account to attach relevant permissions to the member accounts and enable GuardDuty-initiated malware scan in these member accounts.

      ii. If you've already designated a delegated GuardDuty administrator account that is the same as the management account, then you can directly enable GuardDuty-initiated malware scan for the member accounts. For more information, see Auto-enable GuardDuty-initiated malware scan for all member accounts.

   > **Tip**
   >
   > If the delegated GuardDuty administrator account is different from your management account, you must provide permissions to the delegated GuardDuty administrator account to allow enabling GuardDuty-initiated malware scan for member accounts.

3. If you want to allow the delegated GuardDuty administrator account to enable GuardDuty-initiated malware scan for member accounts in other Regions, change your AWS Region, and repeat the steps above.

API/CLI

1. Using your management account credentials, run the following command:

   ```
   aws organizations enable-aws-service-access --service-principal malware-
   protection.guardduty.amazonaws.com
   ```

2. (Optional) to enable GuardDuty-initiated malware scan for the management account that is not a delegated administrator account, the management account will first create the Service-linked role permissions for Malware Protection explicitly in their account, and then enable GuardDuty-initiated malware scan from the delegated administrator account, similar to any other member account.

   ```
   aws iam create-service-linked-role --aws-service-name malware-
   protection.guardduty.amazonaws.com
   ```

3. You have designated the delegated GuardDuty administrator account in the currently selected AWS Region. If you have designated an account as a delegated GuardDuty administrator account in one region, that account must be your delegated GuardDuty administrator account in all other regions. Repeat the step above for all other Regions.

Configuring GuardDuty-initiated malware scan for delegated GuardDuty administrator account

Choose your preferred access method to enable or disable GuardDuty-initiated malware scan for a delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the management account credentials.

2. In the navigation pane, choose **Malware Protection**.
3. On the **Malware Protection** page, choose **Edit** next to **GuardDuty-initiated malware scan**.
4. Do one of the following:

Using Enable for all accounts

- Choose **Enable for all accounts**. This will enable the protection plan for all the active GuardDuty accounts in your AWS organization, including the new accounts that join the organization.
- Choose **Save**.

Using Configure accounts manually

- To enable the protection plan only for the delegated GuardDuty administrator account account, choose **Configure accounts manually**.
- Choose **Enable** under the **delegated GuardDuty administrator account (this account)** section.
- Choose **Save**.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `EBS_MALWARE_PROTECTION` and `status` as `ENABLED` or `DISABLED`.

You can enable or disable GuardDuty-initiated malware scan by running the following AWS CLI command. Make sure to use delegated GuardDuty administrator account's valid *`detector ID`*.

> **Note**
>
> The following example code enables GuardDuty-initiated malware scan. To disable it, replace `ENABLED` with `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 /
              --account-ids 555555555555 /
              --features '[{"Name": "EBS_MALWARE_PROTECTION", "Status": "ENABLED"}]'
```

Auto-enable GuardDuty-initiated malware scan for all member accounts

Choose your preferred access method to enable the GuardDuty-initiated malware scan feature for all member accounts. This includes existing member accounts and the new accounts that join the organization.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   Using the Malware Protection page

   1. In the navigation pane, choose **Malware Protection**.
   2. On the **Malware Protection** page, choose **Edit** in the **GuardDuty-initiated malware scan** section.
   3. Choose **Enable for all accounts**. This action automatically enables GuardDuty-initiated malware scan for both existing and new accounts in the organization.
   4. Choose **Save**.

   > **Note**
   >
   > It may take up to 24 hours to update the configuration for the member accounts.

   Using the Accounts page

   1. In the navigation pane, choose **Accounts**.
   2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
   3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **GuardDuty-initiated malware scan**.
   4. On the **Malware Protection** page, choose **Edit** in the **GuardDuty-initiated malware scan** section.

5. Choose **Enable for all accounts**. This action automatically enables GuardDuty-initiated malware scan for both existing and new accounts in the organization.
6. Choose **Save**.

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

Using the Accounts page

1. In the navigation pane, choose **Accounts**.
2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **GuardDuty-initiated malware scan**.
4. Choose **Save**.

If you can't use the **Enable for all accounts** option, see Selectively enable or disable GuardDuty-initiated malware scan for member accounts.

API/CLI

- To selectively enable or disable GuardDuty-initiated malware scan for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable GuardDuty-initiated malware scan for a single member account. To disable a member account, replace ENABLED with DISABLED.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"Name": "EBS_MALWARE_PROTECTION", "Status": "ENABLED"}]'
  ```

  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.

Enable GuardDuty-initiated malware scan for all existing active member accounts

Choose your preferred access method to enable GuardDuty-initiated malware scan for all the existing active member accounts in the organization.

To configure GuardDuty-initiated malware scan for all existing active member accounts

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **Malware Protection**.
3. On the **Malware Protection**, you can view the current status of the **GuardDuty-initiated malware scan** configuration. Under the **Active member accounts** section, choose **Actions**.
4. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
5. Choose **Save**.

Auto-enable GuardDuty-initiated malware scan for new member accounts

The newly added member accounts must **Enable** GuardDuty before selecting configuring GuardDuty-initiated malware scan. The member accounts managed by invitation can configure GuardDuty-initiated malware scan manually for their accounts. For more information, see Step 3 - Accept an invitation.

Choose your preferred access method to enable GuardDuty-initiated malware scan for new accounts that join your organization.

Console

> The delegated GuardDuty administrator account can enable GuardDuty-initiated malware scan for new member accounts in an organization, using either the **Malware Protection** or **Accounts** page.

To auto-enable GuardDuty-initiated malware scan for new member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   - Using the **Malware Protection** page:

     1. In the navigation pane, choose **Malware Protection**.
     2. On the **Malware Protection** page, choose **Edit** in the **GuardDuty-initiated malware scan**.
     3. Choose **Configure accounts manually**.
     4. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, GuardDuty-initiated malware scan will be automatically enabled for their account. Only the organization delegated GuardDuty administrator account can modify this configuration.
     5. Choose **Save**.

   - Using the **Accounts** page:

     1. In the navigation pane, choose **Accounts**.
     2. On the **Accounts** page, choose **Auto-enable** preferences.
     3. In the **Manage auto-enable preferences** window, select **Enable for new accounts** under **GuardDuty-initiated malware scan**.
     4. Choose **Save**.

API/CLI

- To enable or disable GuardDuty-initiated malware scan for new member accounts, invoke the UpdateOrganizationConfiguration API operation using your own *`detector ID`*.
- The following example shows how you can enable GuardDuty-initiated malware scan for a single member account. To disable it, see Selectively enable or disable GuardDuty-initiated malware scan for member accounts. If you don't want to enable it for all the new accounts joining the organization, set `AutoEnable` to `NONE`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --AutoEnable --features '[{"Name":
 "EBS_MALWARE_PROTECTION", "AutoEnable": NEW}]'
```

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.

Selectively enable or disable GuardDuty-initiated malware scan for member accounts

Choose your preferred access method to configure GuardDuty-initiated malware scan for member accounts selectively.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Accounts**.
3. On the **Accounts** page, review the **GuardDuty-initiated malware scan** column for the status of your member account.
4. Select the account for which you want to configure GuardDuty-initiated malware scan. You can select multiple accounts at a time.
5. From the **Edit protection plans** menu, choose the appropriate option for **GuardDuty-initiated malware scan**.

API/CLI

To selectively enable or disable GuardDuty-initiated malware scan for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.

The following example shows how you can enable GuardDuty-initiated malware scan for a single member account. To disable it, replace ENABLED with DISABLED.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 111122223333 --features '[{"Name": "EBS_MALWARE_PROTECTION",
 "Status": "ENABLED"}]'
```

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.

To selectively enable or disable GuardDuty-initiated malware scan for your member accounts, run the updateMemberDetectors API operation using your own *`detector ID`*. The following example shows how you can enable GuardDuty-initiated malware scan for a single member account. To disable it, replace `true` with `false`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 123456789012 --data-sources '{"MalwareProtection":
{"ScanEc2InstanceWithFindings":{"EbsVolumes":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.

Enable GuardDuty-initiated malware scan for existing accounts in the Organization managed via invitation

The GuardDuty Malware Protection service-linked role (SLR) must be created in member accounts. The administrator account can't enable the GuardDuty-initiated malware scan feature in member accounts that are not managed by AWS Organizations.

Presently, you can perform the following steps through the GuardDuty console at https://console.aws.amazon.com/guardduty/ to enable GuardDuty-initiated malware scan for the existing member accounts.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using your administrator account credentials.
2. In the navigation pane, choose **Accounts**.
3. Select the member account for which you want to enable GuardDuty-initiated malware scan. You can select multiple accounts at a time.
4. Choose **Actions**.
5. Choose **Disassociate member**.
6. In your member account, choose **Malware Protection** under **Protection plans** on the navigation pane.
7. Choose **Enable GuardDuty-initiated malware scan**. GuardDuty will create an SLR for the member account. For more information on SLR, see Service-linked role permissions for Malware Protection.
8. In your administrator account account, choose **Accounts** on the navigation pane.
9. Choose the member account that needs to be added back to the organization.
10. Choose **Actions** and then, choose **Add member**.

API/CLI

1. Use administrator account account to run DisassociateMembers API on the member accounts that want to enable GuardDuty-initiated malware scan.
2. Use your member account to invoke UpdateDetector to enable GuardDuty-initiated malware scan.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --data-sources '{"MalwareProtection":{"ScanEc2InstanceWithFindings":
{"EbsVolumes":true}}}'
```

3. Use administrator account account to run the CreateMembers API to add the member back to the organization.

Findings that invoke GuardDuty-initiated malware scan

A GuardDuty-initiated malware scan gets invoked when GuardDuty detects suspicious behavior indicative of malware on Amazon EC2 instance or container workloads.

- Backdoor:EC2/C&CActivity.B
- Backdoor:EC2/C&CActivity.B!DNS
- Backdoor:EC2/DenialOfService.Dns
- Backdoor:EC2/DenialOfService.Tcp
- Backdoor:EC2/DenialOfService.Udp
- Backdoor:EC2/DenialOfService.UdpOnTcpPorts
- Backdoor:EC2/DenialOfService.UnusualProtocol
- Backdoor:EC2/Spambot
- CryptoCurrency:EC2/BitcoinTool.B
- CryptoCurrency:EC2/BitcoinTool.B!DNS
- Impact:EC2/AbusedDomainRequest.Reputation
- Impact:EC2/BitcoinDomainRequest.Reputation
- Impact:EC2/MaliciousDomainRequest.Reputation
- Impact:EC2/PortSweep
- Impact:EC2/SuspiciousDomainRequest.Reputation
- Impact:EC2/WinRMBruteForce (Outbound only)
- Recon:EC2/Portscan
- Trojan:EC2/BlackholeTraffic

- Trojan:EC2/BlackholeTraffic!DNS
- Trojan:EC2/DGADomainRequest.B
- Trojan:EC2/DGADomainRequest.C!DNS
- Trojan:EC2/DNSDataExfiltration
- Trojan:EC2/DriveBySourceTraffic!DNS
- Trojan:EC2/DropPoint
- Trojan:EC2/DropPoint!DNS
- Trojan:EC2/PhishingDomainRequest!DNS
- UnauthorizedAccess:EC2/RDPBruteForce (Outbound only)
- UnauthorizedAccess:EC2/SSHBruteForce (Outbound only)
- UnauthorizedAccess:EC2/TorClient
- UnauthorizedAccess:EC2/TorRelay
- Backdoor:Runtime/C&CActivity.B
- Backdoor:Runtime/C&CActivity.B!DNS
- CryptoCurrency:Runtime/BitcoinTool.B
- CryptoCurrency:Runtime/BitcoinTool.B!DNS
- Execution:Runtime/NewBinaryExecuted
- Execution:Runtime/NewLibraryLoaded
- Execution:Runtime/ReverseShell
- Impact:Runtime/AbusedDomainRequest.Reputation
- Impact:Runtime/BitcoinDomainRequest.Reputation
- Impact:Runtime/CryptoMinerExecuted
- Impact:Runtime/MaliciousDomainRequest.Reputation
- Impact:Runtime/SuspiciousDomainRequest.Reputation
- PrivilegeEscalation:Runtime/CGroupsReleaseAgentModified
- PrivilegeEscalation:Runtime/ContainerMountsHostDirectory
- PrivilegeEscalation:Runtime/DockerSocketAccessed
- PrivilegeEscalation:Runtime/RuncContainerEscape
- PrivilegeEscalation:Runtime/UserfaultfdUsage

- Trojan:Runtime/BlackholeTraffic
- Trojan:Runtime/BlackholeTraffic!DNS
- Trojan:Runtime/DropPoint
- Trojan:Runtime/DropPoint!DNS
- Trojan:Runtime/DGADomainRequest.C!DNS
- Trojan:Runtime/DriveBySourceTraffic!DNS
- Trojan:Runtime/PhishingDomainRequest!DNS
- UnauthorizedAccess:Runtime/MetadataDNSRebind

On-demand malware scan

On-demand malware scan helps you detect the presence of malware on Amazon Elastic Block Store (Amazon EBS) volumes attached to your Amazon EC2 instances. With no configuration needed, you can initiate an on-demand malware scan by providing the Amazon Resource Name (ARN) of the Amazon EC2 instance that you want to scan. You can initiate an on-demand malware scan either through the GuardDuty console or API. Before initiating an on-demand malware scan, you can set your preferred Snapshots retention setting. The following scenarios can help you identify when to use the On-demand malware scan type with GuardDuty:

- You want to detect the presence of malware in your Amazon EC2 instances without enabling GuardDuty-initiated malware scan.
- You have enabled GuardDuty-initiated malware scan and a scan was invoked automatically. After following the recommended remediation for the generated Malware Protection finding type, if you want to initiate a scan on the same resource, you can initiate an on-demand malware scan after 1 hour has passed from the previous scan start time.

  On-demand malware scan doesn't require that 24 hours have passed from the time the previous malware scan was initiated. One hour should have passed before initiating an On-demand malware scan on the same resource. To avoid duplicating a malware scan on the same EC2 instance, see Re-scanning the same Amazon EC2 instance.

Note

On-demand malware scan is not included in the 30-day free trial period with GuardDuty. The usage cost applies to the total Amazon EBS volume scanned for each malware scan.

For more information, see Amazon GuardDuty pricing. For information about the cost of creating the Amazon EBS volume snapshots and their retention, see Amazon EBS pricing.

How On-demand malware scan works

With On-demand malware scan, you can initiate a malware scan request for your Amazon EC2 instance even when it is currently in use. After you initiate an On-demand malware scan, GuardDuty creates snapshots of the Amazon EBS volumes attached to the Amazon EC2 instance whose Amazon Resource Name (ARN) was provided for the scan. Next, GuardDuty shares these snapshots with the GuardDuty service account. GuardDuty creates encrypted replica EBS volumes from those snapshots in the GuardDuty service account. For more information about how the Amazon EBS volumes are scanned, see Elastic Block Storage (EBS) volume.

ⓘ **Note**

GuardDuty creates the snapshots of the data that has already been written to the Amazon EBS volumes at the point-in-time when you initiate an On-demand malware scan.

If malware is found and you've enabled the snapshots retention setting, the snapshots of your EBS volume are automatically retained in your AWS account. On-demand malware scan generates the Malware Protection finding types. If malware is not found, then regardless of the snapshots retention setting, the snapshots of your EBS volumes are deleted.

By default, the snapshots of your EBS volumes get created with a `GuardDutyScanId` tag. Do not remove this tag because doing so will prevent GuardDuty from accessing the snapshots. Both scan types in Malware Protection do not scan the Amazon EC2 instances or Amazon EBS volumes that have the `GuardDutyExcluded` tag set to `true`. If a Malware Protection scan on such a resource, a scan ID will be generated but the scan will be skipped with an `EXCLUDED_BY_SCAN_SETTINGS` reason. For more information, see Reasons for skipping resource during malware scan.

AWS Organizations service control policy – Denied access

Using the Service control policies (SCPs) in AWS Organizations, the delegated GuardDuty administrator account can restrict permissions and deny actions such as initiating an on-demand malware scan for Amazon EC2 instance owned by your accounts.

As a GuardDuty member account, when you initiate an on-demand malware scan for your Amazon EC2 instances, you may receive an error. You can connect with the management account to understand why an SCP was set up for your member account. For more information, see SCP effects on permissions.

Getting started with On-demand malware scan

As a GuardDuty administrator account, you can initiate an on-demand malware scan on behalf of your active member accounts that have the following prerequisites set up in their accounts. Standalone accounts and active member accounts in GuardDuty can also initiate an on-demand malware scan for their own Amazon EC2 instances.

Prerequisites

- GuardDuty must be enabled in the AWS Regions where you want to initiate the on-demand malware scan.
- Ensure that the AWS managed policy: AmazonGuardDutyFullAccess is attached to the IAM user or the IAM role. You will need the access key and secret key associated with the IAM user or the IAM role.
- As a delegated GuardDuty administrator account, you have the option to initiate an on-demand malware scan on behalf of an active member account.
- If you're a member account that doesn't have the Service-linked role permissions for Malware Protection, then initiating an on-demand malware scan for an Amazon EC2 instance that belongs to your account, will automatically create the SLR for Malware Protection.

⚠ **Important**

Ensure that no one deletes the SLR permissions for Malware Protection when the malware scan, whether GuardDuty-initiated or on-demand, is still in progress. Doing so will prevent the scan from completing successfully and providing definite scan result.

Before you initiate an on-demand malware scan, make sure that no scan was initiated on the same resource in the past 1 hour; otherwise, it will be de-duped. For more information, see Re-scanning the same resource.

Initiating On-demand malware scan

Choose your preferred access method to initiate an on-demand malware scan.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Initiate the scan using one of the following options:

   a. Using the **Malware Protection** page:

      i. In the navigation pane, under **Protection plans**, choose **Malware Protection**.

      ii. On the **Malware Protection** page, provide the **Amazon EC2 instance ARN**[1] for which you want to initiate the scan.

   b. Using the **Malware Scans** page:

      i. In the navigation pane, choose **Malware Scans**.

      ii. Choose **Start on-demand scan** and provide the **Amazon EC2 instance ARN**[1] for which you want to initiate the scan.

      iii. If this is a re-scan, select an **Amazon EC2** instance ID on the **Malware Scans** page.

      Expand the **Start on-demand scan** dropdown and choose **Re-scan selected instance**.
3. After you successfully initiate a scan using either method, a scan ID gets generated. You can use this scan ID to track the progress of the scan. For more information, see Monitoring malware scan statuses and results.

API/CLI

Invoke StartMalwareScan that accepts the `resourceArn` of the Amazon EC2 instance[1] for which you want to initiate an on-demand malware scan.

```
aws guardduty start-malware-scan --resource-arn "arn:aws:ec2:us-
east-1:555555555555:instance/i-b188560f"
```

After you successfully initiate a scan, `StartMalwareScan` returns a `scanId`. Invoke DescribeMalwareScans monitor the progress of the initiated scan.

[1] For information about the format of your Amazon EC2 instance ARN, see Amazon Resource Name (ARN). For Amazon EC2 instances, you can use the following example ARN format by replacing the values for the partition, Region, AWS account ID, and Amazon EC2 instance ID. For information about length of your instance ID, see Resource IDs.

```
arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f
```

Re-scanning the same Amazon EC2 instance

Whether a scan is GuardDuty-initiated or on-demand, you can initiate a new on-demand malware scan on the same EC2 instance after 1 hour from the start time of the previous malware scan. If the new malware scan gets initiated within 1 hour of initiation of the previous malware scan, your request will result in the following error, and no scan ID will get generated for this request.

```
A scan was initiated on this resource recently. You can request a scan on
the same resource one hour after the previous scan start time.
```

For information about how to initiate a new scan on the same resource, see Initiating On-demand malware scan.

To track the status of the malware scans, see Monitoring scan statuses and results in GuardDuty Malware Protection.

Monitoring scan statuses and results in GuardDuty Malware Protection

You can monitor the scan status of each GuardDuty Malware Protection scan. The possible values for scan **Status** are `Completed`, `Running`, `Skipped`, and `Failed`.

After the scan completes, the **Scan result** is populated for scans that have the **Status** as `Completed`. Possible values for **Scan result** are `Clean` and `Infected`. Using **Scan type**, you can identify if the malware scan was `GuardDuty initiated` or `On demand`.

Scan results for each malware scan has a retention period of 90 days. Choose your preferred access method to track the status of your malware scan.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

2. In the navigation pane, choose **Malware scans**.
3. You can filter the malware scans by the following **Properties** available in the **filter criteria**.

   - **Scan ID**
   - **Account ID**
   - **EC2 instance ARN**
   - **Scan type**
   - **Scan status**

   For information on properties used for filter criteria, see Finding details.

API/CLI

- After the malware scan has a scan result, you can filter the malware scans on the basis of `EC2_INSTANCE_ARN`, `SCAN_ID`, `ACCOUNT_ID`, `SCAN_TYPE` `GUARDDUTY_FINDING_ID`, `SCAN_STATUS`, and `SCAN_START_TIME`.

  The `GUARDDUTY_FINDING_ID` filter criteria is available when the `SCAN_TYPE` is GuardDuty initiated. For information about any filter criteria, see Finding details.
- You can change the example *`filter-criteria`* in the command below. Presently, you can filter on the basis of one `CriterionKey` at a time. The options for `CriterionKey` are `EC2_INSTANCE_ARN`, `SCAN_ID`, `ACCOUNT_ID`, `SCAN_TYPE` `GUARDDUTY_FINDING_ID`, `SCAN_STATUS`, and `SCAN_START_TIME`.

  If you use the same `CriterionKey` as below, ensure to replace the example `EqualsValue` with your own valid AWS *`scan-id`*.

  Replace the example `detector-id` with your own valid *`detector-id`*. You can change the *`max-results`* (up to 50) and the *`sort-criteria`*. The `AttributeName` is mandatory and must be `scanStartTime`.

  ```
  aws guardduty describe-malware-scans --detector-
  id 60b8777933648562554d637e0e4bb3b2 --max-results 1 --sort-criteria
   '{"AttributeName": "scanStartTime", "OrderBy": "DESC"}' --filter-criteria
   '{"FilterCriterion":[{"CriterionKey":"SCAN_ID", "FilterCondition":
  {"EqualsValue":"123456789012"}}] }'
  ```

- The response of this command displays a maximum of one result with details about the affected resource and malware findings (if `Infected`).

GuardDuty service accounts by AWS Region

When a snapshot gets created and shared with a GuardDuty service account, a new event gets created in your CloudTrail logs. This event specifies the corresponding `snapshotId` and `userId` (GuardDuty service account for that AWS Region). For more information, see Feature in Malware Protection.

The following example is a snippet from a CloudTrail event that shows the request body for the `ModifySnapshotAttribute` request:

```
"requestParameters": {
        "snapshotId": "snap-1234567890abcdef0",
        "createVolumePermission": {
            "add": {
                "items": [
                    {
                        "userId": "111122223333"
                    }
                ]
            }
        },
        "attributeType": "CREATE_VOLUME_PERMISSION"
    }
```

The following table shows the GuardDuty service accounts for each Region. The `userId` is the GuardDuty service account and depends on the selected Region.

AWS Region	Region code	GuardDuty service account ID (`userId`)
US East (N. Virginia)	us-east-1	652050842985
US East (Ohio)	us-east-2	178123968615
US West (N. California)	us-west-1	669213148797
US West (Oregon)	us-west-2	447226417196

AWS Region	Region code	GuardDuty service account ID (`userId`)
Asia Pacific (Mumbai)	ap-south-1	913179291432
Asia Pacific (Osaka)	ap-northeast-3	089661699081
Asia Pacific (Seoul)	ap-northeast-2	039163547507
Asia Pacific (Tokyo)	ap-northeast-1	874749492622
Asia Pacific (Singapore)	ap-southeast-1	247460962669
Asia Pacific (Sydney)	ap-southeast-2	124839743349
Canada (Central)	ca-central-1	175877067165
Canada West (Calgary)	ca-west-1	894794104037
Europe (Frankfurt)	eu-central-1	002294850712
Europe (Ireland)	eu-west-1	283769539786
Europe (London)	eu-west-2	310125036783
Europe (Paris)	eu-west-3	866607715269
Europe (Stockholm)	eu-north-1	693780578038
China (Beijing)	cn-north-1	448721096076
China (Ningxia)	cn-northwest-1	480864352451
South America (São Paulo)	sa-east-1	546914126324
Asia Pacific (Hyderabad) (Opt-in)	ap-south-2	682251015962
Asia Pacific (Melbourne) (Opt-in)	ap-southeast-4	353488359550

AWS Region	Region code	GuardDuty service account ID (`userId`)
Europe (Spain) (Opt-in)	eu-south-2	936182149045
Europe (Zurich) (Opt-in)	eu-central-2	867642063380
Israel (Tel Aviv) (Opt-in)	il-central-1	619233833001
Europe (Milan) (Opt-in)	eu-south-1	977238331021
Asia Pacific (Hong Kong) (Opt-in)	ap-east-1	249472122084
Middle East (Bahrain) (Opt-in)	me-south-1	404001805210
Africa (Cape Town) (Opt-in)	af-south-1	957664736811
Asia Pacific (Jakarta) (Opt-in)	ap-southeast-3	452118225523
Middle East (UAE) (Opt-in)	me-central-1	828603743433

Malware Protection quotas

Malware Protection has the following default availability of varied resources that the feature uses.

Scope	Default	Comments
Extraction and analysis of data in compressed or archived file	5	The maximum number of nested levels allowed in an archived file.
Number of files within an archived file	1000	The maximum number of files that can be scanned within an archive. This count

Scope	Default	Comments
		is the sum of the number of files extracted from the archive and the number of files extracted from all the nested archives.
Number of threats	32	The maximum number of threats that you can view in the findings panel. GuardDuty Malware Protection may have detected more threat names. If the number of detected threat names is higher than the default value, you can view the JSON details by selecting the **Finding ID** under the finding name in the details panel of the GuardDuty console.
Number of files per detected threat	5	The maximum number of files identified per detected threat. For example, if GuardDuty detects 10 files associate d with a single threat, the threat will display a maximum of 5 files.

Scope	Default	Comments
EBS volumes per scan per instance	11	The maximum number of EBS volumes that GuardDuty can scan per EC2 instance. If there are more than 11 EBS volumes that need to be scanned, GuardDuty Malware Protection sorts the `deviceName` alphabetically, and selects the first 11 EBS volumes.
EBS volume size	1024 GB	The maximum EBS volume size in GB that GuardDuty Malware Protection can scan in each Region.
Supported file system types	GuardDuty Malware Protection can scan the following file system types: • New Technology File System (NTFS) • X File System (XFS) • Second extended (ext2) File System • Fourth extended (ext4) File System • File Allocation Table (FAT) File System • Virtual File Allocation Table (VFAT) File System	N/A.

Scope	Default	Comments
Scan options tags	50	The maximum number of resource tags that you can add to customize your malware scan options setting. For more information, see Scan options with user-defined tags.
Finding retention period	90	The maximum number of days that GuardDuty retains a finding. For the latest information, see Quotas for Amazon GuardDuty.
Malware scan retention period	90	The maximum number of days that GuardDuty Malware Protection retains the history of a scan. For more information on viewing recent malware scans, see Monitoring scan statuses and results in GuardDuty Malware Protection.
Transactions per second (TPS) for On-demand malware scan	1	The number of On-demand malware scan requests that can be initiated per second in each Region.
Burst limit for On-demand malware scan	1	The number of concurrent malware On-demand malware scan requests that can be initiated per second in each Region.

GuardDuty RDS Protection

RDS Protection in Amazon GuardDuty analyzes and profiles RDS login activity for potential access threats to your Amazon Aurora databases (Amazon Aurora MySQL-Compatible Edition and Aurora PostgreSQL-Compatible Edition). This feature allows you to identify potentially suspicious login behavior. RDS Protection doesn't require additional infrastructure; it is designed so as not to affect the performance of your database instances.

When RDS Protection detects a potentially suspicious or anomalous login attempt that indicates a threat to your database, GuardDuty generates a new finding with details about the potentially compromised database.

You can enable or disable the RDS Protection feature for any account in any AWS Region where this feature is available within Amazon GuardDuty, at any time. An existing GuardDuty account can enable RDS Protection with a 30-day trial period. For a new GuardDuty account, RDS Protection is already enabled and included in the 30-day free trial period. For more information, see Estimating cost.

> **Note**
>
> When the RDS Protection feature is not enabled, GuardDuty neither collects your RDS login activity, nor detects anomalous or suspicious login behavior.

For information about the AWS Regions where GuardDuty doesn't yet support RDS Protection, see Region-specific feature availability.

Supported Amazon Aurora databases

The following table shows the supported Aurora database versions.

Amazon Aurora DB engine	Supported engine versions
Aurora MySQL	• 2.10.2 or later • 3.02.1 or later
Aurora PostgreSQL	• 10.17 or later

Amazon Aurora DB engine	Supported engine versions
	• 11.12 or later • 12.7 or later • 13.3 or later • 14.3 or later • 15.2 or later • 16.1 or later

How RDS Protection uses RDS login activity monitoring

RDS Protection in Amazon GuardDuty helps you protect the supported Amazon Aurora (Aurora) databases in your account. After you enable the RDS Protection feature, GuardDuty immediately starts monitoring RDS login activity from Aurora databases in your account. GuardDuty continuously monitors and profiles RDS login activity for suspicious activity, for example, unauthorized access to Aurora database in your account, from a previously unseen external actor. When you enable RDS Protection for the first time or you have a newly created database instance, a learning period is required to baseline normal behavior. For this reason, newly enabled or newly created database instances may not have an associated anomalous login finding for up to two weeks of time. For more information, see RDS login activity monitoring.

When RDS Protection detects a potential threat, such as an unusual pattern in a series of successful, failed, or incomplete login attempts, GuardDuty generates a new finding with details about the potentially compromised database instance. For more information, see RDS Protection finding types. If you disable RDS Protection, GuardDuty immediately stops monitoring RDS login activity and is unable to detect any potential threat to your supported database instances.

> **Note**
>
> GuardDuty doesn't manage your Supported databases or RDS login activity, or make RDS login activity available to you.

Configuring RDS Protection for a standalone account

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **RDS Protection**.
3. The **RDS Protection** page shows the current status for your account. You may enable or disable the feature at any time by selecting **Enable** or **Disable**. **Confirm** your selection.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `RDS_LOGIN_EVENTS` and `status` as `ENABLED` or `DISABLED`.

You can also enable or disable RDS Protection by running the following AWS CLI command. Make sure to use your own valid *`detector ID`*.

> **Note**
>
> The following example code enables RDS Protection. To disable it, replace `ENABLED` with `DISABLED`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
features '[{"Name" : "RDS_LOGIN_EVENTS", "Status" : "ENABLED"}]'
```

Configuring RDS Protection in multiple-account environments

In a multiple-account environment, only the delegated GuardDuty administrator account has the option to enable or disable the RDS Protection feature for the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account manages their member accounts using AWS Organizations. This delegated GuardDuty administrator account can choose to auto-enable

RDS login activity monitoring for all the new accounts as they join the organization. For more information about multiple-account environments, see Managing multiple accounts in Amazon GuardDuty.

Configuring RDS Protection for delegated GuardDuty administrator account

Choose your preferred access method to configure RDS Login Activity Monitoring for the delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the management account credentials.
2. In the navigation pane, choose **RDS Protection**.
3. On the **RDS Protection** page, choose **Edit**.
4. Do one of the following:

   Using Enable for all accounts

   - Choose **Enable for all accounts**. This will enable the protection plan for all the active GuardDuty accounts in your AWS organization, including the new accounts that join the organization.
   - Choose **Save**.

   Using Configure accounts manually

   - To enable the protection plan only for the delegated GuardDuty administrator account account, choose **Configure accounts manually**.
   - Choose **Enable** under the **delegated GuardDuty administrator account (this account)** section.
   - Choose **Save**.

API/CLI

Run the updateDetector API operation using your own regional detector ID and passing the `features` object name as `RDS_LOGIN_EVENTS` and `status` as `ENABLED` or `DISABLED`.

You can enable or disable RDS Protection by running the following AWS CLI command. Make sure to use delegated GuardDuty administrator account's valid *detector ID*.

> **Note**
>
> The following example code enables RDS Protection. To disable it, replace ENABLED with DISABLED.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 555555555555 --features '[{"Name": "RDS_LOGIN_EVENTS", "Status":
 "ENABLED"}]'
```

Auto-enable RDS Protection for all member accounts

Choose your preferred access method to enable the RDS Protection feature for all member accounts. This includes existing member accounts and the new accounts that join the organization.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.
2. Do one of the following:

   Using the RDS Protection page

   1. In the navigation pane, choose **RDS Protection**.
   2. Choose **Enable for all accounts**. This action automatically enables RDS Protection for both existing and new accounts in the organization.
   3. Choose **Save**.

   > **Note**
   >
   > It may take up to 24 hours to update the configuration for the member accounts.

Using the Accounts page

1. In the navigation pane, choose **Accounts**.
2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **RDS Login Activity Monitoring**.
4. Choose **Save**.

If you can't use the **Enable for all accounts** option, see Selectively enable or disable RDS Protection for member accounts.

API/CLI

- To selectively enable or disable RDS Protection for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable RDS Protection for a single member account. To disable it, replace `ENABLED` with `DISABLED`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"name": "RDS_LOGIN_EVENTS", "status": "ENABLED"}]'
  ```

  > **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.

Enable RDS Protection for all existing active member accounts

Choose your preferred access method to enable RDS Protection for all the existing active member accounts in your organization.

Console

To configure RDS Protection for all existing active member accounts

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **RDS Protection**.
3. On the **RDS Protection** page, you can view the current status of the configuration. Under the **Active member accounts** section, choose **Actions**.
4. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
5. Choose **Confirm**.

API/CLI

- To selectively enable or disable RDS Protection for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.
- The following example shows how you can enable RDS Protection for a single member account. To disable it, replace ENABLED with DISABLED.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-member-detectors --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
   '[{"name": "RDS_LOGIN_EVENTS", "status": "ENABLED"}]'
  ```

  > **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.

Auto-enable RDS Protection for new member accounts

Choose your preferred access method to enable RDS login activity for new accounts that join your organization.

Console

The delegated GuardDuty administrator account can enable for new member accounts in an organization through the console, using either the **RDS Protection** or **Accounts** page.

To auto-enable RDS Protection for new member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.
2. Do one of the following:
   - Using the **RDS Protection** page:
     1. In the navigation pane, choose **RDS Protection**.
     2. On the **RDS Protection** page, choose **Edit**.
     3. Choose **Configure accounts manually**.
     4. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, RDS Protection will be automatically enabled for their account. Only the organization delegated GuardDuty administrator account can modify this configuration.
     5. Choose **Save**.
   - Using the **Accounts** page:
     1. In the navigation pane, choose **Accounts**.
     2. On the **Accounts** page, choose **Auto-enable** preferences.
     3. In the **Manage auto-enable preferences** window, select **Enable for new accounts** under **RDS Login Activity Monitoring**.
     4. Choose **Save**.

API/CLI

- To selectively enable or disable RDS Protection for your member accounts, invoke the UpdateOrganizationConfiguration API operation using your own *detector ID*.
- The following example shows how you can enable RDS Protection for a single member account. To disable it, see Selectively enable or disable RDS Protection for member accounts. If you don't want to enable it for all the new accounts joining the organization, set `autoEnable` to `NONE`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-organization-configuration --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable --features '[{"Name":
   "RDS_LOGIN_EVENTS", "AutoEnable": "NEW"}]'
  ```

  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.

Selectively enable or disable RDS Protection for member accounts

Choose your preferred access method to selectively enable or disable monitoring RDS login activity for member accounts.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **Accounts**.

   On the **Accounts** page, review the **RDS login activity** column for the status of your member account.

3. **To selectively enable or disable RDS login activity**

   Select the account for which you want to configure RDS Protection. You can select multiple accounts at a time. In the **Edit Protection Plans** dropdown menu, choose **RDS Login Activity**, and then choose the appropriate option.

API/CLI

To selectively enable or disable RDS Protection for your member accounts, invoke the updateMemberDetectors API operation using your own *detector ID*.

The following example shows how you can enable RDS Protection for a single member account. To disable it, replace ENABLED with DISABLED.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 111122223333 --features '[{"Name": "RDS_LOGIN_EVENTS", "Status":
 "ENABLED"}]'
```

> **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.

Feature in RDS Protection

RDS login activity monitoring

RDS login activity captures both successful and failed login attempts made to the Supported Amazon Aurora databases in your AWS environment. To help you protect your databases, GuardDuty RDS Protection continuously monitors the login activity for potentially suspicious login attempts. For example, an adversary may attempt to brute-force access to an Amazon Aurora database by guessing the database's password.

When you enable the RDS Protection feature, GuardDuty automatically starts to monitor RDS login activity for your databases directly from the Aurora service. If there is an indication of anomalous login behavior, GuardDuty generates a finding with details about the potentially compromised database. When you enable RDS Protection for the first time or you have a newly created database instance, a learning period is required to baseline normal behavior. For this reason, newly enabled or newly created database instances may not have an associated anomalous login finding for up to two weeks of time.

The RDS Protection feature does not require any additional setup; it does not affect any of your existing Amazon Aurora database configurations. GuardDuty doesn't manage your supported databases or RDS login activity, or make the RDS login activity available to you.

If you choose to auto-enable the RDS Protection feature for new member accounts as they join your organization, this action automatically enables GuardDuty for those new member accounts. For more information about configuring RDS login activity monitoring as a feature, see GuardDuty RDS Protection.

GuardDuty Runtime Monitoring

Runtime Monitoring observes and analyzes operating system-level, networking, and file events to help you detect potential threats in specific AWS workloads in your environment.

GuardDuty initially released Runtime Monitoring to support only Amazon Elastic Kubernetes Service (Amazon EKS) resources. However, now you can also use the Runtime Monitoring feature to provide threat detection for your AWS Fargate Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Compute Cloud (Amazon EC2) resources.

In this document and other sections related to Runtime Monitoring, GuardDuty uses the terminology of **resource type** to refer to Amazon EKS, Fargate Amazon ECS, and Amazon EC2 resources.

Runtime Monitoring uses a GuardDuty security agent that adds visibility into runtime behavior, such as file access, process execution, command line arguments, and network connections. For each resource type that you want to monitor for potential threats, you can manage the security agent for that specific resource type either automatically or manually (with an exception to Fargate (Amazon ECS only)). Managing the security agent automatically means that you permit GuardDuty to install and update the security agent on your behalf. On the other hand, when you manage the security agent for your resources manually, you are responsible for installing and updating the security agent, as needed.

With this extended capability, GuardDuty can help you identify and respond to potential threats that may target applications and data running in your individual workloads and instances. For example, a threat can potentially start by compromising a single container that runs a vulnerable web application. This web application might have access permissions to the underlying containers and workloads. In this scenario, incorrectly configured credentials could potentially lead to a broader access to the account, and the data stored within it.

By analyzing the runtime events of the individual containers and workloads, GuardDuty can potentially identify compromise of a container and associated AWS credentials in an initial phase, and detect attempts to escalate privileges, suspicious API requests, and malicious access to the data in your environment.

Contents

- Key concepts - Approaches to manage GuardDuty security agent
- Enabling GuardDuty Runtime Monitoring
- Configuring EKS Runtime Monitoring (API only)
- Migrating from EKS Runtime Monitoring to Runtime Monitoring
- Assessing runtime coverage for your resources
- Setting up CPU and memory monitoring
- Collected runtime event types that GuardDuty uses
- Amazon ECR repository hosting GuardDuty agent
- GuardDuty agent release history

How it works

To use Runtime Monitoring, you must enable Runtime Monitoring and then manage the GuardDuty security agent. The following list explains this two-step process:

1. **Enable Runtime Monitoring** for your account so that GuardDuty can accept the runtime events that it receives from your Amazon EC2 instances, Amazon ECS clusters, and Amazon EKS workloads.
2. **Manage GuardDuty agent** for the individual resources for which you want to monitor the runtime behavior. Based on the resource type, you can choose to deploy the GuardDuty security agent either manually or by allowing GuardDuty to manage it on your behalf, called automated agent configuration.

   GuardDuty uses Instance identity roles that authenticates the security agent for each resource type to send the associated runtime events to the VPC endpoint.

> **Note**
>
> GuardDuty doesn't make the runtime events accessible to you.

When you manage the security agent (either manually or through GuardDuty) in EKS Runtime Monitoring or Runtime Monitoring for EC2 instances, and GuardDuty is presently deployed on an Amazon EC2 instance and receives the Collected runtime event types from this instance, GuardDuty will not charge your AWS account for the analysis of VPC flow logs from this Amazon EC2 instance. This helps GuardDuty avoid double usage cost in the account.

The following topics explain how enabling Runtime Monitoring and managing GuardDuty security agent works differently for each resource type.

Contents

How Runtime Monitoring works with Amazon EC2 instances

Your Amazon EC2 instances can run multiple types of applications and workloads in your AWS environment. When you enable Runtime Monitoring and manage the GuardDuty security agent, GuardDuty helps you detect threats in your existing Amazon EC2 instances and potentially new ones. This feature also supports Amazon ECS managed Amazon EC2 instances.

Enabling Runtime Monitoring makes GuardDuty ready to consume runtime events from currently running and new processes within Amazon EC2 instances. GuardDuty requires a security agent to send runtime events from your EC2 instance to GuardDuty.

For Amazon EC2 instances, GuardDuty security agent operates at the instance level. You can decide if you want to monitor all or selective Amazon EC2 instances in your account. If you want to manage selective instances, the security agent is required only for these instances.

GuardDuty can also consume runtime events from new tasks and existing tasks running in Amazon EC2 instances within Amazon ECS clusters.

To install the GuardDuty security agent, Runtime Monitoring provides the following two options:

- Use automated agent configuration (recommended), or
- Manage security agent manually

Use automated agent configuration through GuardDuty (recommended)

Use automated agent configuration that permits GuardDuty to install the security agent on your Amazon EC2 instances on your behalf. GuardDuty also manages the updates to the security agent.

By default, GuardDuty installs the security agent on all the instances in your account. If you'd want GuardDuty to install and manage the security agent for selected EC2 instances only, add inclusion or exclusion tags to your EC2 instances, as needed.

Sometimes, you may not want to monitor runtime events for all the Amazon EC2 instances that belong to your account. For cases when you want to monitor the runtime events for a limited number of instances, add an inclusion tag as `GuardDutyManaged`:`true` to these selected instances. Starting with the availability of automated agent configuration for Amazon EC2, if your EC2 instance has an inclusion tag (`GuardDutyManaged`:`true`), GuardDuty will honor the tag and manage the security agent for the selected instances even when you do not explicitly enable automated agent configuration.

On the other hand, if there are a limited number of EC2 instances for which you don't want to monitor runtime events, add an exclusion tag (`GuardDutyManaged`:`false`) to these selected instances. GuardDuty will honor the exclusion tag by **neither** installing **nor** managing the security agent for these EC2 resources.

Impact

When you use automated agent configuration in an AWS account or an organization, you permit GuardDuty to take the following steps on your behalf:

- GuardDuty creates one SSM association for all your Amazon EC2 instances that are SSM managed and appear under **Fleet Manager** in the https://console.aws.amazon.com/systems-manager/ console.
- Using inclusion tags with automated agent configuration disabled – After enabling Runtime Monitoring, when you don't enable automated agent configuration but add inclusion tag to your Amazon EC2 instance, it means that you are permitting GuardDuty to manage the security agent on your behalf. SSM association will then install the security agent in each instance that has the inclusion tag (`GuardDutyManaged`:`true`).
- If you enable automated agent configuration – The SSM association will then install the security agent in all the EC2 instances that belong to your account.
- Using exclusion tags with automated agent configuration – Before you enable automated agent configuration, when you add exclusion tag to your Amazon EC2 instance, it means that you are

permitting GuardDuty to prevent installing and managing the security agent for this selected instance.

Now, when you enable automated agent configuration, the SSM association will install and manage the security agent in all the EC2 instances except the ones that are tagged with the exclusion tag.

- GuardDuty creates VPC endpoints in all the VPCs, including shared VPCs, as long as there is at least one Linux EC2 instance in that VPC that are not in the terminated or shutting-down instance states. For information about different instance states, see Instance lifecycle in the *Amazon EC2 User Guide for Linux Instances*.

  GuardDuty also supports Using shared VPC with automated security agents. When all the prerequisites are considered for your organization and AWS account, GuardDuty will use the shared VPC to receive runtime events.

> **Note**
>
> There is no additional cost for the usage of the VPC endpoints that GuardDuty creates.

Manage security agent manually

There are two ways to manage the security agent for Amazon EC2 manually:

- Use GuardDuty managed documents in AWS Systems Manager to install the security agent on your Amazon EC2 instances that are already SSM managed.

  Whenever you launch a new Amazon EC2 instance, ensure that it is SSM enabled.
- Use RPM package manager (RPM) scripts to install the security agent on your Amazon EC2 instances, whether or not they are SSM managed.

Next step

To get started with Runtime Monitoring configuration to monitor your Amazon EC2 instances, see Prerequisites for Amazon EC2 instance support.

How Runtime Monitoring works with Fargate (Amazon ECS only)

When you enable Runtime Monitoring, GuardDuty becomes ready to consume the runtime events from a task. These tasks run within the Amazon ECS clusters, which in turn run on the AWS Fargate (Fargate) instances. For GuardDuty to receive these runtime events, you must use the fully-managed, dedicated security agent.

Presently, Runtime Monitoring doesn't support the tasks launched by AWS Batch and AWS CodePipeline.

Presently, Runtime Monitoring supports managing the security agent for your Amazon ECS clusters (AWS Fargate) only through GuardDuty. There is no support for managing the security agent manually on Amazon ECS clusters.

You can allow GuardDuty to manage the GuardDuty security agent on your behalf, by using Automated agent configuration for an AWS account or an organization. GuardDuty will start deploying the security agent to the new Fargate tasks that are launched in your Amazon ECS clusters. The following list specifies what to expect when you enable the GuardDuty security agent.

Impact of enabling GuardDuty security agent

GuardDuty creates a virtual private cloud (VPC) endpoint

When you deploy the GuardDuty security agent, GuardDuty will create a VPC endpoint through which the security agent delivers the runtime events to GuardDuty.

> **Note**
>
> There is no additional cost for the usage of the VPC endpoints that GuardDuty creates.

GuardDuty adds a sidecar container

For a new Fargate task or service that starts running, a GuardDuty container (sidecar) attaches itself to each container within the Amazon ECS Fargate task. The GuardDuty security agent runs within the attached GuardDuty container. This helps GuardDuty to collect the runtime events of each container running within these tasks.

When you start a Fargate task, should the GuardDuty container (sidecar) be unable to launch in a healthy state, Runtime Monitoring is designed to not prevent the tasks from running.

By default, a Fargate task is immutable. GuardDuty will not deploy the sidecar when a task is already in a running state. If you want to monitor a container in an already running task, you can stop the task and start it again.

How Runtime Monitoring works with Amazon EKS clusters

Runtime Monitoring uses an EKS add-on `aws-guardduty-agent`, also called as GuardDuty security agent. After GuardDuty security agent gets deployed on your EKS clusters, GuardDuty is able to receive runtime events for these EKS clusters.

You can monitor the runtime events of your Amazon EKS clusters at either account or cluster level. You can manage the GuardDuty security agent for only those Amazon EKS clusters that you want to monitor for threat detection. You can manage the GuardDuty security agent either manually or by allowing GuardDuty to manage it on your behalf, by using Automated agent configuration.

When you use the automated agent configuration approach to allow GuardDuty to manage the deployment of the security agent on your behalf, it will automatically **create an Amazon Virtual Private Cloud (Amazon VPC) endpoint**. The security agent delivers the runtime events to GuardDuty by using this Amazon VPC endpoint.

> **Note**
>
> There is no additional cost for the usage of the VPC endpoints that GuardDuty creates.

Presently, GuardDuty supports Amazon EKS clusters running on Amazon EC2 instances. GuardDuty doesn't support Amazon EKS clusters running on AWS Fargate.

After Runtime Monitoring configuration

Assess runtime coverage

After you enable Runtime Monitoring and deploy the GuardDuty security agent, we recommend you to continuously[1] assess the coverage status of the resource where you have deployed the security agent. The coverage status could be either **Healthy** or **Unhealthy**. A **Healthy** coverage status indicates that GuardDuty is receiving the runtime events from the corresponding resource when there is an operating system-level activity.

When the coverage status becomes **Healthy** for the resource, GuardDuty is able to receive the runtime events and analyze them for threat detection. When GuardDuty detects a potential security threat in the tasks or applications running in your container workloads and instances, GuardDuty generates one or more Runtime Monitoring finding types.

[1] You can also configure an Amazon EventBridge (EventBridge) to receive a notification when the coverage status changes from **Unhealthy** to **Healthy** and otherwise.

For more information, see Assessing runtime coverage for your resources.

GuardDuty detects potential threats

As GuardDuty starts to receive the runtime events for your resource, it starts analyzing those events. When GuardDuty detects a potential security threat in any of your Amazon EC2 instances, Amazon ECS clusters, or Amazon EKS clusters, it generates one or more Runtime Monitoring finding types. You can access the finding details to view the impacted resource details.

How does 30-day free trial work in Runtime Monitoring

The 30-day free trial period works differently for the new GuardDuty accounts and the existing accounts that have already enabled EKS Runtime Monitoring prior to when Runtime Monitoring capability extended to Amazon EC2 instances and AWS Fargate (Amazon ECS only).

I am using GuardDuty trial period or I have never enabled EKS Runtime Monitoring

The following list explains how the 30-day free trial period works if you're either using the GuardDuty 30-day trial period or have never enabled EKS Runtime Monitoring:

- When you enable GuardDuty for the first time, Runtime Monitoring and EKS Runtime Monitoring will not be enabled by default.

  When you enable Runtime Monitoring for your account or organization, make sure to also configure the GuardDuty security agent for the resource that you want to monitor for threat detection. For example, if you want to use Runtime Monitoring for your Amazon EC2 instances, then after you enable Runtime Monitoring, you must also configure the security agent for Amazon EC2. You can choose to do this either manually or automatically through GuardDuty.

- The Runtime Monitoring protection plan is enabled at the **account level**. The 30-day free trial period works at the **resource level**. After the GuardDuty security agent gets deployed to a specific resource type, the 30-day free trial starts when GuardDuty receives its **first runtime event** associated with this resource type. For example, you have deployed the GuardDuty agent at the resource level (for Amazon EC2 instance, Amazon ECS cluster, and Amazon EKS cluster). When GuardDuty receives the first runtime event for an Amazon EC2 instance, the 30-day free trial will start for Amazon EC2 only.
- When you want to enable only EKS Runtime Monitoring – When you enable GuardDuty for the first time, EKS Runtime Monitoring is not enabled by default (after the release of Runtime Monitoring). You will need to enable EKS Runtime Monitoring. To use it optimally, make sure that you either manage the GuardDuty security agent manually or enable automated agent configuration so that GuardDuty manages the agent on your behalf. Your 30-day free trial period for EKS Runtime Monitoring starts when GuardDuty receives its first runtime event for the Amazon EKS resource.

I enabled EKS Runtime Monitoring prior to the launch of Runtime Monitoring

- For an existing GuardDuty account that has the EKS Runtime Monitoring protection plan enabled and uses the GuardDuty console experience to use this protection plan – With the announcement of Runtime Monitoring, the EKS Runtime Monitoring console experience has now been consolidated into Runtime Monitoring. Your existing configuration for EKS Runtime Monitoring remains the same. You can continue to use the API/CLI support to perform operations associated with EKS Runtime Monitoring.
- To use EKS Runtime Monitoring as a part of Runtime Monitoring, you will need to configure Runtime Monitoring for your account or organization. To keep the same configuration for Runtime Monitoring, see Migrating from EKS Runtime Monitoring to Runtime Monitoring. However, this will not impact your 30-day free trial for Amazon EKS resource.
- The Runtime Monitoring protection plan is enabled at the account level per Region. After the GuardDuty security agent gets deployed to one of the specified resource types (Amazon EC2 instance and Amazon ECS cluster), the 30-day free trial starts when GuardDuty receives the first runtime event associated with the resource. There is a 30-day free trial associated with each resource type.

  For example, after enabling Runtime Monitoring, you choose to deploy the GuardDuty agent only on Amazon EC2 instance, the 30-day free trial for this resource will start only when

GuardDuty receives its first runtime event for an Amazon EC2 instance. Later, when you deploy the GuardDuty agent for Fargate (Amazon ECS only), the 30-day free trial for this resource will start only when GuardDuty receives its first runtime event for Amazon ECS cluster. Considering you already have EKS Runtime Monitoring enabled for your account, GuardDuty doesn't reset the 30-day free trial for an Amazon EKS resource.

Key concepts - Approaches to manage GuardDuty security agent

Consider the key concepts that will help you manage the security agent on your Amazon EKS clusters and Amazon ECS clusters.

Contents

Fargate (Amazon ECS only) resource - Approaches to manage GuardDuty security agent

Runtime Monitoring provides you the option to detect potential security threats on either all of the Amazon ECS clusters (account level) or selective clusters (cluster level) in your account. When you enable Automated agent configuration for each Amazon ECS Fargate task that will run, GuardDuty will add a sidecar container for each container workload within that task. The GuardDuty security agent gets deployed to this sidecar container. This is how GuardDuty gets visibility into the runtime behavior of the containers inside the Amazon ECS tasks.

Presently, Runtime Monitoring supports managing the security agent for your Amazon ECS clusters (AWS Fargate) only through GuardDuty. There is no support for managing the security agent manually on Amazon ECS clusters.

Before you configure your accounts, assess how you want to manage the GuardDuty security agent and potentially monitor the runtime behavior of the containers that belong to the Amazon ECS tasks. Consider the following approaches.

Topics

- Manage GuardDuty security agent for most of the Amazon ECS clusters but exclude some of the Amazon ECS clusters
- Manage GuardDuty security agent for selective Amazon ECS clusters

Manage GuardDuty security agent for all Amazon ECS clusters

This approach will help you detect potential security threats at account level. Use this approach when you want GuardDuty to detect potential security threats for all the Amazon ECS clusters that belong to your account.

Manage GuardDuty security agent for most of the Amazon ECS clusters but exclude some of the Amazon ECS clusters

Use this approach when you want GuardDuty to detect potential security threats for most of the Amazon ECS clusters in your AWS environment but exclude some of the clusters. This approach helps you monitor the runtime behavior of the containers within your Amazon ECS tasks at the cluster level. For example, the number of Amazon ECS clusters that belong to your account are 1000. However, you want to monitor only 930 Amazon ECS clusters.

This approach requires you to add a pre-defined GuardDuty tag to the Amazon ECS clusters that you don't want to monitor. For more information, see Managing automated security agent for Fargate (Amazon ECS only).

Manage GuardDuty security agent for selective Amazon ECS clusters

Use this approach when you want GuardDuty to detect potential security threats for some of the Amazon ECS clusters. This approach helps you monitor the runtime behavior of the containers within your Amazon ECS tasks at the cluster level. For example, the number of Amazon ECS clusters that belong to your account are 1000. However, you want to monitor 230 clusters only.

This approach requires you to add a pre-defined GuardDuty tag to the Amazon ECS clusters that you want to monitor. For more information, see Managing automated security agent for Fargate (Amazon ECS only).

Amazon EKS clusters - Approaches to manage GuardDuty security agent

For GuardDuty to consume the runtime events from your EKS clusters at account level or cluster level, it is required to manage the GuardDuty security agent for the corresponding clusters.

Approaches to manage GuardDuty security agent

Prior to September 13, 2023, you could configure GuardDuty to manage the security agent at the account level. This behavior indicated that by default, GuardDuty will manage the security agent on all the EKS clusters that belong to an AWS account. Now, GuardDuty provides a granular capability to help you choose the EKS clusters where you want GuardDuty to manage the security agent.

When you choose to Manage GuardDuty security agent manually, you can still select the EKS clusters that you want to monitor. However, to manage the agent manually, creating a Amazon VPC endpoint for your AWS account is a prerequisite.

> **Note**
>
> Regardless of the approach that you use to manage the GuardDuty security agent, EKS Runtime Monitoring is always enabled at the account level.

Topics

- Manage security agent through GuardDuty
- Manage GuardDuty security agent manually

Manage security agent through GuardDuty

GuardDuty deploys and manages the security agent on your behalf. At any point in time, you can monitor the EKS clusters in your account by using one of the following approaches.

Topics

- Monitor all of the EKS clusters
- Monitor all of the EKS clusters and exclude selective EKS clusters
- Monitor selective EKS clusters

Monitor all of the EKS clusters

- **When to use this approach –** Use this approach when you want GuardDuty to deploy and manage the security agent for all the EKS clusters in your account. By default, GuardDuty will also deploy the security agent on a potentially new EKS cluster created in your account.

- **Impact of using this approach**:
  - GuardDuty creates an Amazon Virtual Private Cloud (Amazon VPC) endpoint through which the GuardDuty security agent delivers the runtime events to GuardDuty. There is no additional cost for the creation of the Amazon VPC endpoint when you manage the security agent through GuardDuty.
  - It is required that your worker node has a valid network path to an active `guardduty-data` VPC endpoint. GuardDuty deploys the security agent on your EKS clusters. Amazon Elastic Kubernetes Service (Amazon EKS) will coordinate the deployment of the security agent on the nodes within the EKS clusters.
  - On the basis of IP availability, GuardDuty selects the subnet to create a VPC endpoint. If you use advanced network topologies, you must validate that the connectivity is possible.
- **Consideration –** Presently, when you use this option, EKS Runtime Monitoring doesn't create a shared VPC.

Monitor all of the EKS clusters and exclude selective EKS clusters

- **When to use this approach –** Use this approach when you want GuardDuty to manage the security agent for all EKS clusters in your account but exclude selective EKS clusters. This method uses a tag-based[1] approach wherein you can tag the EKS clusters for which you don't want to receive the runtime events. The pre-defined tag must have `GuardDutyManaged-false` as the key-value pair.
- **Impact of using this approach**:
  - This approach requires that you to enable GuardDuty agent auto-management only after adding tags to the EKS clusters that you want to exclude from monitoring.

    Therefore, the impact when you Manage security agent through GuardDuty applies to this approach too. When you add tags prior to enabling GuardDuty agent auto-management, GuardDuty will neither deploy nor manage the security agent for the EKS clusters that are excluded from monitoring.
- **Considerations**:
  - You must add the tag key-value pair as `GuardDutyManaged:false` for the selective EKS clusters before enabling Automated agent configuration otherwise, the GuardDuty security agent will be deployed on all the EKS clusters until you use the tag.
  - You must prevent the tags from being modified, except by trusted identities.

> ⚠ **Important**
>
> Manage permissions for modifying the value of the `GuardDutyManaged` tag for your EKS cluster by using service control policies or IAM policies. For more information, see Service control policies (SCPs) in the *AWS Organizations User Guide* or Control access to AWS resources in the *IAM User Guide*.

- For a potentially new EKS cluster that you don't want to monitor, make sure to add the `GuardDutyManaged`-`false` key-value pair at the time of creating this EKS cluster.
- This approach will also have the same consideration as specified for Monitor all of the EKS clusters.

Monitor selective EKS clusters

- **When to use this approach** – Use this approach when you want GuardDuty to deploy and manage the updates to the security agent only for selective EKS clusters in your account. This method uses a tag-based[1] approach wherein you can tag the EKS cluster for which you want to receive the runtime events.
- **Impact of using this approach**:
  - By using inclusion tags, GuardDuty will automatically deploy and manage the security agent only for the selective EKS clusters that are tagged with `GuardDutyManaged`-`true` as the key-value pair.
  - Using this approach will also have the same impact as specified for Monitor all of the EKS clusters.
- **Considerations**:
  - If the value of the `GuardDutyManaged` tag is not set to `true`, the inclusion tag will not work as expected and this may impact monitoring your EKS cluster.
  - To ensure that your selective EKS clusters are being monitored, you need to prevent the tags from being modified, except by trusted identities.

> ⚠ **Important**
>
> Manage permissions for modifying the value of the `GuardDutyManaged` tag for your EKS cluster by using service control policies or IAM policies. For more information, see

Service control policies (SCPs) in the *AWS Organizations User Guide* or Control access to AWS resources in the *IAM User Guide*.

- For a potentially new EKS cluster that you don't want to monitor, make sure to add the `GuardDutyManaged-false` key-value pair at the time of creating this EKS cluster.
- This approach will also have the same consideration as specified for Monitor all of the EKS clusters.

[1] For more information about tagging selective EKS clusters, see Tagging your Amazon EKS resources in the Amazon EKS User Guide.

Manage GuardDuty security agent manually

- **When to use this approach –** Use this approach when you want deploy and manage the GuardDuty security agent on all of your EKS clusters manually. Ensure that EKS Runtime Monitoring is enabled for your accounts. The GuardDuty security agent may not work as expected if you don't enable EKS Runtime Monitoring.
- **Impact of using this approach –** You will need to coordinate the deployment of the GuardDuty security agent software within your EKS clusters across all accounts and AWS Regions where this feature is available.
- **Considerations –** You must support secure data flow while monitoring for and addressing coverage gaps as new clusters and workloads are continuously deployed.

Enabling GuardDuty Runtime Monitoring

Before enabling Runtime Monitoring in your account, make sure that the resource type for which you want to monitor the runtime events, supports the platforms requirements. For more information, see Prerequisites.

If you have been using EKS Runtime Monitoring prior to the launch of Runtime Monitoring, you can use the APIs to check and update the existing configuration for EKS Runtime Monitoring. You can also migrate your existing configuration from EKS Runtime Monitoring to Runtime Monitoring. For more information, see Migrating from EKS Runtime Monitoring to Runtime Monitoring.

Note

Presently, this documentation provides steps to enable Runtime Monitoring for your accounts and organization by console only. You can also enable Runtime Monitoring by using API Actions or AWS CLI for GuardDuty.

You can configure Runtime Monitoring by using the steps in the following topics.

Contents

Prerequisites to enabling Runtime Monitoring

To enable Runtime Monitoring and manage the GuardDuty security agent, you must meet the prerequisites for each resource type that you want to monitor for threat detection.

Contents

Prerequisites for Amazon EC2 instance support

Make EC2 instances SSM managed

The Amazon EC2 instances for which you want GuardDuty to monitor runtime events must be AWS Systems Manager (SSM) managed. This is regardless of whether you use GuardDuty to manage the security agent automatically or manage it manually (except Method 2 - By using RPM installation scripts).

To manage your Amazon EC2 instances with AWS Systems Manager, see Setting up Systems Manager for Amazon EC2 instances in the *AWS Systems Manager User Guide*.

Validating architectural requirements

The architecture of your OS distribution might impact how the GuardDuty security agent will behave. You must meet the following requirements before using Runtime Monitoring for Amazon EC2 instances:

- Presently, the Runtime Monitoring support for Amazon EC2 is available for Linux versions only. Although the support for Ubuntu is not available right now, it will be in the near future. To receive notification about updates to this page, subscribe to the RSS feed.

  The following table shows the OS distribution that has been verified to support the GuardDuty security agent for Amazon EC2 instances.

OS distribution	Kernel version	Kernel support	CPU architecture	
			x64 (AMD64)	Graviton (ARM64)
AL2 and AL2023	5.4, 5.10, 5.15, 6.1	eBPF, Tracepoin ts, Kprobe	Supported	Supported

- Additional requirements - Only if you have Amazon ECS/Amazon EC2

  For Amazon ECS/Amazon EC2, we recommend that you use the latest Amazon ECS-optimized AMIs (dated September 29, 2023 or later), or use Amazon ECS agent version v1.77.0.

When using automated agent configuration

To Use automated agent configuration (recommended), your AWS account must meet the following prerequisites:

- When using inclusion tags with automated agent configuration, for GuardDuty to create an SSM association for a new instance, ensure that the new instance is SSM managed and shows up under **Fleet Manager** in the https://console.aws.amazon.com/systems-manager/ console.
- When using exclusion tags with automated agent configuration:
  - Add the `GuardDutyManaged:false` tag before configuring the GuardDuty automated agent for your account.

Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2 instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

- For the exclusion tags to work, update the instance configuration so that the instance identity document is available in instance metadata service (IMDS). Procedure to do this step is already a part of Enabling Runtime Monitoring for your account.

CPU and memory limit for GuardDuty agent

CPU limit

The maximum CPU limit for the GuardDuty security agent associated with Amazon EC2 instances is 10 percent of the total vCPU cores. For example, if your EC2 instance has 4 vCPU cores, then the security agent can use a maximum of 40 percent out of the total available 400 percent.

Memory limit

From the memory associated with your Amazon EC2 instance, there is a limited memory that the GuardDuty security agent can use.

The following table shows the memory limit.

Memory of the Amazon EC2 instance	Maximum memory for GuardDuty agent
Less than 8 GB	128 MB
Less than 32 GB	256 MB
More than or equal to 32 GB	1 GB

Next step

The next step is to configure Runtime Monitoring and also manage the security agent (automatically or manually).

Prerequisites for AWS Fargate (Amazon ECS only) support

Validating architectural requirements

The platform that you use may impact how GuardDuty security agent supports GuardDuty in receiving the runtime events from your Amazon ECS clusters. You must validate that you're using one of the verified platforms.

Initial considerations:

> The AWS Fargate (Fargate) platform for your Amazon ECS clusters must be Linux. The corresponding platform version must be at least `1.4.0`, or `LATEST`. For more information about the platform versions, see [Linux platform versions](#) in the *Amazon Elastic Container Service Developer Guide*.
>
> The Windows platform versions are not yet supported.

Verified platforms

The OS distribution and CPU architecture impacts the support provided by the GuardDuty security agent. The following table shows the verified configuration for deploying the GuardDuty security agent and configuring Runtime Monitoring.

OS distribution	**Kernel support**	**CPU architecture**	
		x64 (AMD64)	**Graviton (ARM64)**
Linux	eBPF, Tracepoints, Kprobe	Supported	Supported

Provide ECR permissions and subnet details

Before enabling Runtime Monitoring, you must provide the following details:

Provide a task execution role with permissions

> The task execution role requires you to have certain Amazon Elastic Container Registry (Amazon ECR) permissions. You can either use the [AmazonECSTaskExecutionRolePolicy](#) managed policy or add the following permissions to your `TaskExecutionRole` policy:

```
...
      "ecr:GetAuthorizationToken",
      "ecr:BatchCheckLayerAvailability",
      "ecr:GetDownloadUrlForLayer",
      "ecr:BatchGetImage",
...
```

To further restrict the Amazon ECR permissions, you can add the Amazon ECR repository URI that hosts the GuardDuty security agent for AWS Fargate (Amazon ECS only). For more information, see Repository for GuardDuty agent on AWS Fargate (Amazon ECS only).

Provide subnet details in task definition

You can either provide the public subnets as an input in your task definition or create an Amazon ECR VPC endpoint.

- **Using task definition option** – Running the CreateService and UpdateService APIs in the *Amazon Elastic Container Service API Reference* requires you to pass the subnet information. For more information, see Amazon ECS task definitions in the *Amazon Elastic Container Service Developer Guide*.
- **Using the Amazon ECR VPC endpoint option** – Provide network path to Amazon ECR - Ensure that the Amazon ECR repository URI that hosts the GuardDuty security agent is network accessible. If your Fargate tasks will run in a private subnet, then Fargate will need the network path to download the GuardDuty container.

  For information about enabling Fargate to download the GuardDuty container, see Using Amazon ECR with Amazon ECS in the *Amazon Elastic Container Service Developer Guide*.

CPU and memory limits

In the Fargate task definition, you must specify the CPU and memory value at the task level. The following table shows the valid combinations of task-level CPU and memory values, and the corresponding GuardDuty security agent maximum memory limit for the GuardDuty container.

CPU value	Memory value	GuardDuty agent maximum memory limit
256 (.25 vCPU)	512 MiB, 1 GB, 2GB	128 MB

CPU value	Memory value	GuardDuty agent maximum memory limit
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	
1024 (1 vCPU)	2 GB, 3 GB, 4 GB	
	5 GB, 6 GB, 7 GB, 8 GB	
2048 (2 vCPU)	Between 4 GB and 16 GB in 1 GB increments	
4096 (4 vCPU)	Between 8 GB and 20 GB in 1 GB increments	
8192 (8 vCPU)	Between 16 GB and 28 GB in 4 GB increments	256 MB
	Between 32 GB and 60 GB in 4 GB increments	512 MB
16384 (16 vCPU)	Between 32 GB and 120 GB in 8 GB increments	1 GB

After you enable Runtime Monitoring and assess that the coverage status of your cluster is **Healthy**, you can set up and view the Container insight metrics. For more information, Setting up monitoring on Amazon ECS cluster.

The next step is to configure Runtime Monitoring and also configure the security agent.

Prerequisites for Amazon EKS cluster support

Validating architectural requirements

The platform that you use may impact how GuardDuty security agent supports GuardDuty in receiving the runtime events from your EKS clusters. You must validate that you're using one of the verified platforms. If you're managing the GuardDuty agent manually, ensure that the Kubernetes version supports the GuardDuty agent version that is currently in use.

Verified platforms

The OS distribution, kernel version, and CPU architecture affect the support provided by the GuardDuty security agent. The following table shows the verified configuration for deploying the GuardDuty security agent and configuring EKS Runtime Monitoring.

OS distribution	Kernel version	Kernel support	CPU architecture		Supported Kubernetes version
			x64 (AMD64)	Graviton (ARM64) (Graviton2 and above)[1]	
Ubuntu	5.4, 5.10, 5.15, 6.1	eBPF Tracepoints, Kprobe	Supported	Supported	v1.21 - v1.29
AL2					
Bottlerocket					v1.23 - v1.29

[1] Runtime Monitoring for Amazon EKS clusters doesn't support the first generation Graviton instance such as A1 instance types.

Kubernetes versions supported by GuardDuty security agent

The following table shows the Kubernetes versions for your EKS clusters that are supported by GuardDuty security agent.

Kubernetes version	Amazon EKS add-on GuardDuty security agent version							
	v1.5.0	v1.4.1	v1.4.0	v1.3.1	v1.3.0	v1.2.0	v1.1.0	v1.0.0
1.29	Supported	Supported	Supported	Not supported	Not supported	Not supported	Not supported	Not supported
1.28				Supported	Supported			
1.27						Supported		

1.26	Supported	
1.25		Supported
1.24		
1.23		
1.22		
1.21		

Some of the GuardDuty security agent versions will reach end of standard support. For information about the agent release versions, see GuardDuty security agent for Amazon EKS clusters.

CPU and memory limits

The following table shows the CPU and memory limits for the Amazon EKS add-on for GuardDuty (`aws-guardduty-agent`).

Parameter	Minimum limit	Maximum limit
CPU	200m	1000m
Memory	256 Mi	1024 Mi

When you use Amazon EKS add-on version 1.5.0 or above, GuardDuty provides the capability to configure the add-on schema for your CPU and memory values. For information about the configurable range, see Configurable parameters and values.

After you enable EKS Runtime Monitoring and assess the coverage status of your EKS clusters, you can set up and view the container insight metrics. For more information, see Setting up CPU and memory monitoring.

Next step

The next step is to configure Runtime Monitoring, and also manage the security agent either manually or automatically through GuardDuty.

Enabling Runtime Monitoring for a standalone account

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Runtime Monitoring**.
3. Under the **Configuration** tab, choose **Enable** to enable Runtime Monitoring for your account.
4. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)
   - Managing security agent automatically for Amazon EKS clusters
   - Managing security agent manually for Amazon EKS cluster

Enabling Runtime Monitoring for multiple-account environments

In a multiple-account environments, only the delegated GuardDuty administrator account can enable or disable Runtime Monitoring for the member accounts, and manage automated agent configuration for the resource types belonging to the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account account manages their member accounts using AWS Organizations. For more information about multi-account environments, see Managing multiple accounts.

For delegated GuardDuty administrator account

To enable Runtime Monitoring for delegated GuardDuty administrator account

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

2. In the navigation pane, choose **Runtime Monitoring**.
3. Under the **Configuration** tab, choose **Edit** in the **Runtime Monitoring configuration** section.
4. **Using Enable for all accounts**

   If you want to enable Runtime Monitoring for all the accounts that belong to the organization, including the delegated GuardDuty administrator account, then choose **Enable for all accounts**.
5. **Using Configure accounts manually**

   If you want to enable Runtime Monitoring for each member account individually, then choose **Configure accounts manually**.

   - Choose **Enable** under the **Delegated Administrator (this account)** section.
6. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)
   - Managing security agent automatically for Amazon EKS clusters
   - Managing security agent manually for Amazon EKS cluster

For all member accounts

To enable Runtime Monitoring for all member accounts in the organization

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account.
2. In the navigation pane, choose **Runtime Monitoring**.
3. On the Runtime Monitoring page, under the **Configuration** tab, choose **Edit** in the **Runtime Monitoring configuration** section.
4. Choose **Enable for all accounts**.

5. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)
   - Managing security agent automatically for Amazon EKS clusters
   - Managing security agent manually for Amazon EKS cluster

For all existing active member accounts

To enable Runtime Monitoring for existing member accounts in the organization

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account for the organization.
2. In the navigation pane, choose **Runtime Monitoring**.
3. On the **Runtime Monitoring** page, under the **Configuration** tab, you can view the current status of the Runtime Monitoring configuration.
4. Within the Runtime Monitoring pane, under the **Active member accounts** section, choose **Actions**.
5. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
6. Choose **Confirm**.
7. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)

- Managing security agent automatically for Amazon EKS clusters
- Managing security agent manually for Amazon EKS cluster

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

Auto-enable Runtime Monitoring for new member accounts only

To enable Runtime Monitoring for new member accounts in your organization

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the designated delegated GuardDuty administrator account of the organization.
2. In the navigation pane, choose **Runtime Monitoring**
3. Under the **Configuration** tab, choose **Edit** in the **Runtime Monitoring configuration** section.
4. Choose **Configure accounts manually**.
5. Select **Automatically enable for new member accounts**.
6. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)
   - Managing security agent automatically for Amazon EKS clusters
   - Managing security agent manually for Amazon EKS cluster

For selective active member accounts only

To enable Runtime Monitoring for individual active member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **Accounts**.
3. On the **Accounts** page, review values in the **Runtime Monitoring** and **Manage agent automatically** columns. These values indicate whether Runtime Monitoring and GuardDuty agent management are **Enabled** or **Not enabled** for the corresponding account.
4. From the Accounts table, select the account for which you want to enable Runtime Monitoring. You can choose multiple accounts at a time.
5. Choose **Confirm**.
6. Choose **Edit protection plans**. Choose the appropriate action.
7. Choose **Confirm**.
8. For GuardDuty to receive the runtime events from one or more resource types – an Amazon EC2 instance, Amazon ECS cluster, or an Amazon EKS cluster, use the following options to manage the security agent for these resources:

   To enable GuardDuty security agent

   - Managing automated security agent for Amazon EC2 instance
   - Managing security agent manually for Amazon EC2 instance
   - Managing automated security agent for Fargate (Amazon ECS only)
   - Managing security agent automatically for Amazon EKS clusters
   - Managing security agent manually for Amazon EKS cluster

Managing GuardDuty security agents

You can manage the GuardDuty security agent for the resource that you want to monitor. If you want to monitor more than one resource type, make sure to manage the GuardDuty agent for that resource.

⚠ **Important**

When working with GuardDuty security agent for an Amazon EC2 instance, you might install and use the agent on the underlying host within an Amazon EKS cluster. If you had already deployed a security agent on that EKS cluster, the same host could have two security agents running on it at the same time. For information about how GuardDuty works in this scenario, see Handling dual security agents.

The following topics will help you with the next steps to manage the security agent.

Contents

Using shared VPC with automated security agents

When you choose GuardDuty to manage the security agent automatically, Runtime Monitoring supports using a shared VPC for the AWS accounts that belong to the same organization in AWS Organizations. On your behalf, GuardDuty can set the Amazon VPC endpoint policy based on the details associated with the shared VPC for your organization.

Prior to this release, GuardDuty supported the use of shared VPCs only when you chose to manage the GuardDuty security agent manually.

Contents

How it works

When the owner account of the shared VPC enables Runtime Monitoring and automated agent configuration for any of the resources (Amazon EKS or AWS Fargate (Amazon ECS only)), all the shared VPCs become eligible for automatic installation of the shared Amazon VPC endpoint and the associated security group in the shared VPC owner account. GuardDuty retrieves the organization ID that is associated with the shared Amazon VPC.

Now, the AWS accounts that belong to the same organization as the shared Amazon VPC owner account can also share the same Amazon VPC endpoint. GuardDuty creates the shared VPC when either the shared VPC owner account or the participating account needs an Amazon VPC endpoint. Examples of needing an Amazon VPC endpoint include enabling GuardDuty, Runtime Monitoring, EKS Runtime Monitoring, or launching a new Amazon ECS-Fargate task. When these accounts enable Runtime Monitoring and automated agent configuration for any resource type, GuardDuty creates a Amazon VPC endpoint and sets the endpoint policy with the same organization ID as that of the shared VPC owner account. GuardDuty adds a `GuardDutyManaged` tag and sets it to `true` for the Amazon VPC endpoint that GuardDuty creates. If the shared Amazon VPC owner account has not enabled Runtime Monitoring or automated agent configuration for any of the resources, GuardDuty will not set the Amazon VPC endpoint policy. For information about configuring Runtime Monitoring and managing the security agent automatically in the shared VPC owner account, see Enabling GuardDuty Runtime Monitoring.

Each of the accounts using the same Amazon VPC endpoint policy is called as the **participant AWS account** of the associated shared Amazon VPC.

The following example shows the default VPC endpoint policy of the shared VPC owner account and the participant account. The `aws:PrincipalOrgID` will show the organization ID associated with the shared VPC resource. The use of this policy is limited to the participant accounts present in the organization of the owner account.

Example

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Action": "*",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": "*"
```

```
        },
        {
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalOrgID": "o-abcdef0123"
                }
            },
            "Action": "*",
            "Resource": "*",
            "Effect": "Deny",
            "Principal": "*"
        }
    ]
}
```

Prerequisites for using shared VPC

Prerequisites for initial setup

Perform the following steps in the AWS account that you want to be the owner of the shared VPC:

1. **Creating an organization** – Create an organization by following the steps in Creating and managing an organization in the *AWS Organizations User Guide*.

   For information about adding or removing member accounts, see Managing AWS accounts in your organization.

2. **Creating a shared VPC resource** – You can create a shared VPC resource from the owner account. For more information, see Share your VPC with other accounts in the *Amazon VPC User Guide*.

Prerequisites specific to GuardDuty Runtime Monitoring

The following list provides the prerequisites that are specific to GuardDuty:

- The owner account of the shared VPC and the participating account can be from different organizations in GuardDuty. However, they must belong to the same organization in AWS Organizations. This is required for GuardDuty to create an Amazon VPC endpoint and a security group for the shared VPC. For information about how shared VPCs work, see Share your VPC with other accounts in *Amazon VPC User Guide*.

- Enable Runtime Monitoring or EKS Runtime Monitoring, and GuardDuty automated agent configuration for any resource in the shared VPC owner account and the participant account. For more information, see Enabling Runtime Monitoring.

  If you have already completed these configurations, continue with the next step.
- When working with either an Amazon EKS or an Amazon ECS (AWS Fargate only) task, make sure to choose the shared VPC resource associated with the owner account and select its subnets.

Frequently asked questions (FAQs)

The following list provides the troubleshooting steps to the frequently asked questions when using a shared VPC resource with GuardDuty automated agent configuration enabled in Runtime Monitoring:

I am already using Runtime Monitoring (or EKS Runtime Monitoring). How do I enable shared VPC?

For information about prerequisites to create a shared VPC, see Prerequisites.

When both the shared VPC owner account and the participant account have met the prerequisites, GuardDuty will attempt to set the Amazon VPC endpoint policy automatically.

If prior to this release, your AWS account experienced a coverage issue about the shared VPC not being supported, follow the prerequisites. When your resource type (Amazon EKS or Amazon ECS (AWS Fargate only) task) invokes the requirement of a shared VPC endpoint, GuardDuty will attempt to set the new VPC endpoint policy.

As a shared VPC owner account, I want the shared VPC endpoint policy to be restricted to a subset of participant accounts in my organization. How can I do that?

If you have a `GuardDutyManaged:true` tag associated with the endpoint, remove it. This prevents GuardDuty to attempt modifying or overriding the VPC endpoint policy of the shared VPC.

For more information, see Control access to VPC endpoints using endpoint policies.

Why does the shared VPC endpoint modify from `aws:PrincipalAccount` to `aws:PrincipalOrgId`? How can I prevent that?

When GuardDuty detects that the VPC is shared by multiple accounts of the same organization in AWS Organizations, GuardDuty attempts to modify the policy to specify the organization ID.

To prevent this, remove the `GuardDutyManaged:true` tag from the shared VPC endpoint. This prevents GuardDuty to attempt modifying or overriding the VPC endpoint policy of the shared VPC.

What happens when the shared VPC owner account or one of the participant accounts disables GuardDuty or Runtime Monitoring (or EKS Runtime Monitoring)?

When the shared VPC owner account disables GuardDuty or Runtime Monitoring (or EKS Runtime Monitoring), GuardDuty checks whether any resource type belonging to the participant account has used the shared VPC endpoint or any participant account has ever enabled GuardDuty agent management for any resource type. If yes, GuardDuty won't delete the VPC endpoint and the security group.

If the shared VPC participant account disables GuardDuty or Runtime Monitoring (or EKS Runtime Monitoring), then there is no impact on the shared VPC owner account and the owner account will neither delete the shared VPC resource nor the security group.

How can I delete the shared VPC resource? What will be its impact?

As a shared VPC owner account, you can delete the shared VPC resource even when it is being used by your account or any of the participating accounts in Runtime Monitoring. For information about deleting the shared VPC and understanding its impact, see To delete VPC endpoint.

Handling dual security agents installed on a host

Amazon EC2 instances can support multiple types of workloads. When you configure automated security agent on an Amazon EC2 instance, the same EC2 instance might have another security agent through EKS.

Overview

Consider a scenario where you have enabled Runtime Monitoring. Now, you enable the automated agent for Amazon EKS through GuardDuty. You have also enabled the automated agent for Amazon EC2. It may happen that the same underlying host gets installed with two security agents - one for Amazon EKS and the other for Amazon EC2. This could result in two security agents running inside the same host, collecting runtime events and sending them to GuardDuty, and potentially generating duplicate findings.

Impact

- When there is more than one security agent running on the same host, your account may experience double the amount of CPU and memory processing needs. For information about the CPU and memory limits for each resource type, see Prerequisites for that resource.
- GuardDuty has designed the Runtime Monitoring feature in a way that even if there is an overlap of two security agents collecting runtime events from the same underlying host, your account will only be charged for one stream of runtime events.

How GuardDuty handles multiple agents

GuardDuty detects when two security agents are running on the same host and designates only one of them to be the security agent that actively collects runtime events. The second agent will consume minimum system resources so as to prevent any impact to the performance of your applications.

GuardDuty considers the following scenarios:

- When an EC2 instance falls under the scope of both Amazon EKS and Amazon EC2 security agents, the EKS security agent takes priority. This will apply only when you use the security agent v1.1.0 or above for Amazon EC2. Older agent versions will continue to run and collect runtime events because older agent versions are not affected by prioritization.
- When both Amazon EKS and Amazon EC2 have GuardDuty managed security agents and your Amazon EC2 instance is also SSM managed, both the security agents will be installed at the host level. Once the agents are installed, GuardDuty decides which security agent will keep running. When both the security agents are running, eventually only one of them will collect runtime events.
- When the security agents associated with both EC2 and EKS run at the same time, GuardDuty might generate duplicate findings during the overlap period only.

  This can happen when:

  - Security agents for both EC2 and EKS are configured through GuardDuty (automatically), or
  - Your Amazon EKS resource has automated security agent.
- When the EKS security agent is already running, if you deploy the EC2 security agent manually on the same underlying host and meet all the prerequisites, GuardDuty might not install a second security agent.

Managing automated security agent for Amazon EC2 instance

Note

Before you continue, make sure to follow all the Prerequisites for Amazon EC2 instance support.

Migrating from Amazon EC2 manual agent to automated agent

This section applies to your AWS account if you were previously managing the security agent manually and now want to use the GuardDuty automated agent configuration. If this doesn't apply to you, continue with configuring the security agent for your account.

When you enable GuardDuty automated agent, GuardDuty manages the security agent on your behalf. For information about what steps does GuardDuty take, see Use automated agent configuration (recommended).

Clean up resources

Delete SSM association

- Delete any SSM association that you may have created when you were managing the security agent for Amazon EC2 manually. For more information, see Deleting associations.
- This is done so that GuardDuty can take over the management of SSM actions whether you use automated agents at the account level or instance level (by using inclusion or exclusion tags). For more information about what SSM actions can GuardDuty take, see Service-linked role permissions for GuardDuty.
- When you delete an SSM association that was previously created for managing the security agent manually, there might be a brief period of overlap when GuardDuty creates an SSM association for managing the security agent automatically. During this period, you could experience conflicts based on SSM scheduling. For more information, see Amazon EC2 SSM scheduling.

Manage inclusion and exclusion tags for your Amazon EC2 instances

- **Inclusion tags –** When you don't enable GuardDuty automated agent configuration but tag any of your Amazon EC2 instances with an inclusion tag (`GuardDutyManaged`:`true`), GuardDuty creates an SSM association that will install and manage the security agent on the selected EC2 instances. This is an expected behavior that helps you manage the security agent on selected EC2 instances only. For more information, see How Runtime Monitoring works with Amazon EC2 instances.

To prevent GuardDuty from installing and managing the security agent, remove the inclusion tag from these EC2 instances. For more information, see Add and delete tags in the *Amazon EC2 User Guide for Linux Instances*.

- **Exclusion tags** – When you want to enable GuardDuty automated agent configuration for all the EC2 instances in your account, make sure that no EC2 instance is tagged with an exclusion tag (`GuardDutyManaged`:`false`).

Configuring GuardDuty agent for standalone account

Configure for all instances

To configure GuardDuty security agent for all instances in your standalone account

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Runtime Monitoring**.
3. Under the **Configuration** tab, choose **Edit**.
4. In the **EC2** section, choose **Enable**.
5. Choose **Save**.
6. You can verify that the SSM association that GuardDuty creates will install and manage the security agent on all the EC2 resources belonging to your account.

    a. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

    b. Open the **Targets** tab for the SSM association (`GuardDutyRuntimeMonitoring-do-not-delete`). Observe that the **Tag key** appears as **InstanceIds**.

Using inclusion tag in selected instances

To configure GuardDuty security agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. Add the `GuardDutyManaged:true` tag to the instances that you want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. You can verify that the SSM association that GuardDuty creates will install and manage the security agent only on the EC2 resources that are tagged with the inclusion tags.

   Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

   - Open the **Targets** tab for the SSM association that gets created (`GuardDutyRuntimeMonitoring-do-not-delete`). The **Tag key** appears as **tag:GuardDutyManaged**.

Using exclusion tag in selected instances

> **Note**
>
> Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2 instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

To configure GuardDuty security agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:false` tag to the instances that you **don't** want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. **For the exclusion tags to be available in the instance metadata, perform the following steps:**

   a. Under the **Details** tab of your instance, view the status for **Allow tags in instance metadata**.

      If it is currently **Disabled**, use the following steps to change the status to **Enabled**. Otherwise, skip this step.

   b. Select the instance for which you want to allow tags.

   c. Under the **Actions** menu, choose **Instance settings**.

   d. Choose **Allow tags in instance metadata**.

   e. Under **Access to tags in instance metadata**, select **Allow**.

   f. Choose **Save**.

4. After you have added the exclusion tag perform the same steps as sepcified in the **Configure for all instances** tab.

You can now assess runtime Coverage for Amazon EC2 instance.

Configuring GuardDuty agent in multiple-account environment

For delegated GuardDuty administrator account

Configure for all instances

If you chose **Enable for all accounts** for Runtime Monitoring, then choose one of the following options for the delegated GuardDuty administrator account:

- **Option 1**

  Under **Automated agent configuration**, in the **EC2** section, select **Enable for all accounts**.
- **Option 2**
  - Under **Automated agent configuration**, in the **EC2** section, select **Configure accounts manually**.
  - Under **Delegated Administrator (this account)**, choose **Enable**.
- Choose **Save**.

If you chose **Configure accounts manually** for Runtime Monitoring, then perform the following steps:

- Under **Automated agent configuration**, in the **EC2** section, select **Configure accounts manually**.
- Under **Delegated Administrator (this account)**, choose **Enable**.
- Choose **Save**.

Regardless of which option you choose to enable the automated agent configuration for delegated GuardDuty administrator account, you can verify that the SSM association that GuardDuty creates will install and manage the security agent on all the EC2 resources belonging to this account.

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Open the **Targets** tab for the SSM association (`GuardDutyRuntimeMonitoring-do-not-delete`). Observe that the **Tag key** appears as **InstanceIds**.

Using inclusion tag in selected instances

To configure GuardDuty agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged`:`true` tag to the instances that you want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.

   Adding this tag will permit GuardDuty to install and manage the security agent for these selected EC2 instances. You **don't** need to enable automated agent configuration explicitly.
3. You can verify that the SSM association that GuardDuty creates will install and manage the security agent only on the EC2 resources that are tagged with the inclusion tags.

   Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

   - Open the **Targets** tab for the SSM association that gets created (`GuardDutyRuntimeMonitoring-do-not-delete`). The **Tag key** appears as **tag:GuardDutyManaged**.

Using exclusion tag in selected instances

> **Note**
>
> Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2

> instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

To configure GuardDuty agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:false` tag to the instances that you **don't** want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. **For the exclusion tags to be available in the instance metadata, perform the following steps:**

   a. Under the **Details** tab of your instance, view the status for **Allow tags in instance metadata**.

      If it is currently **Disabled**, use the following steps to change the status to **Enabled**. Otherwise, skip this step.

   b. Under the **Actions** menu, choose **Instance settings**.

   c. Choose **Allow tags in instance metadata**.
4. After you have added the exclusion tag, perform the same steps as specified in the **Configure for all instances** tab.

You can now assess the runtime Coverage for Amazon EC2 instance.

Auto-enable for all member accounts

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

Configure for all instances

The following steps assume that you chose **Enable for all accounts** in the Runtime Monitoring section:

1. Choose **Enable for all accounts** in the **Automated agent configuration** section for **Amazon EC2**.
2. You can verify that the SSM association that GuardDuty creates (`GuardDutyRuntimeMonitoring-do-not-delete`) will install and manage the security agent on all the EC2 resources belonging to this account.

   a. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

   b. Open the **Targets** tab for the SSM association. Observe that the **Tag key** appears as **InstanceIds**.

Using inclusion tag in selected instances

To configure GuardDuty agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged`:`true` tag to the EC2 instances that you want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.

   Adding this tag will permit GuardDuty to install and manage the security agent for these selected EC2 instances. You **don't** need to enable automated agent configuration explicitly.
3. You can verify that the SSM association that GuardDuty creates will install and manage the security agent on all the EC2 resources belonging to your account.

   a. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

   b. Open the **Targets** tab for the SSM association (`GuardDutyRuntimeMonitoring-do-not-delete`). Observe that the **Tag key** appears as **InstanceIds**.

Using exclusion tag in selected instances

> **Note**
>
> Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2

> instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

To configure GuardDuty security agent for selected Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:false` tag to the instances that you **don't** want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. **For the exclusion tags to be available in the instance metadata, perform the following steps:**

   a. Under the **Details** tab of your instance, view the status for **Allow tags in instance metadata**.

      If it is currently **Disabled**, use the following steps to change the status to **Enabled**. Otherwise, skip this step.

   b. Under the **Actions** menu, choose **Instance settings**.

   c. Choose **Allow tags in instance metadata**.
4. After you have added the exclusion tag, perform the same steps as specified in the **Configure for all instances** tab.

You can now assess the runtime Coverage for Amazon EC2 instance.

Auto-enable for new member accounts only

The delegated GuardDuty administrator account can set the automated agent configuration for Amazon EC2 resource to enable automatically for the new member accounts as they join the organization.

Configure for all instances

The following steps assume that you selected **Automatically enable for new member accounts** under the **Runtime Monitoring** section:

1. In the navigation pane, choose **Runtime Monitoring**.

2. On the **Runtime Monitoring** page, choose **Edit**.
3. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, automated agent configuration for Amazon EC2 will be automatically enabled for their account. Only the delegated GuardDuty administrator account of the organization can modify this selection.
4. Choose **Save**.

When a new member account joins the organization, this configuration will be enabled for them automatically. For GuardDuty to manage the security agent for the Amazon EC2 instances that belong to this new member account, make sure that all the prerequisites For EC2 instance are met.

When an SSM association gets created (`GuardDutyRuntimeMonitoring-do-not-delete`), you can verify that the SSM association will install and manage the security agent on all the EC2 instances belonging to the new member account.

- Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
- Open the **Targets** tab for the SSM association. Observe that the **Tag key** appears as **InstanceIds**.

Using inclusion tag in selected instances

To configure GuardDuty security agent for selected instances in your account

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:true` tag to the instances that you want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.

   Adding this tag will permit GuardDuty to install and manage the security agent for these selected instances. You don't need to enable automated agent configuration explicitly.
3. You can verify that the SSM association that GuardDuty creates will install and manage the security agent only on the EC2 resources that are tagged with the inclusion tags.

a. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

b. Open the **Targets** tab for the SSM association that gets created. The **Tag key** appears as **tag:GuardDutyManaged**.

Using exclusion tag in selected instances

> **Note**
>
> Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2 instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

To configure GuardDuty security agent for specific instances in your standalone account

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:false` tag to the instances that you **don't** want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. **For the exclusion tags to be available in the instance metadata, perform the following steps:**

   a. Under the **Details** tab of your instance, view the status for **Allow tags in instance metadata**.

      If it is currently **Disabled**, use the following steps to change the status to **Enabled**. Otherwise, skip this step.

   b. Under the **Actions** menu, choose **Instance settings**.

   c. Choose **Allow tags in instance metadata**.
4. After you have added the exclusion tag, perform the same steps as specified in the **Configure for all instances** tab.

You can now assess the runtime Coverage for Amazon EC2 instance.

Selective member accounts only

Configure for all instances

1. On the **Accounts** page, select one or more accounts for which you want to enable **Runtime Monitoring-Automated agent configuration (Amazon EC2)**. Make sure that the accounts that you select in this step already have Runtime Monitoring enabled.
2. From **Edit protection plans**, choose the appropriate option to enable **Runtime Monitoring-Automated agent configuration (Amazon EC2)**.
3. Choose **Confirm**.

Using inclusion tag in selected instances

To configure GuardDuty security agent for selected instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged`:`true` tag to the instances that you want GuardDuty to monitor and detect potential threats. For information about adding this tag, see To add a tag to an individual resource.

   Adding this tag will permit GuardDuty to manage the security agent for your tagged Amazon EC2 instances. You don't need to explicitly enable automated agent configuration (**Runtime Monitoring - Automated agent configuration (EC2)**).

Using exclusion tag in selected instances

> **Note**
>
> Ensure that you add the exclusion tag to your Amazon EC2 instances before you launch them. When you enable automated agent configuration for Amazon EC2, any EC2 instance that launches without an exclusion tag will be covered under GuardDuty automated agent configuration.

To configure GuardDuty security agent for selected instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Add the `GuardDutyManaged:false` tag to the EC2 instances that you **don't** want GuardDuty to monitor or detect potential threats. For information about adding this tag, see To add a tag to an individual resource.
3. **For the exclusion tags to be available in the instance metadata, perform the following steps:**

   a. Under the **Details** tab of your instance, view the status for **Allow tags in instance metadata**.

      If it is currently **Disabled**, use the following steps to change the status to **Enabled**. Otherwise, skip this step.

   b. Under the **Actions** menu, choose **Instance settings**.

   c. Choose **Allow tags in instance metadata**.
4. After you have added the exclusion tag, perform the same steps as specified in the **Configure for all instances** tab.

You can now assess Coverage for Amazon EC2 instance.

Managing security agent manually for Amazon EC2 instance

After you enable Runtime Monitoring, you will need to install the GuardDuty security agent manually. By installing the agent, GuardDuty will receive the runtime events from the Amazon EC2 instances.

To manage the GuardDuty security agent, you must create an Amazon VPC endpoint and then follow the steps to install the security agent manually.

Creating Amazon VPC endpoint manually

Before you can install the GuardDuty security agent, you must create an Amazon Virtual Private Cloud (Amazon VPC) endpoint. This will help GuardDuty receive the runtime events of your Amazon EC2 instances.

Note

There is no additional cost for the usage of the VPC endpoint.

To create a Amazon VPC endpoint

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC private cloud**, choose **Endpoints**.
3. Choose **Create Endpoint**.
4. On the **Create endpoint** page, for **Service category**, choose **Other endpoint services**.
5. For **Service name**, enter **`com.amazonaws.`*`us-east-1`*`.guardduty-data`**.

   Make sure to replace *`us-east-1`* with your AWS Region. This must be the same Region as the Amazon EC2 instance that belongs to your AWS account ID.
6. Choose **Verify service**.
7. After the service name is successfully verified, choose the **VPC** where your instance resides. Add the following policy to restrict Amazon VPC endpoint usage to the specified account only. With the organization `Condition` provided below this policy, you can update the following policy to restrict access to your endpoint. To provide the Amazon VPC endpoint support to specific account IDs in your organization, see Organization condition to restrict access to your endpoint.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Action": "*",
   "Resource": "*",
   "Effect": "Allow",
   "Principal": "*"
  },
  {
   "Condition": {
    "StringNotEquals": {
     "aws:PrincipalAccount": "111122223333"
    }
```

```
        },
        "Action": "*",
        "Resource": "*",
        "Effect": "Deny",
        "Principal": "*"
      }
    ]
  }
```

The `aws:PrincipalAccount` account ID must match the account containing the VPC and VPC endpoint. The following list shows how to share the VPC endpoint with other AWS account IDs:

- To specify multiple accounts to access the VPC endpoint, replace `"aws:PrincipalAccount: "111122223333"` with the following block:

  ```
  "aws:PrincipalAccount": [
            "666666666666",
            "555555555555"
        ]
  ```

  Make sure to replace the AWS account IDs with the account IDs of those accounts that need to access the VPC endpoint.

- To allow all the members from an organization to access the VPC endpoint, replace `"aws:PrincipalAccount: "111122223333"` with the following line:

  ```
  "aws:PrincipalOrgID": "o-abcdef0123"
  ```

  Make sure to replace the organization *`o-abcdef0123`* with your organization ID.

- To restrict accessing a resource by an organization ID, add your `ResourceOrgID` to the policy. For more information, see `aws:ResourceOrgID` in the *IAM User Guide*.

  ```
  "aws:ResourceOrgID": "o-abcdef0123"
  ```

8. Under **Additional settings**, choose **Enable DNS name**.
9. Under **Subnets**, choose the subnets in which your instance resides.
10. Under **Security groups**, choose a security group that has the in-bound port 443 enabled from your VPC (or your Amazon EC2 instance). If you don't already have a security group that has an

in-bound port 443 enabled, see Create a security group in the *Amazon EC2 User Guide for Linux Instances*.

If there is an issue while restricting the in-bound permissions to your VPC (or instance), provide the support to in-bound 443 port from any IP address (`0.0.0.0/0`).

Installing the security agent manually

GuardDuty provides the following two methods to install the GuardDuty security agent on your Amazon EC2 instances:

- Method 1 - By using AWS Systems Manager – This method requires your Amazon EC2 instance to be AWS Systems Manager managed.
- Method 2 - By using RPM installation scripts – You can use this method whether or not your Amazon EC2 instances are AWS Systems Manager managed.

Method 1 - By using AWS Systems Manager

To use this method, make sure that your Amazon EC2 instances are AWS Systems Manager managed and then make install the agent.

AWS Systems Manager managed Amazon EC2 instance

Use the following steps to make your Amazon EC2 instances AWS Systems Manager managed.

- AWS Systems Manager helps you manage your AWS applications and resources end-to-end and enable secure operations at scale.

  To manage your Amazon EC2 instances with AWS Systems Manager, see Setting up Systems Manager for Amazon EC2 instances in the *AWS Systems Manager User Guide*.
- The following table shows the new GuardDuty managed AWS Systems Manager documents:

Document name	Document type	Purpose
`AmazonGuardDuty-RuntimeMonitoringSsmPlugin`	Distributor	To package the GuardDuty security agent.

Document name	Document type	Purpose
AmazonGuardDuty-ConfigureRuntimeMonitoringSsmPlugin	Command	To run installation/un-installation script to install the GuardDuty security agent.

For more information about AWS Systems Manager, see Amazon EC2 Systems Manager Documents in the *AWS Systems Manager User Guide*.

To install the GuardDuty agent for Amazon EC2 instance by using AWS Systems Manager

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Documents**
3. In **Owned by Amazon**, choose `AmazonGuardDuty-ConfigureRuntimeMonitoringSsmPlugin`.
4. Choose **Run Command**.
5. Enter the following Run Command parameters
    - Action: Choose **Install**.
    - Installation Type: Choose **Install or Uninstall.**
    - Name: `AmazonGuardDuty-RuntimeMonitoringSsmPlugin`
    - Version: If this remains empty, you'll get latest version of the GuardDuty security agent. For more information about the release versions, GuardDuty security agent for Amazon EC2 instances.
6. Select the targeted Amazon EC2 instance. You can select one or more Amazon EC2 instances. For more information, see AWS Systems Manager Running commands from the console in the *AWS Systems Manager User Guide*
7. Validate if the GuardDuty agent installation is healthy. For more information, see Validating GuardDuty security agent installation status.

Method 2 - By using RPM installation scripts

> **Important**
>
> We strongly recommend verifying the GuardDuty security agent RPM signature before installing it on your machine.

1. Verify the GuardDuty security agent RPM signature

   a. **Download the appropriate public key, signature of x86_64 RPM, signature of arm64 RPM, and the corresponding access link to the RPM scripts hosted in Amazon S3 buckets**

      You can use the following templates to form the public key, signature of x86_64 RPM, signature of arm64 RPM, and the corresponding access link to the RPM scripts. Replace the value of the AWS Region, AWS account ID, and the GuardDuty agent version to access the RPM scripts.

      - **Public key**:

        ```
        s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/publickey.pem
        ```

      - **GuardDuty security agent RPM signature**:

        Signature of x86_64 RPM

        ```
        s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/x86_64/
        amazon-guardduty-agent-1.1.0.x86_64.sig
        ```

        Signature of arm64 RPM

        ```
        s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/arm64/
        amazon-guardduty-agent-1.1.0.arm64.sig
        ```

      - **Access links to the RPM scripts in Amazon S3 bucket**:

        Access link for x86_64 RPM

        ```
        s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/x86_64/
        amazon-guardduty-agent-1.1.0.x86_64.rpm
        ```

Access link for arm64 RPM

```
s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/arm64/
amazon-guardduty-agent-1.1.0.arm64.rpm
```

In the following command to download appropriate public key, signature of x86_64 RPM, signature of arm64 RPM, and the corresponding access link to the RPM scripts hosted in Amazon S3 buckets, make sure to replace the account ID with the appropriate AWS account ID and the Region with your current Region.

```
aws s3 cp s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/
x86_64/amazon-guardduty-agent-1.1.0.x86_64.rpm ./amazon-guardduty-
agent-1.1.0.x86_64.rpm
aws s3 cp s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/
x86_64/amazon-guardduty-agent-1.1.0.x86_64.sig ./amazon-guardduty-
agent-1.1.0.x86_64.sig
aws s3 cp s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/
publickey.pem ./publickey.pem
```

AWS Region	Region name	AWS account ID
`eu-west-1`	Europe (Ireland)	694911143906
`us-east-1`	US East (N. Virginia)	593207742271
`us-west-2`	US West (Oregon)	733349766148
`eu-west-3`	Europe (Paris)	665651866788
`us-east-2`	US East (Ohio)	307168627858
`eu-central-1`	Europe (Frankfurt)	323658145986
`ap-northeast-2`	Asia Pacific (Seoul)	914738172881
`eu-north-1`	Europe (Stockholm)	591436053604
`ap-east-1`	Asia Pacific (Hong Kong)	258348409381

`me-south-1`	Middle East (Bahrain)	536382113932
`eu-west-2`	Europe (London)	892757235363
`ap-northeast-1`	Asia Pacific (Tokyo)	533107202818
`ap-southeast-1`	Asia Pacific (Singapore)	174946120834
`ap-south-1`	Asia Pacific (Mumbai)	251508486986
`ap-southeast-3`	Asia Pacific (Jakarta)	510637619217
`sa-east-1`	South America (São Paulo)	758426053663
`ap-northeast-3`	Asia Pacific (Osaka)	273192626886
`eu-south-1`	Europe (Milan)	266869475730
`af-south-1`	Africa (Cape Town)	197869348890
`ap-southeast-2`	Asia Pacific (Sydney)	005257825471
`me-central-1`	Middle East (UAE)	000014521398
`us-west-1`	US West (N. California)	684579721401
`ca-central-1`	Canada (Central)	354763396469
`ap-south-2`	Asia Pacific (Hyderabad)	950823858135
`eu-south-2`	Europe (Spain)	919611009337
`eu-central-2`	Europe (Zurich)	529164026651
`ap-southeast-4`	Asia Pacific (Melbourne)	251357961535
`il-central-1`	Israel (Tel Aviv)	870907303882

b. Import the public key to the database

```
gpg --import publickey.pem
```

gpg shows import successfully

```
gpg: key 093FF49D: public key "AwsGuardDuty" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

c. Verify the signature

```
gpg --verify amazon-guardduty-agent-1.1.0.x86_64.sig amazon-guardduty-
agent-1.1.0.x86_64.rpm
```

If verification passes, you will see a message similar to the result below. You can now proceed to install the GuardDuty security agent using RPM.

Example output:

```
gpg: Signature made Fri 17 Nov 2023 07:58:11 PM UTC using ? key ID 093FF49D
gpg: Good signature from "AwsGuardDuty"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 7478 91EF 5378 1334 4456  7603 06C9 06A7 093F F49D
```

If verification fails, it means the signature on RPM has been potentially tampered. You must remove the public key from the database and retry the verification process.

Example:

```
gpg: Signature made Fri 17 Nov 2023 07:58:11 PM UTC using ? key ID 093FF49D
gpg: BAD signature from "AwsGuardDuty"
```

d. Remove the public key from the database.

```
gpg --delete-keys AwsGuardDuty
```

2. Connect with SSH from Linux or macOS.

3. Install the GuardDuty security agent by using the following command:

```
sudo rpm -ivh amazon-guardduty-agent-1.1.0.x86_64.rpm
```

4. Validate if the GuardDuty agent installation is healthy. For more information about the steps, see Validating GuardDuty security agent installation status.
5. (Optional) remove the GuardDuty security agent by using the following command:

   ```
   sudo rpm -ev amazon-guardduty-agent
   ```

Out of memory error

If you experience an `out-of-memory` error while installing or updating the GuardDuty security agent for Amazon EC2 manually, see Troubleshooting out of memory error.

Validating GuardDuty security agent installation status

To validate if the GuardDuty security agent is healthy

1. Connect with SSH from Linux or macOS.
2. Run the following command to check the status of the GuardDuty security agent:

   ```
   sudo systemctl status amazon-guardduty-agent
   ```

If you want to view the security agent installation logs, they are available under `/var/log/amzn-guardduty-agent/`.

To view the logs, do `sudo journalctl -u amazon-guardduty-agent`.

Updating the GuardDuty security agent manually

You can update the GuardDuty security agent by using the **Run command**. You can follow the same steps that you used to install the GuardDuty security agent.

Uninstalling security agent manually

When you disable Runtime Monitoring, GuardDuty doesn't remove the security agent associated with your Amazon EC2 instance. You can uninstall the GuardDuty security agent for Amazon EC2 instance by using one of the following two methods.

Method 1 - By using the Run command

To uninstall the GuardDuty security agent by using Run command

1. You can uninstall the GuardDuty security agent by following the steps as specified in AWS Systems Manager Run Command in the *AWS Systems Manager User Guide*. Use the Uninstall action in the parameters to uninstall the GuardDuty security agent.

   In the **Targets** section, make sure that the impact is only on those Amazon EC2 instances from which you want to uninstall the security agent.

   Use the following GuardDuty document and distributor:

   - Document name: `AmazonGuardDuty-ConfigureRuntimeMonitoringSsmPlugin`
   - Distributor: `AmazonGuardDuty-RuntimeMonitoringSsmPlugin`

2. After providing all the details, when you choose **Run**, the security agent that it deployed on the targeted Amazon EC2 instances is removed.

   To remove the Amazon VPC endpoint configuration, you must disable both Runtime Monitoring and Amazon EKS Runtime Monitoring.

Method 2 - By using the RPM script

To uninstall the GuardDuty security agent by using the rpm

1. Connect with SSH from Linux or macOS.
2. The following command will uninstall the GuardDuty security agent from the Amazon EC2 instance to which you connect:

   ```
   sudo rpm -e amazon-guardduty-agent
   ```

   You can also check the logs associated with this command.

Delete the Amazon VPC endpoint

When you want to disable Runtime Monitoring or uninstall the GuardDuty security agent for your account, you can also choose to delete the Amazon VPC endpoint that was created manually (Creating Amazon VPC endpoint manually).

To delete the Amazon VPC endpoint by using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**.
3. Select the endpoint that was created manually at the time of enabling Runtime Monitoring.
4. Choose **Actions**, **Delete VPC endpoints**.
5. When prompted for confirmation, enter **`delete`**.
6. Choose **Delete**.

To delete the Amazon VPC endpoint by using AWS CLI

- delete-vpc-endpoints (AWS Command Line Interface)
- Remove-EC2VpcEndpoint Cmdlet (Tools for Windows PowerShell)

Managing automated security agent for Fargate (Amazon ECS only)

Configuring GuardDuty agent for a standalone account

Presently, Runtime Monitoring supports managing the security agent for your Amazon ECS clusters (AWS Fargate) only through GuardDuty. There is no support for managing the security agent manually on Amazon ECS clusters.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Runtime Monitoring**.
3. Under the **Configuration** tab:

   a. **To manage Automated agent configuration for all Amazon ECS clusters (account level)**

      Choose **Enable** in the **Automated agent configuration** section for **AWS Fargate (ECS only)**. When a new Fargate Amazon ECS task launches, GuardDuty will manage the deployment of the security agent.

      - Choose **Save**.

b. **To manage Automated agent configuration by excluding some of the Amazon ECS clusters (cluster level)**

   i. Add a tag to the Amazon ECS cluster for which you want to exclude all of the tasks. The key-value pair must be `GuardDutyManaged`-`false`.

   ii. Prevent modification of these tags, except by trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles]() in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged":
 "${aws:PrincipalTag/GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged":
 "${aws:PrincipalTag/GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

iii. Under the **Configuration** tab, choose **Enable** in the **Automated agent configuration** section.

> **Note**
>
> Always add the exclusion tag to your Amazon ECS cluster before enabling GuardDuty agent auto-management for your account; otherwise, the security agent will be deployed in all the tasks that are launched within the corresponding Amazon ECS cluster.

For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.

iv. Choose **Save**.

c. **To manage Automated agent configuration by including some of the Amazon ECS clusters (cluster level)**

i. Add a tag to an Amazon ECS cluster for which you want to include all of the tasks. The key-value pair must be `GuardDutyManaged`-`true`.

ii. Prevent modification of these tags, except by trusted entities. The policy provided in Prevent tags from being modified except by authorized principles in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged":
 "${aws:PrincipalTag/GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
```

```
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged":
 "${aws:PrincipalTag/GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
org-admins/iam-admin"
```

```
                    },
                    "Null": {
                        "aws:PrincipalTag/GuardDutyManaged": true
                    }
                }
            }
        ]
}
```

Configuring GuardDuty agent for multi-account environment

In a multiple-account environment, only the delegated GuardDuty administrator account can enable or disable Automated agent configuration for the member accounts, and manage automated agent configuration for Amazon ECS clusters that belong to the member accounts in their organization. A GuardDuty member account can't modify this configuration. The delegated GuardDuty administrator account manages their member accounts using AWS Organizations. For more information about multi-account environments, see Managing multiple accounts in GuardDuty.

Enabling automated agent configuration for delegated GuardDuty administrator account

Manage for all Amazon ECS clusters (account level)

If you chose **Enable for all accounts** for Runtime Monitoring, then you have the following options:

- Choose **Enable for all accounts** in the Automated agent configuration section. GuardDuty will deploy and manage the security agent for all the Amazon ECS tasks that get launched.
- Choose **Configure accounts manually**.

If you chose **Configure accounts manually** in the Runtime Monitoring section, then do the following:

1. Choose **Configure accounts manually** in the Automated agent configuration section.
2. Choose **Enable** in the **delegated GuardDuty administrator account (this account)** section.

Choose **Save**.

Manage for all Amazon ECS clusters but exclude some of the clusters (cluster level)

1. Add a tag to this Amazon ECS cluster with the key-value pair as `GuardDutyManaged`-`false`.
2. Prevent modification of tags, except by the trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles]() in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

4. In the navigation pane, choose **Runtime Monitoring**.

5. **Note**

   Always add the exclusion tag to your Amazon ECS clusters before enabling Automated agent configuration for your account; otherwise the GuardDuty sidecar container will be attached to all the containers in the Amazon ECS tasks that get launched.

   Under the **Configuration** tab, choose **Enable** in the **Automated agent configuration**.

   For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.

6. Choose **Save**.

Manage for selective (inclusion only) Amazon ECS clusters (cluster level)

1. Add a tag to an Amazon ECS cluster for which you want to include all of the tasks. The key-value pair must be `GuardDutyManaged`-`true`.
2. Prevent modification of these tags, except by trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles]() in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

> **Note**
>
> When using inclusion tags for your Amazon ECS clusters, you don't need to enable GuardDuty agent through automated agent congifuration explicitly.

Auto-enable for all member accounts

Manage for all Amazon ECS clusters (account level)

The following steps assume that you chose **Enable for all accounts** in the Runtime Monitoring section.

1. Choose **Enable for all accounts** in the Automated agent configuration section. GuardDuty will deploy and manage the security agent for all the Amazon ECS tasks that get launched.
2. Choose **Save**.

Manage for all Amazon ECS clusters but exclude some of the clusters (cluster level)

1. Add a tag to this Amazon ECS cluster with the key-value pair as `GuardDutyManaged`-`false`.
2. Prevent modification of tags, except by the trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles](#) in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

4. In the navigation pane, choose **Runtime Monitoring**.

5. > **Note**
   >
   > Always add the exclusion tag to your Amazon ECS clusters before enabling Automated agent configuration for your account; otherwise the GuardDuty sidecar container will be attached to all the containers in the Amazon ECS tasks that get launched.

   Under the **Configuration** tab, choose **Edit**.

6. Choose **Enable for all accounts** in the **Automated agent configuration** section

   For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.

7. Choose **Save**.

Manage for selective (inclusion-only) Amazon ECS clusters (cluster level)

Regardless of how you choose to enable Runtime Monitoring, the following steps will help you monitor selective Amazon ECS Fargate tasks for all of the member accounts in your organization.

1. Do not enable any configuration in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as you selected in the previous step.
2. Choose **Save**.
3. Prevent modification of these tags, except by trusted entities. The policy provided in Prevent tags from being modified except by authorized principles in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
```

```
}
```

Note

When using inclusion tags for your Amazon ECS clusters, you don't need to enable **GuardDuty agent auto-management** explicitly.

Enabling automated agent configuration for existing active member accounts

Manage for all Amazon ECS clusters (account level)

1. On the Runtime Monitoring page, under the **Configuration** tab, you can view the current status of Automated agent configuration.
2. Within the Automated agent configuration pane, under the **Active member accounts** section, choose **Actions**.
3. From **Actions**, choose **Enable for all existing active member accounts**.
4. Choose **Confirm**.

Manage for all Amazon ECS clusters but exclude some of the clusters (cluster level)

1. Add a tag to this Amazon ECS cluster with the key-value pair as `GuardDutyManaged`-`false`.
2. Prevent modification of tags, except by the trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles]() in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
4. In the navigation pane, choose **Runtime Monitoring**.
5. > **Note**
   >
   > Always add the exclusion tag to your Amazon ECS clusters before enabling Automated agent configuration for your account; otherwise the GuardDuty sidecar container will be attached to all the containers in the Amazon ECS tasks that get launched.

   Under the **Configuration** tab, in the Automated agent configuration section, under **Active member accounts**, choose **Actions**.
6. From **Actions**, choose **Enable for all active member accounts**.

   For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.
7. Choose **Confirm**.

Manage for selective (inclusion only) Amazon ECS clusters (cluster level)

1. Add a tag to an Amazon ECS cluster for which you want to include all of the tasks. The key-value pair must be `GuardDutyManaged`-`true`.

2. Prevent modification of these tags, except by trusted entities. The policy provided in Prevent tags from being modified except by authorized principles in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

Note

When using inclusion tags for your Amazon ECS clusters, you don't need to enable **Automated agent configuration** explicitly.

Auto-enable Automated agent configuration for new members

Manage for all Amazon ECS clusters (account level)

1. On the Runtime Monitoring page, choose **Edit** to update the existing configuration.
2. In the Automated agent configuration section, select **Automatically enable for new member accounts**.
3. Choose **Save**.

Manage for all Amazon ECS clusters but exclude some of the clusters (cluster level)

1. Add a tag to this Amazon ECS cluster with the key-value pair as `GuardDutyManaged`-`false`.
2. Prevent modification of tags, except by the trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles](#) in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

4. In the navigation pane, choose **Runtime Monitoring**.

5. > **Note**
   >
   > Always add the exclusion tag to your Amazon ECS clusters before enabling Automated agent configuration for your account; otherwise the GuardDuty sidecar container will be attached to all the containers in the Amazon ECS tasks that get launched.

   Under the **Configuration** tab, select **Automatically enable for new member accounts** in the **Automated agent configuration** section.

   For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.

6. Choose **Save**.

Manage for selective (inclusion only) Amazon ECS clusters (cluster level)

1. Add a tag to an Amazon ECS cluster for which you want to include all of the tasks. The key-value pair must be `GuardDutyManaged`-`true`.

2. Prevent modification of these tags, except by trusted entities. The policy provided in [Prevent tags from being modified except by authorized principles]() in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

> **Note**
>
> When using inclusion tags for your Amazon ECS clusters, you don't need to enable **Automated agent configuration** explicitly.

Enabling Automated agent configuration for active member accounts selectively

Manage for all Amazon ECS (account level)

1. On the Accounts page, select the accounts for which you want to enable Runtime Monitoring-Automated agent configuration (ECS-Fargate). You can select multiple accounts. Make sure that the accounts that you select in this step are already enabled with Runtime Monitoring.
2. From **Edit protection plans**, choose the appropriate option to enable **Runtime Monitoring-Automated agent configuration (ECS-Fargate)**.
3. Choose **Confirm**.

Manage for all Amazon ECS clusters but exclude some of the clusters (cluster level)

1. Add a tag to this Amazon ECS cluster with the key-value pair as `GuardDutyManaged`-`false`.
2. Prevent modification of tags, except by the trusted entities. The policy provided in Prevent tags from being modified except by authorized principles in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

4. In the navigation pane, choose **Runtime Monitoring**.

5.
> **Note**
>
> Always add the exclusion tag to your Amazon ECS clusters before enabling GuardDuty agent auto-management for your account; otherwise the GuardDuty sidecar container will be attached to all the containers in the Amazon ECS tasks that get launched.

   On the Accounts page, select the accounts for which you want to enable Runtime Monitoring-Automated agent configuration (ECS-Fargate). You can select multiple accounts. Make sure that the accounts that you select in this step are already enabled with Runtime Monitoring.

   For the Amazon ECS clusters that have not been excluded, GuardDuty will manage the deployment of the security agent in the sidecar container.

6. From **Edit protection plans**, choose the appropriate option to enable **Runtime Monitoring-Automated agent configuration (ECS-Fargate)**.
7. Choose **Save**.

Manage for selective (inclusion only) Amazon ECS clusters (cluster level)

1. Make sure you don't enable **Automated agent configuration** (or **Runtime Monitoring-Automated agent configuration (ECS-Fargate)**) for the selected accounts that have the Amazon ECS clusters that you want to monitor.
2. Add a tag to an Amazon ECS cluster for which you want to include all of the tasks. The key-value pair must be `GuardDutyManaged`-`true`.
3. Prevent modification of these tags, except by trusted entities. The policy provided in Prevent tags from being modified except by authorized principles in the *AWS Organizations User Guide* has been modified to be applicable here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ecs:ResourceTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "ecs:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/GuardDutyManaged": "${aws:PrincipalTag/
GuardDutyManaged}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ecs:CreateTags",
                "ecs:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-
admins/iam-admin"
                },
                "Null": {
                    "aws:PrincipalTag/GuardDutyManaged": true
                }
            }
        }
    ]
}
```

> **Note**
>
> When using inclusion tags for your Amazon ECS clusters, you don't need to enable **Automated agent configuration** explicitly.

Managing security agent automatically for Amazon EKS clusters

Configuring Automated agent for standalone account

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Runtime Monitoring**.
3. Under the **Configuration** tab, choose **Enable** to enable automated agent configuration for your account.

Preferred approach to deploy GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	1. Choose **Enable** in the **Automated agent configuration** section. GuardDuty will manage the deployment of and updates to the security agent for all the existing and potentially new EKS clusters in your account. 2. Choose **Save**.

Preferred approach to deploy GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tag)	From the following procedures, choose one of the scenarios that is applicable to you. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource`. - Replace *`ec2:DeleteTags`* with `eks:UntagResource`. - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn`: `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:`

Preferred approach to deploy GuardDuty security agent	Steps
	`role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/. 4. In the navigation pane, choose **Runtime Monitoring**. **Note** Always add the exclusion tag to your EKS clusters before enabling GuardDuty agent auto-management for your account; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. 5. Under the **Configuration** tab, choose **Enable** in the **GuardDuty agent management** section. For the EKS clusters that have not been excluded from monitoring, GuardDuty will manage the deployment of and updates to the GuardDuty security agent. 6. Choose **Save**. **To exclude an EKS cluster from monitoring after the GuardDuty security agent has already been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`.

Preferred approach to deploy GuardDuty security agent	Steps
	For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. After this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on receiving the runtime events from this EKS cluster. This may impact your usage statistics. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`

Preferred approach to deploy GuardDuty security agent	Steps
	3. To stop receiving the runtime events from this cluster, you must remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources.

Preferred approach to deploy GuardDuty security agent	Steps
Monitor selective EKS clusters using inclusion tags	1. Make sure to choose **Disable** in the **Automated agent configuration** section. Keep Runtime Monitoring enabled. 2. Choose **Save** 3. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `true`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 4. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` :

Preferred approach to deploy GuardDuty security agent	Steps
	`"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`
Manage agent manually	1. Make sure to choose **Disable** in the **Automated agent configuration** section. Keep Runtime Monitoring enabled. 2. Choose **Save**. 3. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Configuring Automated agent for multi-account environments

In a multiple-account environments, only the delegated GuardDuty administrator account can enable or disable Automated agent configuration for the member accounts, and manage Automated agent for the EKS clusters belonging to the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account account manages their member accounts using AWS Organizations. For more information about multi-account environments, see Managing multiple accounts.

Configuring Automated agent configuration for delegated GuardDuty administrator account

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	If you chose **Enable for all accounts** in the Runtime Monitoring section, then you have the following options:

Preferred approach to manage GuardDuty security agent	Steps
	• Choose **Enable for all accounts** in the Automated agent configuration section. GuardDuty will deploy and manage the security agent for all the EKS clusters that belong to the delegated GuardDuty administrator account account and also for all the EKS clusters that belong to all the existing and potentially new member accounts in the organization. • Choose **Configure accounts manually**. If you chose **Configure accounts manually** in the Runtime Monitoring section, then do the following: 1. Choose **Configure accounts manually** in the Automated agent configuration section. 2. Choose **Enable** in the **delegated GuardDuty administrator account (this account)** section. Choose **Save**.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tags)	From the following procedures, choose one of the scenarios that apply to you. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: • Replace *`ec2:CreateTags`* with `eks:TagResource` . • Replace *`ec2:DeleteTags`* with `eks:UntagResource` . • Replace *`access-project`* with `GuardDutyManaged` • Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/. 4. In the navigation pane, choose **Runtime Monitoring**.

Preferred approach to manage GuardDuty security agent	Steps
	Note Always add the exclusion tag to your EKS clusters before enabling GuardDuty agent auto-management for your account; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. 5. Under the **Configuration** tab, choose **Enable** in the **GuardDuty agent management** section. For the EKS clusters that have not been excluded from monitoring, GuardDuty will manage the deployment of and updates to the GuardDuty security agent. 6. Choose **Save**. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: • Replace *`ec2:CreateTags`* with `eks:TagResource`. • Replace *`ec2:DeleteTags`* with `eks:UntagResource`.

Preferred approach to manage GuardDuty security agent	Steps
	- Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. If you had automated agent enabled for this EKS cluster, then after this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on receiving the runtime events from this EKS cluster. This may impact your usage statistics. To stop receiving the runtime events from this cluster, you must remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources 4. If you were managing the GuardDuty security agent for this EKS cluster manually, then see Cleaning up GuardDuty security agent resources.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters using inclusion tags	Regardless of how you chose to enable Runtime Monitoring, the following steps will help you monitor selective EKS clusters in your account: 1. Make sure to choose **Disable** for **delegated GuardDuty administrator account (this account)** in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. Add a tag to your EKS cluster with the key as `GuardDutyManaged` and its value as `true`. For more information about tagging your Amazon EKS cluster, see [Working with tags using the console]() in the Amazon EKS User Guide. GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 4. To prevent modification of tags, except by the trusted entities, use the policy provided in [Prevent tags from being modified except by authorized principals]() in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` :

Preferred approach to manage GuardDuty security agent	Steps
	`"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`
Manage the GuardDuty security agent manually	Regardless of how you chose to enable Runtime Monitoring, you can manage the security agent manually for your EKS clusters. 1. Make sure to choose **Disable** for **delegated GuardDuty administrator account (this account)** in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Auto-enable Automated agent for all member accounts

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	This topic is to enable Runtime Monitoring for all member accounts and therefore, the following steps assume that you must have chosen **Enable for all accounts** in the Runtime Monitoring section.

Preferred approach to manage GuardDuty security agent	Steps
	1. Choose **Enable for all accounts** in the Automated agent configuration section. GuardDuty will deploy and manage the security agent for all the EKS clusters that belong to the delegated GuardDuty administrator account account and also for all the EKS clusters that belong to all the existing and potentially new member accounts in the organization. 2. Choose **Save**.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tags)	From the following procedures, choose one of the scenarios that apply to you. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: • Replace *`ec2:CreateTags`* with `eks:TagResource` . • Replace *`ec2:DeleteTags`* with `eks:UntagResource` . • Replace *`access-project`* with `GuardDutyManaged` • Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/. 4. In the navigation pane, choose **Runtime Monitoring**.

Preferred approach to manage GuardDuty security agent	Steps
	Note Always add the exclusion tag to your EKS clusters before enabling Automated agent for your account; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. 5. Under the **Configuration** tab, choose **Edit** in the **Runtime Monitoring configuration** section. 6. Choose **Enable for all accounts** in the Automated agent configuration section. For the EKS clusters that have not been excluded from monitoring, GuardDuty will manage the deployment of and updates to the GuardDuty security agent. 7. Choose **Save**. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. If you had Automated agent configuration enabled for this EKS cluster, then after this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on receiving the runtime events from this EKS cluster. This may impact your usage statistics.

Preferred approach to manage GuardDuty security agent	Steps
	To stop receiving the runtime events from this cluster, you must remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources 3. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 4. If you were managing the GuardDuty security agent for this EKS cluster manually, then see Cleaning up GuardDuty security agent resources.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters using inclusion tags	Regardless of how you chose to enable Runtime Monitoring, the following steps will help you monitor selective EKS clusters for all member accounts in your organization: 1. Do not enable any configuration in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. Add a tag to your EKS cluster with the key as `GuardDutyManaged` and its value as `true`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 4. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: • Replace *`ec2:CreateTags`* with `eks:TagResource` . • Replace *`ec2:DeleteTags`* with `eks:UntagResource` . • Replace *`access-project`* with `GuardDutyManaged` • Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` :

Preferred approach to manage GuardDuty security agent	Steps
	`"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`
Manage the GuardDuty security agent manually	Regardless of how you chose to enable Runtime Monitoring, you can manage the security agent manually for your EKS clusters. 1. Do not enable any configuration in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Enabling automated agent for all existing active member accounts

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

To manage GuardDuty security agent for existing active member accounts in your organization

- For GuardDuty to receive the runtime events from the EKS clusters that belong to the existing active member accounts in the organization, you must choose a preferred approach to manage the GuardDuty security agent for these EKS clusters. For more information about each of these approaches, see Approaches to manage GuardDuty security agent.

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	**To monitor all EKS clusters for all existing active member accounts** 1. On the Runtime Monitoring page, under the **Configuration** tab, you can view the current status of Automated agent configuration. 2. Within the **Automated agent configuration** pane, under the **Active member accounts** section, choose **Actions**. 3. From **Actions**, choose **Enable for all existing active member accounts**. 4. Choose **Confirm**.

Preferred approach to manage GuardDuty security agent	**Steps**
Monitor all EKS clusters but exclude some of them (using exclusion tag)	From the following procedures, choose one of the scenarios that apply to you. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see [Working with tags using the console]() in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in [Prevent tags from being modified except by authorized principals]() in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:`

Preferred approach to manage GuardDuty security agent	Steps
	`role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/. 4. In the navigation pane, choose **Runtime Monitoring**. **Note** Always add the exclusion tag to your EKS clusters before enabling Automated agent configuration for your account; otherwise , the GuardDuty security agent will be deployed on all the EKS clusters in your account. 5. Under the **Configuration** tab, in the **Automated agent configuration** pane, under **Active member accounts**, choose **Actions**. 6. From **Actions**, choose **Enable for all active member accounts**. 7. Choose **Confirm**. **To exclude an EKS cluster from monitoring after the GuardDuty security agent has already been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide.

Preferred approach to manage GuardDuty security agent	Steps
	After this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on receiving the runtime events from this EKS cluster. This may impact your usage statistics. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Regardless of how you manage the security agent (through GuardDuty or manually), to stop receiving the runtime events from this cluster, you must

Preferred approach to manage GuardDuty security agent	Steps
	remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters using inclusion tags	1. On the Accounts page, after you enable Runtime Monitoring, do not enable **Runtime Monitoring - Automated agent configuration**. 2. Add a tag to the EKS cluster that belongs to the selected account that you want to monitor. The key-value pair of the tag must be `GuardDuty Managed` `-true`. For more information about tagging your Amazon EKS cluster, see [Working with tags using the console]() in the Amazon EKS User Guide. GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 3. To prevent modification of tags, except by the trusted entities, use the policy provided in [Prevent tags from being modified except by authorized principals]() in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDuty Managed` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` :

Preferred approach to manage GuardDuty security agent	**Steps**

	`"aws:PrincipalArn":["arn:aws:iam::12`
	`3456789012:role/org-admins/iam-`
	`admin", "arn:aws:iam::123456789012:`
	`role/org-admins/iam-admin", "arn:aws:`
	`iam::123456789012:role/org-admins/ia`
	`m-admin"]`

Manage the GuardDuty security agent manually	1. Make sure you don't choose **Enable** in the **Automated agent configuration** section. Keep Runtime Monitoring enabled. 2. Choose **Save**. 3. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Auto-enable automated agent configuration for new members

Preferred approach to manage GuardDuty security agent	**Steps**
Manage security agent through GuardDuty (Monitor all EKS clusters)	1. On the Runtime Monitoring page, choose **Edit** to update the existing configuration. 2. In the Automated agent configuration section, select **Automatically enable for new member accounts**. 3. Choose **Save**.
Monitor all EKS clusters but exclude some of them (using exclusion tags)	From the following procedures, choose one of the scenarios that apply to you.

Preferred approach to manage GuardDuty security agent	Steps
	To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource`. - Replace *`ec2:DeleteTags`* with `eks:UntagResource`. - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn`: `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`

Preferred approach to manage GuardDuty security agent	Steps
	3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/. 4. In the navigation pane, choose **Runtime Monitoring**. **Note** Always add the exclusion tag to your EKS clusters before enabling Automated agent configuration for your account; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. 5. Under the **Configuration** tab, select **Automatically enable for new member accounts** in the **GuardDuty agent management** section. For the EKS clusters that have not been excluded from monitoring, GuardDuty will manage the deployment of and updates to the GuardDuty security agent. 6. Choose **Save**. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has been deployed on this cluster** 1. Regardless of whether you manage the GuardDuty security agent through GuardDuty or manually, add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide.

Preferred approach to manage GuardDuty security agent	Steps
	If you had Automated agent enabled for this EKS cluster, then after this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on receiving the runtime events from this EKS cluster. This may impact your usage statistics. To stop receiving the runtime events from this cluster, you must remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-`

Preferred approach to manage GuardDuty security agent	Steps
	`admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. If you were managing the GuardDuty security agent for this EKS cluster manually, then see Cleaning up GuardDuty security agent resources.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters using inclusion tags	Regardless of how you chose to enable Runtime Monitoring, the following steps will help you monitor selective EKS clusters for the new member accounts in your organization. 1. Make sure to clear **Automatically enable for new member accounts** in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. Add a tag to your EKS cluster with the key as `GuardDutyManaged` and its value as `true`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 4. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged`

Preferred approach to manage GuardDuty security agent	Steps
	- Replace *123456789012* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `Principal Arn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]`
Manage the GuardDuty security agent manually	Regardless of how you chose to enable Runtime Monitoring, you can manage the security agent manually for your EKS clusters. 1. Make sure clear the checkbox **Automatically enable for new member accounts** in the Automated agent configuration section. Keep the Runtime Monitoring configuration the same as configured in the previous step. 2. Choose **Save**. 3. To manage the security agent, see [Managing security agent manually for Amazon EKS cluster]().

Configuring Automated agent for active member accounts selectively

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty	1. On the Accounts page, select the accounts for which you want to enable Automated agent configuration. You can select

Preferred approach to manage GuardDuty security agent	Steps
(Monitor all EKS clusters)	more than one account at a time. Make sure that the accounts you select in this step already have EKS Runtime Monitoring enabled. 2. From **Edit Protection plans** choose the appropriate option to enable **Runtime Monitoring - Automated agent configuration**. 3. Choose **Confirm**.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tags)	From the following procedures, choose one of the scenarios that apply to you. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has not been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDutyManaged` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource`. - Replace *`ec2:DeleteTags`* with `eks:UntagResource`. - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn`: `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

Preferred approach to manage GuardDuty security agent	Steps
	Note Always add the exclusion tag to your EKS clusters before enabling Automated agent configuration for your account; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. 4. On the Accounts page, select the account for which you want to enable **Manage agent automatically**. You can select more than one account at a time. 5. From **Edit protection plans**, choose the appropriate option to enable **Runtime Monitoring-Automated agent configuration** for the selected account. For the EKS clusters that have not been excluded from monitoring, GuardDuty will manage the deployment of and updates to the GuardDuty security agent. 6. Choose **Save**. **To exclude an EKS cluster from monitoring when the GuardDuty security agent has been deployed on this cluster** 1. Add a tag to this EKS cluster with the key as `GuardDuty Managed` and its value as `false`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. If you had previously Automated agent configuration enabled for this EKS cluster, then after this step, GuardDuty will not update the security agent for this cluster. However, the security agent will remain deployed and GuardDuty will keep on

Preferred approach to manage GuardDuty security agent	Steps
	receiving the runtime events from this EKS cluster. This may impact your usage statistics. To stop receiving the runtime events from this cluster, you must remove the deployed security agent from this EKS cluster. For more information about removing the deployed security agent, see Cleaning up GuardDuty security agent resources 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *ec2:CreateTags* with `eks:TagResource` . - Replace *ec2:DeleteTags* with `eks:UntagResource` . - Replace *access-project* with `GuardDutyManaged` - Replace *123456789012* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. If you were managing the GuardDuty security agent for this EKS cluster manually, you must remove it. For more information, see Cleaning up GuardDuty security agent resources.

Preferred approach to manage GuardDuty security agent	**Steps**
Monitor selective EKS clusters using inclusion tags	Regardless of how you chose to enable Runtime Monitoring, the following steps will help you monitor selective EKS clusters that belong to the selected accounts: 1. Make sure that you do not enable **Runtime Monitoring- Automated agent configuration** for the selected accounts that have the EKS clusters that you want to monitor. 2. Add a tag to your EKS cluster with the key as `GuardDuty Managed` and its value as `true`. For more information about tagging your Amazon EKS cluster, see Working with tags using the console in the Amazon EKS User Guide. After adding the tag, GuardDuty will manage the deployment of and updates to the security agent for the selective EKS clusters that you want to monitor. 3. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::1234`

Preferred approach to manage GuardDuty security agent	Steps

	`56789012:role/org-admins/iam-admin", "arn:aws:`
	`iam::123456789012:role/org-admins/iam-admin"]`

Manage the GuardDuty security agent manually	1. Keep the Runtime Monitoring configuration the same as configured in the previous step. Make sure that you don't enable **Runtime Monitoring- Automated agent configuration** for any of the selected accounts. 2. Choose **Confirm**. 3. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Managing security agent manually for Amazon EKS cluster

This section describes how you can manage your Amazon EKS add-on agent (GuardDuty agent) after you enable Runtime Monitoring. To use Runtime Monitoring, you must enable Runtime Monitoring and configure the Amazon EKS add-on, `aws-guardduty-agent`. Performing only one of these two steps will not help GuardDuty detect potential threats or generate findings.

Prerequisites to deploying GuardDuty security agent

This section describes the prerequisites to deploying GuardDuty security agent for your EKS clusters manually. Before proceeding, make sure you have already configured Runtime Monitoring for your accounts. The GuardDuty security agent (EKS add-on) will not work if you don't configure Runtime Monitoring. For more information, see Enabling GuardDuty Runtime Monitoring. After you complete the following steps, see Deploying GuardDuty security agent.

Choose your preferred access method to create an Amazon VPC endpoint.

Console

Create VPC endpoint

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **Virtual private cloud**, choose **Endpoints**.

3. Choose **Create Endpoint**.

4. On the **Create endpoint** page, for **Service category**, choose **Other endpoint services**.

5. For **Service name**, enter **`com.amazonaws.`*`us-east-1`*`.guardduty-data`**.

   Make sure to replace *`us-east-1`* with the correct Region. This must be the same Region as the EKS cluster that belongs to your AWS account ID.

6. Choose **Verify service**.

7. After the service name is successfully verified, choose the **VPC** where your cluster resides. Add the following policy to restrict VPC endpoint usage to specified account only. With the organization `Condition` provided below this policy, you can update the following policy to restrict access to your endpoint. To provide VPC endpoint support to specific account IDs in your organization, see Organization condition to restrict access to your endpoint.

   ```
   {
    "Version": "2012-10-17",
    "Statement": [
     {
      "Action": "*",
      "Resource": "*",
      "Effect": "Allow",
      "Principal": "*"
     },
     {
      "Condition": {
       "StringNotEquals": {
        "aws:PrincipalAccount": "111122223333"
       }
      },
      "Action": "*",
      "Resource": "*",
      "Effect": "Deny",
      "Principal": "*"
     }
    ]
   }
   ```

   The `aws:PrincipalAccount` account ID must match the account containing the VPC and VPC endpoint. The following list shows how to share the VPC endpoint with other AWS account IDs:

Organization condition to restrict access to your endpoint

- To specify multiple accounts to access the VPC endpoint, replace `"aws:PrincipalAccount": "111122223333"` with the following:

  ```
  "aws:PrincipalAccount": [
            "666666666666",
            "555555555555"
      ]
  ```

- To allow all the members from an organization to access the VPC endpoint, replace `"aws:PrincipalAccount": "111122223333"` with the following:

  ```
  "aws:PrincipalOrgID": "o-abcdef0123"
  ```

- To restrict accessing a resource to an organization ID, add your `ResourceOrgID` to the policy.

  For more information, see ResourceOrgID.

  ```
  "aws:ResourceOrgID": "o-abcdef0123"
  ```

8. Under **Additional settings**, choose **Enable DNS name**.
9. Under **Subnets**, choose the subnets in which your cluster resides.
10. Under **Security groups**, choose a security group that has the in-bound port 443 enabled from your VPC (or your EKS cluster). If you don't already have a security group that has an in-bound port 443 enabled, Create a security group.

    If there is an issue while restricting the in-bound permissions to your VPC (or cluster), provide the support to in-bound 443 port from any IP address (`0.0.0.0/0`).

API/CLI

- Invoke CreateVpcEndpoint.
- Use the following values for the parameters:
  - For **Service name**, enter **`com.amazonaws.us-east-1.guardduty-data`**.

Make sure to replace *us-east-1* with the correct Region. This must be the same Region as the EKS cluster that belongs to your AWS account ID.

  - For DNSOptions, enable private DNS option by setting it to `true`.

- For AWS Command Line Interface, see create-vpc-endpoint.

Configure GuardDuty security agent (add-on) parameters for Amazon EKS

You can configure specific parameters of your GuardDuty security agent for Amazon EKS. This support is available for GuardDuty security agent version 1.5.0 and above. For information about latest add-on versions, see GuardDuty security agent for Amazon EKS clusters.

Why should I update the security agent configuration schema

> Configuration schema for the GuardDuty security agent is the same across all containers within your Amazon EKS clusters. When the default values do not align with the associated workloads and instance size, consider configuring the CPU settings, memory settings, `PriorityClass`, and `dnsPolicy` settings. Regardless of how you manage the GuardDuty agent for your Amazon EKS clusters, you can configure or update the existing configuration of these parameters.

Automated agent configuration behavior with configured parameters

When GuardDuty manages the security agent (EKS add-on) on your behalf, it updates the add-on, as needed. GuardDuty will set the value of the configurable parameters to a default value. However, you can still update the parameters to a desired value. If this leads to a conflict, the default option to resolveConflicts is None.

Configurable parameters and values

For information about the steps to configure the add-on parameters, see:

- Deploying GuardDuty security agent or
- Updating security agent manually

The following tables provide the ranges and values that you can use to deploy the Amazon EKS add-on manually or update the existing add-on settings.

CPU settings

Parameters	Default value	Configurable range
Requests	200m	Between 200m and 10000m, both inclusive
Limits	1000m	

Memory settings

Parameters	Default value	Configurable range
Requests	256Mi	Between 256Mi and 20000Mi, both inclusive
Limits	1024Mi	

`PriorityClass` settings

When GuardDuty creates an Amazon EKS add-on for you, the assigned `PriorityClass` is `aws-guardduty-agent.priorityclass`. This means that no action will be taken based on the priority of the agent pod. You can configure this add-on parameter by choosing one of the following `PriorityClass` options:

Configurable `PriorityClass`	`preemptionPolicy` value	`preemptionPolicy` description	Pod value
`aws-guardduty-agent.priorityclass`	`Never`	No action	1000000
`aws-guardduty-agent.priorityclass-high`	`PreemptLowerPriority`	Assigning this value will preempt a pod running with the priority value lower than the agent pod value.	100000000
`system-cluster-critical` [1]	`PreemptLowerPriority`		2000000000

Configurable PriorityClass	preemptionPolicy value	preemptionPolicy description	Pod value
`system-node-critical` [1]	`PreemptLowerPriority`		2000001000

[1] Kubernetes provides these two `PriorityClass` options – `system-cluster-critical` and `system-node-critical`. For more information, see [PriorityClass](#) in the *Kubernetes documentation*.

`dnsPolicy` settings

Choose one of the following DNS policy options that Kubernetes supports. When no configuration is specified, `ClusterFirst` is used as the default value.

- `ClusterFirst`
- `ClusterFirstWithHostNet`
- `Default`

For information about these policies, see [Pod's DNS Policy](#) in the *Kubernetes documentation*.

Deploying GuardDuty security agent

This section describes how you can deploy the GuardDuty security agent for the first time for specific EKS clusters. Before you proceed with this section, make sure you have already set up the prerequisites and enabled Runtime Monitoring for your accounts. The GuardDuty security agent (EKS add-on) will not work if you do not enable Runtime Monitoring.

Choose your preferred access method to deploy the GuardDuty security agent for the first time.

Console

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. Choose your **Cluster name**.
3. Choose the **Add-ons** tab.

4. Choose **Get more add-ons**.
5. On the **Select add-ons** page, choose **Amazon GuardDuty Runtime Monitoring**.
6. On the **Configure selected add-on settings** page, use the default settings. If the **Status** of your EKS add-on is **Requires activation**, choose **Activate GuardDuty**. This action will open the GuardDuty console to configure Runtime Monitoring for your accounts.
7. After you've configured Runtime Monitoring for your accounts, switch back to the Amazon EKS console. The **Status** of your EKS add-on should have changed to **Ready to install**.
8. **(Optional) Providing EKS add-on configuration schema**

   For the add-on **Version**, if you choose **v1.5.0** and above, Runtime Monitoring supports configuring specific parameters of the GuardDuty agent. For information about parameter ranges, see Configure EKS add-on parameters.

   a. Expand **Optional configuration settings** to view the configurable parameters and their expected value and format.
   b. Set the parameters. The values must be in the range provided in Configure EKS add-on parameters.
   c. Choose **Save changes** to create the add-on based on the advanced configuration.
   d. For **Conflict resolution method**, the option that you choose will be used to resolve a conflict when you update the value of a parameter to a non-default value. For more information about the listed options, see resolveConflicts in the *Amazon EKS API Reference*.
9. Choose **Next**.
10. On the **Review and create** page, verify all the details, and choose **Create**.
11. Navigate back to the cluster details and choose the **Resources** tab.
12. You can view the new pods with the prefix **aws-guardduty-agent**.

API/CLI

You can configure the Amazon EKS add-on agent (`aws-guardduty-agent`) using either of the following options:

- Run CreateAddon for your account.

- > **Note**
  >
  > For the add-on `version`, if you choose **v1.5.0** and above, Runtime Monitoring supports configuring specific parameters of the GuardDuty agent. For more information, see Configure EKS add-on parameters.

  Use the following values for the request parameters:

  - For `addonName`, enter `aws-guardduty-agent`.

    You can use the following AWS CLI example when using configurable values supported for addon versions v1.5.0 and above. Make sure to replace the placeholder values highlighted in red and the associated `Example.json` with the configured values.

    ```
    aws eks create-addon --region us-east-1 --cluster-name myClusterName --addon-name aws-guardduty-agent --addon-version v1.5.0-eksbuild.1 --configuration-values 'file://example.json'
    ```

    Example Example.json

    ```
    {
     "priorityClassName": "aws-guardduty-agent.priorityclass-high",
     "dnsPolicy": "Default",
     "resources": {
      "requests": {
       "cpu": "237m",
       "memory": "512Mi"
      },
      "limits": {
       "cpu": "2000m",
       "memory": "2048Mi"
      }
     }
    }
    ```

  - For information about supported `addonVersion`, see Kubernetes versions supported by GuardDuty security agent.

- Alternatively, you can use AWS CLI. For more information, see create-addon.

Updating security agent manually

When you manage the GuardDuty security agent manually, you are responsible to update it for your account. For notification about new agent versions, you can subscribe to an RSS feed to GuardDuty agent release history.

You can update the security agent to the latest version to benefit from the added support and improvements. If your current agent version is reaching an end of standard support, then to continue using Runtime Monitoring (or EKS Runtime Monitoring), you must update your current agent version. For information about release versions, see GuardDuty security agent for Amazon EKS clusters.

Prerequisite

> Before you update the security agent version, make sure that the agent version that you're planning to use now, is compatible with your Kubernetes version. For more information, see Kubernetes versions supported by GuardDuty security agent.

Console

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. Choose your **Cluster name**.
3. Choose **Add-ons**.
4. Under **Add-ons**, select **GuardDuty Runtime Monitoring**.
5. Choose **Edit** to update the agent details.
6. On the **Configure GuardDuty Runtime Monitoring** page, update the details.
7. **(Optional) Updating add-on configuration parameters**

   If your EKS add-on **Version** is 1.5.0 or above, you can also update the add-on configuration settings.

   a. Expand **Optional configuration settings** to view the configuration schema.
   b. Update the parameter values based on the range provided in Configure EKS add-on parameters.
   c. Choose **Save changes** to start the update.
   d. For **Conflict resolution method**, the option that you choose will be used to resolve a conflict when you update the value of a parameter to a non-default value. For more

information about the listed options, see resolveConflicts in the *Amazon EKS API Reference*.

API/CLI

To update the GuardDuty security agent for your Amazon EKS clusters, see Updating an add-on.

> **Note**
>
> For the add-on `version`, if you choose **v1.5.0** and above, Runtime Monitoring supports configuring specific parameters of the GuardDuty agent. For information about parameter ranges, see Configure EKS add-on parameters.

You can use the following AWS CLI example when using configurable values supported for addon versions v1.5.0 and above. Make sure to replace the placeholder values highlighted in red and the associated `Example.json` with the configured values.

```
aws eks update-addon --region us-east-1 --cluster-name myClusterName --addon-name aws-guardduty-agent --addon-version v1.5.0-eksbuild.1 --configuration-values 'file://example.json'
```

Example Example.json

```
{
 "priorityClassName": "aws-guardduty-agent.priorityclass-high",
 "dnsPolicy": "Default",
 "resources": {
  "requests": {
   "cpu": "237m",
   "memory": "512Mi"
  },
  "limits": {
   "cpu": "2000m",
   "memory": "2048Mi"
  }
 }
}
```

If your Amazon EKS add-on version is 1.5.0 or above, and you have configured the add-on schema, you can verify whether or not the values appear correctly for your cluster. For more information, see Verifying configuration schema updates.

Verifying configuration schema updates

After you have configured the parameters, perform the following steps to verify that the configuration schema has been updated:

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. In the navigation pane, choose **Clusters**.
3. On the **Clusters** page, select the **Cluster name** for which you want to verify the updates.
4. Choose the **Resources** tab.
5. From the **Resource types** pane, under **Workloads**, choose **DaemonSets**.
6. Select **aws-guardduty-agent**.
7. On the **aws-guardduty-agent** page, choose **Raw view** to view the unformatted JSON response. Verify that the configurable parameters display the value that you provided.

After you verify, switch to the GuardDuty console. Select the corresponding AWS Region and view the coverage status for your Amazon EKS clusters. For more information, see Coverage for Amazon EKS clusters.

Configuring EKS Runtime Monitoring (API only)

Before configuring EKS Runtime Monitoring in your account, make sure that you're using one of the verified platforms that supports the Kubernetes version that is presently in use. For more, see Validating architectural requirements.

Configuring EKS Runtime Monitoring for a standalone account

For the accounts associated with AWS Organizations, see Configuring EKS Runtime Monitoring for multiple-account environments.

Choose your preferred access method to enable EKS Runtime Monitoring for your account.

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

Preferred approach to manage GuardDuty security agent	**Steps**
Manage security agent through GuardDuty (Monitor all EKS clusters)	1. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account. 2. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] ]'`
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged -false`. For more information

Preferred approach to manage GuardDuty security agent	Steps
	about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource`. - Replace *`ec2:DeleteTags`* with `eks:UntagResource`. - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn`: `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. **Note** Always add the exclusion tag to your EKS cluster before setting the `STATUS` of `EKS_RUNTIME_MONITORING` to `ENABLED`; otherwise, the GuardDuty security agent will

Preferred approach to manage GuardDuty security agent	Steps
	be deployed on all the EKS clusters in your account. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] ]'`

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` -`true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`.

Preferred approach to manage GuardDuty security agent	Steps
	Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged` -`true` pair. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "DISABLED"}] ]'`

Preferred approach to manage GuardDuty security agent	Steps
Manage the security agent manually	1. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " DISABLED"}] ]'` 2. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Configuring EKS Runtime Monitoring for multiple-account environments

In a multiple-account environments, only the delegated GuardDuty administrator account can enable or disable EKS Runtime Monitoring for the member accounts, and manage GuardDuty agent management for the EKS clusters belonging to the member accounts in their organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account account manages their member accounts using AWS Organizations. For more information about multi-account environments, see Managing multiple accounts.

Configuring EKS Runtime Monitoring for delegated GuardDuty administrator account

Choose your preferred access method to enable EKS Runtime Monitoring and manage the GuardDuty security agent for the EKS clusters that belong to the delegated GuardDuty administrator account.

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --feature`

Preferred approach to manage GuardDuty security agent	Steps
	`s '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] ]'`

Preferred approach to manage GuardDuty security agent	**Steps**
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` -`false`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. **Note** Always add the exclusion tag to your EKS cluster before setting the STATUS of

Preferred approach to manage GuardDuty security agent	Steps
	EKS_RUNTIME_MONITORING to ENABLED; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] ]'`

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` -`true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`.

Preferred approach to manage GuardDuty security agent	Steps
	Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged` -`true` pair. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " DISABLED"}] ]'`

<table>
<tr><th>Preferred approach to manage GuardDuty security agent</th><th>Steps</th></tr>
<tr><td>Manage the security agent manually</td><td>

1. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`.

   Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`.

   Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` :

   ```
   aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 555555555555 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] }]'
   ```
2. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

</td></tr>
</table>

Auto-enable EKS Runtime Monitoring for all member accounts

Choose your preferred access method to enable EKS Runtime Monitoring for all member accounts. This includes the delegated GuardDuty administrator account, existing member accounts, and the new accounts that join the organization. Choose your preferred approach to manage GuardDuty security agent for the EKS clusters that belong to these member accounts.

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

Preferred approach to manage GuardDuty security agent	**Steps**
Manage security agent through GuardDuty (Monitor all EKS clusters)	To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] ]'` **Note** You can also pass a list of account IDs separated by a space.

Preferred approach to manage GuardDuty security agent	Steps
	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.

Preferred approach to manage GuardDuty security agent	**Steps**
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDuty Managed` -`false`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: • Replace *`ec2:CreateTags`* with `eks:TagResource` . • Replace *`ec2:DeleteTags`* with `eks:UntagResource` . • Replace *`access-project`* with `GuardDutyManaged` • Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. **Note** Always add the exclusion tag to your EKS cluster before setting the `STATUS` of `EKS_RUNTIME_MONITORING` to `ENABLED`; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account.

Preferred approach to manage GuardDuty security agent	Steps
	Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] ]'` **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

Preferred approach to manage GuardDuty security agent	Steps
	problems changing the detector settings for an account, that account ID is listed along with a summary of the issue.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` -`true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`.

Preferred approach to manage GuardDuty security agent	Steps
	GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged -true` pair. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " DISABLED"}] ]'` **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.

Preferred approach to manage GuardDuty security agent	**Steps**
Manage the security agent manually	1. Run the updateDetector API by using your own regional detector ID and passing the `features` object name as `EKS_RUNTIME_MONITORING` and status as `ENABLED`. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 555555555555 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] } ]'` 2. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Configuring EKS Runtime Monitoring for all existing active member accounts

Choose your preferred access method to enable EKS Runtime Monitoring and manage GuardDuty security agent for existing active member accounts in your organization.

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

Preferred approach to manage GuardDuty security agent	Steps
Manage security agent through GuardDuty (Monitor all EKS clusters)	To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] ]'` **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.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged -false`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. ⓘ **Note** Always add the exclusion tag to your EKS cluster before setting the `STATUS` of `EKS_RUNTIME_MONITORING` to `ENABLED`; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account.

Preferred approach to manage GuardDuty security agent	Steps
	To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*. Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] ]'` **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

Preferred approach to manage GuardDuty security agent	Steps
	problems changing the detector settings for an account, that account ID is listed along with a summary of the issue.

Preferred approach to manage GuardDuty security agent	**Steps**
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDuty Managed` -`true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *`detector ID`*. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`.

Preferred approach to manage GuardDuty security agent	Steps
	GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged -true` pair.

Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` :

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "DISABLED"}] ]'
```

> **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.

<table>
<tr><th>Preferred approach to manage GuardDuty security agent</th><th>Steps</th></tr>
<tr><td>Manage the security agent manually</td><td>

1. To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*.

   Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`.

   Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` :

   ```
   aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 555555555555 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : " ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] } ]'
   ```

2. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

</td></tr>
</table>

Auto-enable EKS Runtime Monitoring for new members

The delegated GuardDuty administrator account can auto-enable EKS Runtime Monitoring and choose an approach for how to manage the GuardDuty security agent for new accounts that join your organization.

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

<table>
<tr><th>Preferred approach to manage GuardDuty security agent</th><th>Steps</th></tr>
<tr><td>Manage security agent through GuardDuty (Monitor all EKS clusters)</td><td>
<p>To selectively enable EKS Runtime Monitoring for your new accounts, invoke the UpdateOrganizationConfiguration API operation using your own <em>detector ID</em>.</p>
<p>Set the status for <code>EKS_ADDON_MANAGEMENT</code> as <code>ENABLED</code>.</p>
<p>GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account.</p>
<p>Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the <code>detectorId</code> for your account and current Region, see <strong>Settings</strong> page in the https://console.aws.amazon.com/guardduty/ console.</p>
<p>The following example enables both <code>EKS_RUNTIME_MONITORING</code> and <code>EKS_ADDON_MANAGEMENT</code> for a single account. You can also pass a list of account IDs separated by a space.</p>
<p>To find the <code>detectorId</code> for your account and current Region, see <strong>Settings</strong> page in the https://console.aws.amazon.com/guardduty/ console.</p>
<pre><code>aws guardduty update-organization-configuration --detector-id 12abc34d567e8fa901bc2d34e56789f0 --autoEnable --features '[{"Name" : "EKS_RUNTIME_MONITORING", "AutoEnable": "NEW", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "AutoEnable": "NEW"}]]'</code></pre>
<p>When the code has successfully executed, it returns an empty list of <code>UnprocessedAccounts</code>. If there were</p>
</td></tr>
</table>

Preferred approach to manage GuardDuty security agent	Steps
	any problems changing the detector settings for an account, that account ID is listed along with a summary of the issue.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` -`false`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. **Note** Always add the exclusion tag to your EKS cluster before setting the STATUS of

Preferred approach to manage GuardDuty security agent	Steps
	EKS_RUNTIME_MONITORING to ENABLED; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. To selectively enable EKS Runtime Monitoring for your new accounts, invoke the [UpdateOrganizationConfiguration](#) API operation using your own *detector ID*. Set the status for EKS_ADDON_MANAGEMENT as ENABLED. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both EKS_RUNTIME_MONITORING and EKS_ADDON_MANAGEMENT for a single account. You can also pass a list of account IDs separated by a space. To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. `aws guardduty update-organization-configuration --detector-id 12abc34d567e8fa901`

Preferred approach to manage GuardDuty security agent	Steps
	`bc2d34e56789f0 --autoEnable --features '[{"Name" : "EKS_RUNTIME_MONITORING", "AutoEnable": "NEW", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "AutoEnable": "NEW"}] ]'` 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.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged -true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. To selectively enable EKS Runtime Monitoring for your new accounts, invoke the UpdateOrganizationConfiguration API operation using your own *`detector ID`*.

Preferred approach to manage GuardDuty security agent	Steps
	Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged` -`true` pair. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` for a single account. You can also pass a list of account IDs separated by a space. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. `aws guardduty update-organization-configuration --detector-id 12abc34d567e8fa901bc2d34e56789f0 --autoEnable --features '[{"Name" : "EKS_RUNTIME_MONITORING", "AutoEnable": "NEW", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "AutoEnable": "NEW"}] ]'` When the code has successfully executed, it returns an empty list of `UnprocessedAccounts`. If there were any problems changing the detector settings

Preferred approach to manage GuardDuty security agent	Steps
	for an account, that account ID is listed along with a summary of the issue.

Preferred approach to manage GuardDuty security agent	**Steps**
Manage the security agent manually	1. To selectively enable EKS Runtime Monitoring for your new accounts, invoke the UpdateOrganizationConfiguration API operation using your own *detector ID*. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` for a single account. You can also pass a list of account IDs separated by a space. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. `aws guardduty update-organization-configuration --detector-id 12abc34d567e8fa901bc2d34e56789f0 --autoEnable --features '[{"Name" : "EKS_RUNTIME_MONITORING", "AutoEnable": "NEW", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "AutoEnable": "NEW"}] ]'` 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.

Preferred approach to manage GuardDuty security agent	Steps
	2. To manage the security agent, see Managing security agent manually for Amazon EKS cluster.

Enable EKS Runtime Monitoring for individual active member accounts

API/CLI

Based on the Approaches to manage GuardDuty security agent, you can choose a preferred approach and follow the steps as mentioned in the following table.

Preferred approach to manage GuardDuty security agent	Steps

Manage security agent through GuardDuty (Monitor all EKS clusters)	To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*.
	Set the status for `EKS_ADDON_MANAGEMENT` as `ENABLED`.
	GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters in your account.
	Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.
	The following example enables both `EKS_RUNTIME_MONITORING` and `EKS_ADDON_MANAGEMENT` :
	`aws guardduty update-member-detectors --`
	`detector-id 12abc34d567e8fa901bc2d34e56`
	`789f0 --account-ids 111122223333 --feature`

Preferred approach to manage GuardDuty security agent	Steps
	`s '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : "ENABLED"}] ]'` **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.

Preferred approach to manage GuardDuty security agent	Steps
Monitor all EKS clusters but exclude some of them (using exclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged -false`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. **Note** Always add the exclusion tag to your EKS cluster before setting the STATUS of

Preferred approach to manage GuardDuty security agent	Steps
	EKS_RUNTIME_MONITORING to ENABLED; otherwise, the GuardDuty security agent will be deployed on all the EKS clusters in your account. To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *detector ID*. Set the status for EKS_ADDON_MANAGEMENT as ENABLED. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have not been excluded from being monitored. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables both EKS_RUNTIME_MONITORING and EKS_ADDON_MANAGEMENT : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] ]'`

Preferred approach to manage GuardDuty security agent	Steps
	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.

Preferred approach to manage GuardDuty security agent	Steps
Monitor selective EKS clusters (using inclusion tag)	1. Add a tag to the EKS cluster that you want to exclude from being monitored. The key-value pair is `GuardDutyManaged` - `true`. For more information about adding the tag, see Working with tags using the CLI, API, or eksctl in the Amazon EKS User Guide. 2. To prevent modification of tags, except by the trusted entities, use the policy provided in Prevent tags from being modified except by authorized principals in the *AWS Organizations User Guide*. In this policy, replace the following details: - Replace *`ec2:CreateTags`* with `eks:TagResource` . - Replace *`ec2:DeleteTags`* with `eks:UntagResource` . - Replace *`access-project`* with `GuardDutyManaged` - Replace *`123456789012`* with the AWS account ID of the trusted entity. When you have more than one trusted entities, use the following example to add multiple `PrincipalArn` : `"aws:PrincipalArn":["arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin", "arn:aws:iam::123456789012:role/org-admins/iam-admin"]` 3. To selectively enable EKS Runtime Monitoring for your member accounts, run the updateMemberDetectors API operation using your own *`detector ID`*.

Preferred approach to manage GuardDuty security agent	Steps
	Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. GuardDuty will manage the deployment of and updates to the security agent for all the Amazon EKS clusters that have been tagged with the `GuardDutyManaged` -`true` pair. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " DISABLED"}] ]'` **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

Preferred approach to manage GuardDuty security agent	Steps
	for an account, that account ID is listed along with a summary of the issue.
Manage the security agent manually	1. To selectively enable EKS Runtime Monitoring for your member accounts, run the [updateMemberDetectors](#) API operation using your own *detector ID*. Set the status for `EKS_ADDON_MANAGEMENT` as `DISABLED`. Alternatively, you can use the AWS CLI command by using your own regional detector ID. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console. The following example enables `EKS_RUNTIME_MONITORING` and disables `EKS_ADDON_MANAGEMENT` : `aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 555555555555 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "ENABLED", "AdditionalConfiguration" : [{"Name" : "EKS_ADDON_MANAGEMENT", "Status" : " ENABLED"}] } ]'` 2. To manage the security agent, see [Managing security agent manually for Amazon EKS cluster](#).

Migrating from EKS Runtime Monitoring to Runtime Monitoring

With the launch of GuardDuty Runtime Monitoring, the threat detection coverage has been expanded to Amazon ECS containers and Amazon EC2 instances. EKS Runtime Monitoring has now been consolidated into Runtime Monitoring. You can enable Runtime Monitoring and manage individual GuardDuty security agents for each resource type (Amazon EC2 instance, Amazon ECS cluster, and Amazon EKS cluster) for which you want to monitor the runtime behavior.

There is no separate GuardDuty console experience for EKS Runtime Monitoring. To continue using EKS Runtime Monitoring, you need to configure it using APIs or the AWS Command Line Interface.

To migrate from EKS Runtime Monitoring to Runtime Monitoring

1. The GuardDuty console supports EKS Runtime Monitoring as a part of Runtime Monitoring.

   You can start using Runtime Monitoring by Checking EKS Runtime Monitoring configuration status of your organization and accounts.

   Make sure to not disable EKS Runtime Monitoring before enabling Runtime Monitoring. If you disable EKS Runtime Monitoring, the Amazon EKS add-on management will also get disabled. Continue with the following steps in the listed order.
2. Make sure you meet all the Prerequisites to enabling Runtime Monitoring.
3. Enable Runtime Monitoring by replicating the same organization configuration settings for Runtime Monitoring as you have for EKS Runtime Monitoring. For more information, see Enabling Runtime Monitoring.
   - If you have a standalone account, you need to enable Runtime Monitoring.

     If your GuardDuty security agent is deployed already, the corresponding settings are replicated automatically and you don't need to configure the settings again.
   - If you have an organization with auto-enablement settings, make sure to replicate the same auto-enablement settings for Runtime Monitoring.
   - If you have an organization with settings configured for existing active member accounts individually, make sure to enable Runtime Monitoring and configure the GuardDuty security agent for these members individually.
4. After you have ensured that the Runtime Monitoring and GuardDuty security agent settings are correct, disable EKS Runtime Monitoring by using either the API or the AWS CLI command.

5. (Optional) if you want to clean any resource associated with the GuardDuty security agent, see Cleaning up GuardDuty security agent resources.

If you want to continue using EKS Runtime Monitoring without enabling Runtime Monitoring, see Configuring EKS Runtime Monitoring (API only).

Checking EKS Runtime Monitoring configuration status

Use the following APIs or AWS CLI commands to check the existing configuration status of EKS Runtime Monitoring.

To check existing EKS Runtime Monitoring configuration status in your account

- Run GetDetector to check the configuration status of your own account.
- Alternatively, you can run the following command by using AWS CLI:

  ```
  aws guardduty get-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
  region us-east-1
  ```

  Make sure to replace the detector ID of your AWS account and the current Region. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

To check existing EKS Runtime Monitoring configuration status for your organization (as a delegated GuardDuty administrator account only)

- Run DescribeOrganizationConfiguration to check the configuration status of your organization.

  Alternatively, you can run the following command using AWS CLI:

  ```
  aws guardduty describe-organization-configuration --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --region us-east-1
  ```

  Make sure to replace the detector ID with the detector ID of your delegated GuardDuty administrator account and the Region with your current Region. To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

Disabling EKS Runtime Monitoring after migrating to Runtime Monitoring

After you have ensured that the existing settings for your account or organization have been replicated to Runtime Monitoring, you can disable EKS Runtime Monitoring.

To disable EKS Runtime Monitoring

- **To disable EKS Runtime Monitoring in your own account**

  Run the UpdateDetector API with your own regional *detector-id*.

  Alternatively, you can use the following AWS CLI command. Replace *12abc34d567e8fa901bc2d34e56789f0* with your own regional *detector-id*.

  ```
  aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
  features '[{"Name" : "EKS_RUNTIME_MONITORING", "Status" : "DISABLED"}]'
  ```

- **To disable EKS Runtime Monitoring for member accounts in your organization**

  Run the UpdateMemberDetectors API with the regional *detector-id* of the delegated GuardDuty administrator account of the organization.

  Alternatively, you can use the following AWS CLI command. Replace *12abc34d567e8fa901bc2d34e56789f0* with the regional *detector-id* of the delegated GuardDuty administrator account of the organization and *111122223333* with the AWS account ID of the member account for which you want to disable this feature.

  ```
  aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
   --account-ids 111122223333 --features '[{"Name" : "EKS_RUNTIME_MONITORING",
   "Status" : "DISABLED"}]'
  ```

- **To update EKS Runtime Monitoring auto-enable settings for your organization**

  Perform the following step only if you have configured the EKS Runtime Monitoring auto-enablement settings to either new (`NEW`) or all (`ALL`) member accounts in the organization. If you had already configured it as `NONE`, then you can skip this step.

> **Note**
>
> Setting the EKS Runtime Monitoring auto-enable configuration to `NONE` means that EKS Runtime Monitoring will not be enabled automatically for any existing member account or when a new member account joins your organization.

Run the [UpdateOrganizationConfiguration](https://docs.aws.amazon.com/guardduty/latest/APIReference/API_UpdateOrganizationConfiguration.html) API with the regional *detector-id* of the delegated GuardDuty administrator account of the organization.

Alternatively, you can use the following AWS CLI command. Replace *12abc34d567e8fa901bc2d34e56789f0* with the regional *detector-id* of the delegated GuardDuty administrator account of the organization. Replace the *EXISTING_VALUE* with your current configuration for auto-enabling GuardDuty.

```
aws guardduty update-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable-organization-members EXISTING_VALUE
 --features '[{"Name" : "EKS_RUNTIME_MONITORING", "AutoEnable": "NONE"}]'
```

Cleaning up GuardDuty security agent resources

You must clean up the resources in the following scenarios:

When you disable Automated agent configuration

GuardDuty doesn't remove the security agent that is deployed on your resource. However, GuardDuty will stop managing the updates to the security agent.

GuardDuty continues to receive the runtime events from your resource type. To prevent an impact on your usage statistics, make sure to remove the GuardDuty security agent from your resource.

Whether or not an AWS account uses a shared VPC endpoint, GuardDuty doesn't delete the VPC endpoint. If required, you will need to delete the VPC endpoint manually.

When you disable Runtime Monitoring or EKS Runtime Monitoring

In general, GuardDuty deletes the VPC endpoint that it had created. GuardDuty tags them as `GuardDutyManaged:true`. However, when you disable the Runtime Monitoring or EKS

Runtime Monitoring for a shared VPC participant account - if the shared VPC endpoint has been used by at least one participant account, GuardDuty neither deletes the VPC endpoint nor the security group associated with the shared VPC resource.

When you stop managing the security agent manually

Regardless of which approach you use to deploy and manage the GuardDuty security agent, to stop monitoring the runtime events in your resource, you must remove the GuardDuty security agent. When you want to stop monitoring the runtime events from a resource type in an account, you may also delete the Amazon VPC endpoint.

Process to clean up security agent resources

To delete Amazon VPC endpoint

- Without a shared VPC – When you no longer want to monitor a resource in an account, consider deleting the Amazon VPC endpoint.
- With a shared VPC – When the shared VPC owner account deletes the shared VPC resource, any participant account that is currently using the shared VPC endpoint, the Runtime Monitoring coverage status for the resources in your shared VPC owner account and the participating account might become unhealthy. For more information, see Assessing runtime coverage for your resources.

For more information, see Delete an interface endpoint.

To delete the security group

- Without a shared VPC – When you no longer want to monitor a resource type in an account, consider deleting the security group associated with the Amazon VPC.
- With a shared VPC – When the shared VPC owner account deletes the security group, any participant account that is currently using the security group associated with the shared VPC, the Runtime Monitoring coverage status for the resources in your shared VPC owner account and the participating account might become unhealthy. For more information, see Assessing runtime coverage for your resources.

For more information, see Delete a security group.

To remove GuardDuty security agent from an EKS cluster

To remove the security agent from your EKS cluster that you no longer want to monitor, see Deleting an add-on.

Removing the EKS add-on agent doesn't remove the `amazon-guardduty` namespace from the EKS cluster. To delete the `amazon-guardduty` namespace, see Deleting a namespace.

To delete the `amazon-guardduty` namespace (EKS cluster)

Disabling Automated agent configuration doesn't automatically remove the `amazon-guardduty` namespace from your EKS cluster. To delete the `amazon-guardduty` namespace, see Deleting a namespace.

Assessing runtime coverage for your resources

After you enable Runtime Monitoring and the GuardDuty security agent gets deployed to your resource, GuardDuty provides coverage statistics for the corresponding resource type and individual coverage status for the resources that belong to your account. Coverage status is determined by making sure that you have enabled Runtime Monitoring, your Amazon VPC endpoint has been created, and the GuardDuty security agent for the corresponding resource has been deployed. A **Healthy** coverage status indicates that when there is a runtime event related to your resource, GuardDuty is able to receive the said runtime event through the Amazon VPC endpoint, and monitor the behavior. If there was an issue at the time of configuring Runtime Monitoring, creating an Amazon VPC endpoint, or deploying the GuardDuty security agent, the coverage status appears as **Unhealthy**. When the coverage status is unhealthy, GuardDuty will not be able to receive or monitor the runtime behavior of the corresponding resource, or generate any Runtime Monitoring findings.

The following topics will help you review coverage statistics, configure EventBridge notifications, and troubleshoot the coverage issues for a specific resource type.

Contents

Coverage for Amazon EC2 instance

For an Amazon EC2 resource, the runtime coverage is evaluated at the instance level. Your Amazon EC2 instances can run multiple types of applications and workloads amongst others in your AWS environment. This feature also supports Amazon ECS managed Amazon EC2 instances and if you have Amazon ECS clusters running on an Amazon EC2 instance, the coverage issues at the instance level will show up under Amazon EC2 runtime coverage.

Topics

- Reviewing coverage statistics
- Configuring coverage status change notifications
- Troubleshooting coverage issues

Reviewing coverage statistics

The coverage statistics for the Amazon EC2 instances associated with your own accounts or your member accounts is the percentage of the healthy EC2 instances over all EC2 instances in the selected AWS Region. The following equation represents this as:

*(Healthy instances/All instances)*100*

If you have also deployed the GuardDuty security agent for your Amazon ECS clusters, then any instance level coverage issue associated with Amazon ECS clusters running on an Amazon EC2 instance will appear as an Amazon EC2 instance runtime coverage issue.

Choose one of the access methods to review the coverage statistics for your accounts.

Console

- Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
- In the navigation pane, choose **Runtime Monitoring**.
- Choose the **Runtime coverage** tab.
- Under the **EC2 instance runtime coverage** tab, you can view the coverage statistics aggregated by the coverage status of each Amazon EC2 instance that is available in the **Instances list** table.
  - You can filter the **Instance list** table by the following columns:
    - **Account ID**
    - **Agent management type**
    - **Agent version**

    - **Coverage status**
    - **Instance ID**
    - **Cluster ARN**
- If any of your EC2 instances have the **Coverage status** as **Unhealthy**, the **Issue** column includes additional information about the reason for the **Unhealthy** status.

API/CLI

- Run the ListCoverage API with your own valid detector ID, current Region, and service endpoint. You can filter and sort the instance list using this API.
  - You can change the example `filter-criteria` with one of the following options for `CriterionKey`:
    - `ACCOUNT_ID`
    - `RESOURCE_TYPE`
    - `COVERAGE_STATUS`
    - `AGENT_VERSION`
    - `MANAGEMENT_TYPE`
    - `INSTANCE_ID`
    - `CLUSTER_ARN`
  - When the `filter-criteria` includes `RESOURCE_TYPE` as **EC2**, Runtime Monitoring doesn't support the use of **ISSUE** as the `AttributeName`. If you use it, the API response will result in `InvalidInputException`.

    You can change the example `AttributeName` in `sort-criteria` with the following options:
    - `ACCOUNT_ID`
    - `COVERAGE_STATUS`
    - `INSTANCE_ID`
    - `UPDATED_AT`
  - You can change the *`max-results`* (up to 50).
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 list-coverage --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --sort-criteria '{"AttributeName":
 "EKS_CLUSTER_NAME", "OrderBy": "DESC"}' --filter-criteria
 '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID", "FilterCondition":
{"EqualsValue":"111122223333"}}] }'  --max-results 5
```

- Run the GetCoverageStatistics API to retrieve coverage aggregated statistics based on the `statisticsType`.
  - You can change the example `statisticsType` to one of the following options:
    - `COUNT_BY_COVERAGE_STATUS` – Represents coverage statistics for EKS clusters aggregated by coverage status.
    - `COUNT_BY_RESOURCE_TYPE` – Coverage statistics aggregated based on the type of AWS resource in the list.
    - You can change the example `filter-criteria` in the command. You can use the following options for `CriterionKey`:
      - `ACCOUNT_ID`
      - `RESOURCE_TYPE`
      - `COVERAGE_STATUS`
      - `AGENT_VERSION`
      - `MANAGEMENT_TYPE`
      - `INSTANCE_ID`
      - `CLUSTER_ARN`
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 get-coverage-statistics --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --statistics-type COUNT_BY_COVERAGE_STATUS
 --filter-criteria '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID",
 "FilterCondition":{"EqualsValue":"123456789012"}}] }'
```

If the coverage status of your EC2 instance is **Unhealthy**, see Troubleshooting coverage issues.

Configuring coverage status change notifications

The coverage status of your Amazon EC2 instance might appear as **Unhealthy**. To know when the coverage status changes, we recommend you to monitor the coverage status periodically, and troubleshoot if the status becomes **Unhealthy**. Alternatively, you can create an Amazon EventBridge rule to receive a notification when the coverage status changes from either **Unhealthy** to **Healthy** or otherwise. By default, GuardDuty publishes this in the EventBridge bus for your account.

Sample notification schema

In an EventBridge rule, you can use the pre-defined sample events and event patterns to receive coverage status notification. For more information about creating an EventBridge rule, see Create rule in the *Amazon EventBridge User Guide*.

Additionally, you can create a custom event pattern by using the following example notification schema. Make sure to replace the values for your account. To get notified when the coverage status of your Amazon EC2 instance changes from `Healthy` to `Unhealthy`, the `detail-type` should be *`GuardDuty Runtime Protection Unhealthy`*. To get notified when the coverage status changes from `Unhealthy` to `Healthy`, replace the value of `detail-type` with *`GuardDuty Runtime Protection Healthy`*.

```
{
  "version": "0",
  "id": "event ID",
  "detail-type": "GuardDuty Runtime Protection Unhealthy",
  "source": "aws.guardduty",
  "account": "AWS account ID",
  "time": "event timestamp (string)",
  "region": "AWS Region",
  "resources": [
       ],
  "detail": {
    "schemaVersion": "1.0",
    "resourceAccountId": "string",
    "currentStatus": "string",
    "previousStatus": "string",
    "resourceDetails": {
        "resourceType": "EC2",
        "ec2InstanceDetails": {
          "instanceId":"",
     "instanceType":"",
     "clusterArn": "",
     "agentDetails": {
      "version":""
     },
     "managementType":""
    }
  },
  "issue": "string",
  "lastUpdatedAt": "timestamp"
 }
}
```

Troubleshooting coverage issues

If the coverage status of your Amazon EC2 instance is **Unhealthy**, you can view the reason under the **Issue** column.

If your EC2 instance is associated with an EKS cluster and the security agent for EKS was installed either manually or through automated agent configuration, then to troubleshoot the coverage issue, see Coverage for Amazon EKS clusters.

The following table lists the issue types and the corresponding troubleshooting steps.

Issue type	Issue message	Troubleshooting steps
No Agent Reporting	Waiting for SSM notification	Make sure that the Amazon EC2 instance is already SSM managed. Receiving the SSM notification may take a few minutes.
	(Empty on purpose)	If you are managing the GuardDuty security agent manually, this is not actionable.
		If you've enabled automated agent configuration: • Your EC2 instance is SSM managed. • View the status of your security agent periodically. For more information, see Validating GuardDuty security agent installation status.

Issue type	Issue message	Troubleshooting steps
	Agent disconnected	- View the status of your security agent. For more information, see Validating GuardDuty security agent installation status. - View the security agent logs to identify the potential root cause. The logs provide detailed errors that you can use to troubleshoot the issue yourself. The log files are available under `/var/log/amzn-guardduty-agent/` . Do `sudo journalctl -u amazon-guardduty-agent` .
SSM Association Creation Failed	GuardDuty SSM association already exists in your account	1. Delete the existing association manually. For more information, see Deleting associations in the *AWS Systems Manager User Guide*. 2. After you delete the association, disable and then re-enable the GuardDuty automated agent configuration for Amazon EC2.
	Your account has too many SSM associations	Choose **one** of the following two options: - Delete any unused SSM associations. For more information, see Deleting associations in the *AWS Systems Manager User Guide*. - Check if your account is eligible for a quota increase. For more information, see Systems Manager Service quotas in the *AWS General Reference*.
SSM Association Updation Failed	GuardDuty SSM association does not exist in your account	The SSM association is missing. Disable and then re-enable Runtime Monitoring.

Issue type	Issue message	Troubleshooting steps
SSM Association Deletion Failed	GuardDuty SSM association does not exist in your account	The SSM association is not present in your account. If the SSM association was deleted intentionally, then no action is needed.
SSM Instance Association Execution Failed	Architectural requirements or other prerequisites are not met.	For information about verified operating system distributions, see Prerequisites for Amazon EC2 instance support. If you still experience this issue, the following steps will help you identify and potentially resolve the issue: 1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/. 2. In the navigation pane, under **Node management**, select **State Manager**. 3. Filter by **Document Name** property and enter **AmazonGuardDuty-ConfigureRuntimeMonitoringSsmPlugin**. 4. Select the corresponding association ID and view its **Execution history**. 5. Using the execution history, view the failures, identify the potential root cause, and try to resolve it.
VPC Endpoint Creation Failed	VPC endpoint creation not supported for shared VPC *`vpcId`*	Runtime Monitoring supports the use of a shared VPC within an organization. For more information, see Using shared VPC with automated security agents.

Issue type	Issue message	Troubleshooting steps
	Only when using shared VPC with automated agent configuration Owner account ID *111122223333* for shared VPC *vpcId* doesn't have either Runtime Monitoring, automated agent configuration, or both, enabled	The shared VPC owner account must enable Runtime Monitoring and automated agent configuration for at least one resource type (Amazon EKS or Amazon ECS (AWS Fargate)). For more information, see Prerequisites specific to GuardDuty Runtime Monitoring.

Issue type	Issue message	Troubleshooting steps
	Enabling private DNS requires both `enableDnsSupport` and `enableDnsHostnames` VPC attributes set to `true` for *`vpcId`* (Service: Ec2, Status Code:400, Request ID: *`a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`*).	Ensure that the following VPC attributes are set to `true` – `enableDnsSupport` and `enableDnsHostnames` . For more information, see [DNS attributes in your VPC](). If you're using Amazon VPC Console at https://console.aws.amazon.com/vpc/ to create the Amazon VPC, make sure to select both **Enable DNS hostnames** and **Enable DNS resolution**. For more information, see [VPC configuration options]().
Shared VPC Endpoint Deletion Failed	Shared VPC endpoint deletion not allowed for account ID *`111122223333`* , shared VPC *`vpcId`*, owner account ID *`555555555555`* .	**Potential steps:** • Disabling the Runtime Monitoring status of the shared VPC participant account doesn't impact the shared VPC endpoint policy and the security group that exists in the owner account. To delete the shared VPC endpoint and security group, you must disable Runtime Monitoring or automated agent configuration status in the shared VPC owner account. • The shared VPC participant account can't delete the shared VPC endpoint and security group hosted in the shared VPC owner account.

Issue type	Issue message	Troubleshooting steps
Agent not reporting	(Empty on purpose)	The issue type has reached end of support. If you continue experiencing this issue and not already done so, enable GuardDuty automated agent for Amazon EC2. If the issue still persists, consider disabling Runtime Monitoring for a few minutes and then enable it again.

Coverage for Amazon ECS clusters

The runtime coverage for Amazon ECS clusters includes the tasks running on AWS Fargate (Fargate) and Amazon ECS container instances[1].

For an Amazon ECS cluster that runs on Fargate, the runtime coverage is assessed at the task level. The ECS clusters runtime coverage includes those Fargate tasks that have started running after you have enabled Runtime Monitoring and automated agent configuration for Fargate (ECS only). By default, a Fargate task is immutable. GuardDuty will not be able to install the security agent to monitor containers on already running tasks. To include such a Fargate task, you must stop and start the task again. Make sure to check if the associated service is supported.

Presently, Runtime Monitoring doesn't support the tasks launched by AWS Batch and AWS CodePipeline.

For information about Amazon ECS container, see Capacity creation.

Contents

Reviewing coverage statistics

The coverage statistics for the Amazon ECS resources associated with your own account or your member accounts is the percentage of the healthy Amazon ECS clusters over all the Amazon ECS clusters in the selected AWS Region. This includes the coverage for Amazon ECS clusters associated with both Fargate and Amazon EC2 instances. The following equation represents this as:

*(Healthy clusters/All clusters)*100*

Considerations

- The coverage statistics for the ECS cluster include the coverage status of the Fargate tasks or ECS container instances associated with that ECS cluster. The coverage status of the Fargate tasks include tasks that either are in running state or have recently finished running.
- In the **ECS clusters runtime coverage** tab, the **Container instances covered** field indicates the coverage status of the container instances associated with your Amazon ECS cluster.

  If your Amazon ECS cluster contains only Fargate tasks, the count appears as **0/0**.
- If your Amazon ECS cluster is associated with an Amazon EC2 instance that doesn't have a security agent, the Amazon ECS cluster will also have an **Unhealthy** coverage status.

  To identify and troubleshoot the coverage issue for the associated Amazon EC2 instance, see Troubleshooting coverage issues for Amazon EC2 instances.

Choose one of the access methods to review the coverage statistics for your accounts.

Console

- Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
- In the navigation pane, choose **Runtime Monitoring**.
- Choose the **Runtime coverage** tab.
- Under the **ECS clusters runtime coverage** tab, you can view the coverage statistics aggregated by the coverage status of each Amazon ECS cluster that is available in the **Clusters list** table.
  - You can filter the **Cluster list** table by the following columns:
    - **Account ID**
    - **Cluster Name**
    - **Agent management type**
    - **Coverage status**
- If any of your Amazon ECS clusters have the **Coverage status** as **Unhealthy**, the **Issue** column includes additional information about the reason for the **Unhealthy** status.

If you Amazon ECS clusters are associated with an Amazon EC2 instance, navigate to the **EC2 instance runtime coverage** tab and filter by the **Cluster name** field to view the associated **Issue**.

API/CLI

- Run the ListCoverage API with your own valid detector ID, current Region, and service endpoint. You can filter and sort the instance list using this API.
  - You can change the example `filter-criteria` with one of the following options for `CriterionKey`:
    - `ACCOUNT_ID`
    - `ECS_CLUSTER_NAME`
    - `COVERAGE_STATUS`
    - `MANAGEMENT_TYPE`
  - You can change the example `AttributeName` in `sort-criteria` with the following options:
    - `ACCOUNT_ID`
    - `COVERAGE_STATUS`
    - `ISSUE`
    - `ECS_CLUSTER_NAME`
    - `UPDATED_AT`

      The field gets updated only when either a new task gets created in the associated Amazon ECS cluster or there is change in the corresponding coverage status.
  - You can change the *`max-results`* (up to 50).
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 list-coverage --detector-id 12abc34d567e8fa901bc2d34e56789f0 --sort-criteria '{"AttributeName": "ECS_CLUSTER_NAME", "OrderBy": "DESC"}' --filter-criteria '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID", "FilterCondition":{"EqualsValue":"111122223333"}}] }'  --max-results 5
```

- Run the GetCoverageStatistics API to retrieve coverage aggregated statistics based on the `statisticsType`.
  - You can change the example `statisticsType` to one of the following options:
    - `COUNT_BY_COVERAGE_STATUS` – Represents coverage statistics for ECS clusters aggregated by coverage status.
    - `COUNT_BY_RESOURCE_TYPE` – Coverage statistics aggregated based on the type of AWS resource in the list.
    - You can change the example `filter-criteria` in the command. You can use the following options for `CriterionKey`:
      - `ACCOUNT_ID`
      - `ECS_CLUSTER_NAME`
      - `COVERAGE_STATUS`
      - `MANAGEMENT_TYPE`
      - `INSTANCE_ID`
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 get-coverage-statistics --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --statistics-type COUNT_BY_COVERAGE_STATUS
 --filter-criteria '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID",
 "FilterCondition":{"EqualsValue":"123456789012"}}] }'
```

For more information about coverage issues, see Troubleshooting coverage issues.

Configuring coverage status change notifications

The coverage status of your Amazon ECS cluster might appear as **Unhealthy**. To know when the coverage status changes, we recommend you to monitor the coverage status periodically, and troubleshoot if the status becomes **Unhealthy**. Alternatively, you can create an Amazon EventBridge rule to receive a notification when the coverage status changes from either **Unhealthy** to **Healthy** or otherwise. By default, GuardDuty publishes this in the EventBridge bus for your account.

Sample notification schema

In an EventBridge rule, you can use the pre-defined sample events and event patterns to receive coverage status notification. For more information about creating an EventBridge rule, see Create rule in the *Amazon EventBridge User Guide*.

Additionally, you can create a custom event pattern by using the following example notification schema. Make sure to replace the values for your account. To get notified when the coverage status of your Amazon ECS cluster changes from `Healthy` to `Unhealthy`, the `detail-type` should be *`GuardDuty Runtime Protection Unhealthy`*. To get notified when the coverage status changes from `Unhealthy` to `Healthy`, replace the value of `detail-type` with *`GuardDuty Runtime Protection Healthy`*.

```
{
  "version": "0",
  "id": "event ID",
  "detail-type": "GuardDuty Runtime Protection Unhealthy",
  "source": "aws.guardduty",
  "account": "AWS account ID",
  "time": "event timestamp (string)",
  "region": "AWS Region",
  "resources": [
       ],
  "detail": {
    "schemaVersion": "1.0",
    "resourceAccountId": "string",
    "currentStatus": "string",
    "previousStatus": "string",
    "resourceDetails": {
        "resourceType": "ECS",
        "ecsClusterDetails": {
          "clusterName":"",
          "fargateDetails":{
            "issues":[],
            "managementType":""
          },
          "containerInstanceDetails":{
            "coveredContainerInstances":int,
            "compatibleContainerInstances":int
          }
        }
    },
    "issue": "string",
    "lastUpdatedAt": "timestamp"
  }
}
```

Troubleshooting coverage issues

If the coverage status of your Amazon ECS cluster is **Unhealthy**, you can view the reason under the **Issue** column.

The following table provides the recommended troubleshooting steps for Fargate (Amazon ECS only) issues. For information about Amazon EC2 instancecoverage issues, see Troubleshooting coverage issues for Amazon EC2 instances.

Issue type	Extra information	Recommended troubleshooting steps
Agent not reporting	Agent not reporting for tasks in `TaskDefinition - '` *`TASK_DEFINITION`* `'`	Validate that your Amazon VPC endpoint configuration is correct.
	`VPC_ISSUE` `; for task in TaskDefinition - '` *`TASK_DEFINITION`* `'`	View the VPC issue details in the extra information.
Agent exited	ExitCode: EXIT_CODE for tasks in `TaskDefinition - '` *`TASK_DEFINITION`* `'`	View the issue details in the extra information.
	Reason: *REASON* for tasks in `TaskDefinition -` `'`*`TASK_DEFINITION`* `'`	
	ExitCode: EXIT_CODE with reason: '*EXIT_CODE* ' for tasks	

Issue type	Extra information	Recommended troubleshooting steps
	in `TaskDefinition` - '*TASK_DEFINITION*'	

	Agent exited: Reason: `CannotPullContainerError` : pull image manifest has been retried...	The task execution role must have the following Amazon Elastic Container Registry (Amazon ECR) permissions:
		`...`
		`        "ecr:GetAuthorizationToken",`
		`        "ecr:BatchCheckLayerAvailability",`
		`        "ecr:GetDownloadUrlForLayer",`
		`        "ecr:BatchGetImage",`
		`...`
		For more information, see Provide ECR permissions and subnet details.
		After you add the Amazon ECR permissions, you must restart the task.
		If the issue persists, see My AWS Step Functions workflow is failing unexpectedly.

ExitCode: `EXIT_CODE` for tasks in `TaskDefinition` - '*TASK_DEFINITION*'	View the issue details in the extra information.	
ExitCode: `EXIT_CODE` with reason: '*EXIT_CODE*' for tasks in `TaskDefinition` - '*TASK_DEFINITION*'		

Issue type	Extra information	Recommended troubleshooting steps
Reason:*REASON* for tasks in TaskDefinition - '*TASK_DEFINITION* '		
Agent not reporting	Agent not reporting for tasks in TaskDefinition - ' *TASK_DEFINITION* '	Validate that your Amazon VPC endpoint configuration is correct.
	VPC_ISSUE ; for task in TaskDefinition - ' *TASK_DEFINITION* '	View the VPC issue details in the extra information.

<table>
<tr><th>Issue type</th><th>Extra information</th><th>Recommended troubleshooting steps</th></tr>
<tr><td>Others or Agent not provisioned</td><td>Unidentified issue, for tasks in <code>TaskDefinition - '<i>TASK_DEFINITION</i>'</code></td><td>Use the following questions to identify the root cause of the issue:
<table>
<tr><th>Question</th><th>Explanation</th></tr>
<tr><td>Did the task start before you enabled Runtime Monitoring?</td><td>In Amazon ECS, the tasks are immutable. To assess the runtime behavior of a running Fargate task, make sure that Runtime Monitoring is already enabled, and then restart the task for GuardDuty to add the container sidecar.</td></tr>
<tr><td>Was the task launched by an unsupported service?</td><td>Presently, Runtime Monitoring doesn't support the tasks launched by AWS Batch and AWS CodePipeline.</td></tr>
<tr><td>Is this task part of a service</td><td>If yes, you can either restart</td></tr>
</table></td></tr>
</table>

Issue type	Extra information	Recommended troubleshooting steps	
		Question	**Explanation**
		deployment that started before you enabled Runtime Monitoring?	the service or update the service with `forceNewDeployment` by using the steps in [Updating a service](). You can also use [UpdateService]() or [AWS CLI]().
		Did the task launch after excluding the ECS cluster from Runtime Monitoring?	When you change the pre-defined GuardDuty tag from `GuardDuty Managed -true` to `GuardDuty Managed -false`, GuardDuty will not receive the runtime events for the ECS cluster.
		Is your task missing a `TaskExecutionRole` ?	It is mandatory to add a `TaskExecutionRole` because

Issue type	Extra information	Recommended troubleshooting steps	
		Question	Explanation
			GuardDuty needs permissions to download the GuardDuty container from the ECR repository. For more information, see Provide ECR permissions and subnet details.
		Does your service contain a task that has an old format of `taskArn`?	GuardDuty Runtime Monitoring doesn't support the coverage for tasks that have the old format of `taskArn`. For information about Amazon Resource Names (ARNs) for Amazon ECS resources, see Amazon Resource Names (ARNs) and IDs.

Coverage for Amazon EKS clusters

After you enable Runtime Monitoring and install the GuardDuty security agent (add-on) for EKS either manually or through automated agent configuration, you can start assessing the coverage for your EKS clusters.

Contents

Reviewing coverage statistics

The coverage statistics for the EKS clusters associated with your own accounts or your member accounts is the percentage of the healthy EKS clusters over all EKS clusters in the selected AWS Region. The following equation represents this as:

*(Healthy clusters/All clusters)*100*

Choose one of the access methods to review the coverage statistics for your accounts.

Console

- Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
- In the navigation pane, choose **Runtime Monitoring**.
- Choose the **EKS clusters runtime coverage** tab.
- Under the **EKS clusters runtime coverage** tab, you can view the coverage statistics aggregated by the coverage status that is available in the **Clusters list** table.
  - You can filter the **Clusters list** table by the following columns:
    - **Cluster name**
    - **Account ID**
    - **Agent management type**
    - **Coverage status**
    - **Add-on version**

- If any of your EKS clusters have the **Coverage status** as **Unhealthy**, the **Issue** column may include additional information about the reason for the **Unhealthy** status.

API/CLI

- Run the ListCoverage API with your own valid detector ID, Region, and service endpoint. You can filter and sort the cluster list using this API.
  - You can change the example `filter-criteria` with one of the following options for `CriterionKey`:
    - `ACCOUNT_ID`
    - `CLUSTER_NAME`
    - `RESOURCE_TYPE`
    - `COVERAGE_STATUS`
    - `ADDON_VERSION`
    - `MANAGEMENT_TYPE`
  - You can change the example `AttributeName` in `sort-criteria` with the following options:
    - `ACCOUNT_ID`
    - `CLUSTER_NAME`
    - `COVERAGE_STATUS`
    - `ISSUE`
    - `ADDON_VERSION`
    - `UPDATED_AT`
  - You can change the *`max-results`* (up to 50).
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 list-coverage --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --sort-criteria '{"AttributeName":
 "EKS_CLUSTER_NAME", "OrderBy": "DESC"}' --filter-criteria
 '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID", "FilterCondition":
{"EqualsValue":"111122223333"}}] }'  --max-results 5
```

- Run the GetCoverageStatistics API to retrieve coverage aggregated statistics based on the `statisticsType`.
  - You can change the example `statisticsType` to one of the following options:
    - `COUNT_BY_COVERAGE_STATUS` – Represents coverage statistics for EKS clusters aggregated by coverage status.
    - `COUNT_BY_RESOURCE_TYPE` – Coverage statistics aggregated based on the type of AWS resource in the list.
    - You can change the example `filter-criteria` in the command. You can use the following options for `CriterionKey`:
      - `ACCOUNT_ID`
      - `CLUSTER_NAME`
      - `RESOURCE_TYPE`
      - `COVERAGE_STATUS`
      - `ADDON_VERSION`
      - `MANAGEMENT_TYPE`
  - To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty --region us-east-1 get-coverage-statistics --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --statistics-type COUNT_BY_COVERAGE_STATUS
 --filter-criteria '{"FilterCriterion":[{"CriterionKey":"ACCOUNT_ID",
 "FilterCondition":{"EqualsValue":"123456789012"}}] }'
```

If the coverage status of your EKS cluster is **Unhealthy**, see Troubleshooting EKS coverage issues.

Configuring coverage status change notifications

The coverage status of an EKS cluster in your account may show up as **Unhealthy**. To detect when the coverage status becomes **Unhealthy**, we recommend you monitor the coverage status periodically and troubleshoot, if the status is **Unhealthy**. Alternatively, you can create an Amazon EventBridge rule to notify you when the coverage status changes from either `Unhealthy` to `Healthy` or otherwise. By default, GuardDuty publishes this in the EventBridge bus for your account.

Sample notification schema

In an EventBridge rule, you can use the pre-defined sample events and event patterns to receive coverage status notification. For more information about creating an EventBridge rule, see Create rule in the *Amazon EventBridge User Guide*.

Additionally, you can create a custom event pattern by using the following example notification schema. Make sure to replace the values for your account. To get notified when the coverage status of your Amazon EKS cluster changes from `Healthy` to `Unhealthy`, the `detail-type` should be *`GuardDuty Runtime Protection Unhealthy`*. To get notified when the coverage status changes from `Unhealthy` to `Healthy`, replace the value of `detail-type` with *`GuardDuty Runtime Protection Healthy`*.

```
{
  "version": "0",
  "id": "event ID",
  "detail-type": "GuardDuty Runtime Protection Unhealthy",
  "source": "aws.guardduty",
  "account": "AWS account ID",
  "time": "event timestamp (string)",
  "region": "AWS Region",
  "resources": [
       ],
  "detail": {
    "schemaVersion": "1.0",
    "resourceAccountId": "string",
    "currentStatus": "string",
    "previousStatus": "string",
    "resourceDetails": {
        "resourceType": "EKS",
        "eksClusterDetails": {
            "clusterName": "string",
            "availableNodes": "string",
             "desiredNodes": "string",
             "addonVersion": "string"
         }
    },
    "issue": "string",
    "lastUpdatedAt": "timestamp"
  }
}
```

Troubleshooting EKS coverage issues

If the coverage status for your EKS cluster is `Unhealthy`, you can view the corresponding error either under the **Issue** column in the GuardDuty console, or by using the CoverageResource data type.

When working with inclusion or exclusion tags for monitoring your EKS clusters selectively, it may take some time for the tags to sync. This may impact the coverage status of the associated EKS cluster. You can try removing and adding the corresponding tag (inclusion or exclusion) again. For more information, see Tagging your Amazon EKS resources in the Amazon EKS User Guide.

The structure of a coverage issue is `Issue type:Extra information`. Typically, the issues will have an optional *Extra information* that may include specific client-side exception or description about the issue. Based on *Extra information*, the following tables provide the recommended steps to troubleshoot the coverage issues for your EKS clusters.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
Addon Creation Failed	Addon `aws-guardduty-agent` is not compatible with current cluster version of cluster *`ClusterName`* . Addon specified is not supported.	Make sure that you're using one of those Kubernetes versions that support deploying the `aws-guardduty-agent` EKS add-on. For more information, see Kubernetes versions supported by GuardDuty security agent. For information about updating your Kubernetes version, see Updating an Amazon EKS cluster Kubernetes version.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
Addon Creation Failed Addon Updation Failed Addon Status Unhealthy	EKS Addon issue - `AddonIssueCode : AddonIssueMessage`	For information about recommended steps for a specific add-on issue code, see Troubleshooting steps for Addon creation/updatation error with Addon issue code. For a list of addon issue codes that you might experience in this issue, see AddonIssue.
VPC Endpoint Creation Failed	VPC endpoint creation not supported for shared VPC *`vpcId`*	Runtime Monitoring now supports the use of a shared VPC within an organization. Make sure your accounts meet all the prerequisites. For more information, see Prerequisites for using shared VPC.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
	Only when using shared VPC with automated agent configuration Owner account ID *111122223333* for shared VPC *vpcId* doesn't have either Runtime Monitoring, automated agent configuration, or both, enabled.	The shared VPC owner account must enable Runtime Monitoring and automated agent configuration for at least one resource type (Amazon EKS or Amazon ECS (AWS Fargate)). For more information, see Prerequisites specific to GuardDuty Runtime Monitoring.
	Enabling private DNS requires both `enableDnsSupport` and `enableDnsHostnames` VPC attributes set to `true` for *vpcId* (Service: Ec2, Status Code:400, Request ID: *a1b2c3d4-5678-90ab-cdef-EXAMPLE11111*).	Ensure that the following VPC attributes are set to `true` – `enableDnsSupport` and `enableDnsHostnames` . For more information, see DNS attributes in your VPC. If you're using Amazon VPC Console at https://console.aws.amazon.com/vpc/ to create the Amazon VPC, make sure to select both **Enable DNS hostnames** and **Enable DNS resolution**. For more information, see VPC configuration options.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
Shared VPC Endpoint Deletion Failed	Shared VPC endpoint deletion not allowed for account ID *111122223333* , shared VPC *vpcId*, owner account ID *555555555555* .	**Potential steps:** • Disabling the Runtime Monitoring status of the shared VPC participant account doesn't impact the shared VPC endpoint policy and the security group that exists in the owner account. To delete the shared VPC endpoint and security group, you must disable Runtime Monitoring or automated agent configuration status in the shared VPC owner account. • The shared VPC participant account can't delete the shared VPC endpoint and security group hosted in the shared VPC owner account.
Local EKS clusters	EKS addons are not supported on local outpost clusters.	Not actionable. For more information, see Amazon EKS on AWS outposts.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
EKS Runtime Monitoring enablement permission not granted	(may or may not show extra information)	1. If the extra informati on is available for this issue, fix the root cause and follow the next step. 2. Toggle EKS Runtime Monitoring to turn it off and then turn it on again. Ensure that the GuardDuty agent also gets deployed, whether automatically through GuardDuty or manually.
EKS Runtime Monitoring enablement resource provisioning in progress	(may or may not show extra information)	Not actionable. After you enable EKS Runtime Monitoring, the coverage status might remain `Unhealthy` until the resource provisioning step completes. The coverage status gets monitored and updated periodically.

Issue type (prefix)	Extra information	Recommended troubleshooting steps
Others (any other issue)	Error due to authorization failure	Toggle EKS Runtime Monitoring to turn it off and then turn it on again. Ensure that the GuardDuty agent also gets deployed, either automatically through GuardDuty or manually.

Addon creation or updation error	Troubleshooting steps
EKS Addon Issue - `InsufficientNumberOfReplicas` : The add-on is unhealthy because it doesn't have the desired number of replicas.	Using the issue message, you can identify and fix the root cause. You can start by describing your cluster. For example, use `kubectl describe pods` to identify the root cause for pod failure. After you fix the root cause, retry the step (add-on creation or update).
EKS Addon Issue - `AdmissionRequestDenied` : admission webhook `"validate.kyverno.svc-fail"` denied the request: policy `DaemonSet/amazon-guardduty/aws-guardduty-agent` for resource violation: restrict-image-registries: `autogen-validate-registries` :...	1. Amazon EKS cluster or the security administrator must review the security policy that is blocking the Addon update. 2. You must either disable the controller (webhook) or have the controller accept the requests from Amazon EKS.
EKS Addon Issue - `ConfigurationConflict` : Conflicts found when trying to apply. Will not continue due to resolve conflicts mode. `Conflicts: DaemonSet.apps`	When creating or updating the Addon, provide the `OVERWRITE` resolve conflict flag. This will potentially overwrite any changes that have

<table>
<tr><th>Addon creation or updation error</th><th>Troubleshooting steps</th></tr>
<tr><td>

`aws-guardduty-agent - .spec.template.spec.containers[name="aws-guardduty-agent"].image`

</td><td>

been made directly to the related resources in Kubernetes by using the Kubernetes API.

You can first delete the Addon and then reinstall.

</td></tr>
<tr><td>

EKS Addon Issue - `AccessDenied: priorityclasses.scheduling.k8s.io "aws-guardduty-agent.priorityclass" is forbidden: User "eks:addon-manager" cannot patch resource "priorityclasses" in API group "scheduling.k8s.io" at the cluster scope`

</td><td>

You must add the missing permission to the `eks:addon-cluster-admin ClusterRoleBinding` manually. Add the following `yaml` to `eks:addon-cluster-admin` :

```
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization
.k8s.io/v1
metadata:
  name: eks:addon-cluster-admin
subjects:
- kind: User
  name: eks:addon-manager
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
---
```

You can now apply this `yaml` to your Amazon EKS cluster by using the following command:

```
kubectl apply -f eks-addon-cluster-
admin.yaml
```

</td></tr>
</table>

Addon creation or updation error	Troubleshooting steps
EKS Addon Issue - `AccessDenied: admission webhook "validation.gatekeeper.sh" denied the request: [all-namespace-must-have-label-owner] All namespaces must have an `owner` label`	You must either disable the controller or have the controller accept the requests from the Amazon EKS cluster. Prior to creating or updating the add-on, you can also create a GuardDuty namespace and label it as `owner`.

Frequently asked questions (FAQs)

Contents

Why is the coverage status for my resource `Unhealthy` even after enabling Runtime Monitoring, deploying the GuardDuty security agent, and meeting all the prerequisites?

If you have just deployed the GuardDuty security agent (either through automated agent configuration or manually) or followed the recommended steps to troubleshoot a coverage issue, it might take a few minutes for the coverage status to become healthy. You can either check the coverage status periodically or configure Amazon EventBridge (EventBridge) to receive a notification when the coverage status changes.

Who can view the runtime coverage status of a resource that belongs to my AWS account?

As a member account or standalone account, you can view the coverage statistics of the resources associated with your own accounts. As a delegated GuardDuty administrator account of an organization, you can view the coverage statistics for the resources associated with your account and the member accounts that belong to your organization.

Setting up CPU and memory monitoring

After you enable Runtime Monitoring and assess that the coverage status of your cluster is **Healthy**, you can set up and view the insight metrics.

The following topics can help you evaluate how the deployed agent performs against the CPU and memory limits for the GuardDuty agent.

Setting up monitoring on Amazon ECS cluster

The following steps from the *Amazon CloudWatch User Guide* can help you evaluate how the deployed agent performs against the CPU and memory limits for the GuardDuty agent:

1. Setting up Container Insights on Amazon ECS for cluster- and service-level metrics
2. Amazon ECS Container Insights metrics

Setting up monitoring on Amazon EKS cluster

After the GuardDuty security agent gets deployed and you assess that the coverage status of your cluster is **Healthy**, you can set up and view the Container insight metrics.

Evaluate performance of the security agent

1. Setting up Container Insights on Amazon EKS and Kubernetes in the *Amazon CloudWatch User Guide*
2. Amazon EKS and Kubernetes Container Insights metrics in the *Amazon CloudWatch User Guide*

Manage performance with security agent v1.5.0 and above

With security agent v1.5.0 and above, when the insights indicate that the associated GuardDuty agent is reaching the assigned limits, you can configure specific parameters. For more information, see Configure EKS add-on parameters.

Collected runtime event types that GuardDuty uses

The GuardDuty security agent collects the following events types and sends them to the GuardDuty backend for threat detection and analysis. GuardDuty doesn't make these events accessible to you. If GuardDuty detects a potential threat and generates a Runtime Monitoring finding, you can view the corresponding finding details. For more information about how

GuardDuty uses the collected event types, see Opting out of using your data for service improvement.

Process events

Field name	Description
Process name	Name of the observed process.
Process Path	Absolute path of the process executable.
Process ID	The ID assigned to the process by the operating system.
Namespace PID	The process ID of the process in a secondary PID namespace other than the host level PID namespace. For processes inside a container, it is the process ID observed inside the container.
Process User ID	The unique ID of the user that executed the process.
Process UUID	The unique ID assigned to the process by GuardDuty.
Process GID	Process ID of the process group.
Process EGID	Effective group ID of the process group.
Process EUID	Effective user ID of the process.
Process User Name	The user name that executed the process.
Process Start Time	The time when the process was created. This field is in the UTC date string format (`2023-03-22T19:37:20.168Z`).
Process Executable SHA-256	The SHA256 hash of the process executable.
Process Script Path	Path of the script file that was executed.

Field name	Description
Process Environment Variable	The environment variable made available to the process. Only `LD_PRELOAD` and `LD_LIBRARY_PATH` get collected.
Process Present Working Directory (PWD)	Present working directory of the process.
Parent process	Process details of the parent process. A parent process is a process that created the observed process.
Command Line Arguments Presently, this field is limited to specific agent versions corresponding to the resource type: • Fargate (Amazon ECS only) with GuardDuty security agent v1.0.0 and above. • Amazon EC2 instances with GuardDuty security agent v1.0.0 and above. • Amazon EKS clusters with security agent v1.4.0 and above. For more information, see GuardDuty agent release history.	Command-line arguments provided at the time of process execution. This field might contain sensitive customer data.

Container events

Field name	Description
Container Name	Name of the container. When available, this field displays the value of the label `io.kubenetes.container.name` .

Field name	Description
Container UID	The unique ID of the container assigned by the container runtime.
Container Runtime	The container runtime (such as `docker` or `containerd`) used to run the container.
Container Image ID	The ID of the container image.
Container Image Name	Name of the container image.

AWS Fargate (Amazon ECS only) task events

Field name	Description
Task Amazon Resource Name (ARN)	The ARN of the task.
Cluster Name	The name of the Amazon ECS cluster.
Family Name	The task definition's family name. The `family` is used as a name for the task definition that is used to launch the task.
Service Name	The name of the Amazon ECS service, if the task was launched as part of a service.
Launch Type	The infrastructure on which your task runs. For Runtime Monitoring with resource type as `ECSCluster` , the launch type could be either `EC2` or `FARGATE`.
CPU	The number of CPU units used by the task as expressed in the task definition.

Kubernetes pod events

Field name	Description
Pod ID	The ID of the Kubernetes pod.
Pod name	Name of the Kubernetes pod.
Pod Namespace	Name of the Kubernetes namespace to which the Kubernetes workload belongs.
Kubernetes Cluster Name	Name of the Kubernetes cluster.

DNS events

Field name	Description
Socket Type	Type of socket to indicate communication semantics. For example, `SOCK_RAW`.
Address Family	Represents the communication protocol associated with the address. For example, the address family `AF_INET` is used for IP v4 protocol.
Direction ID	The ID of the connection direction.
Protocol Number	The layer 4 protocol number such as 17 for UDP and 6 for TCP.
DNS Remote Endpoint IP	The remote IP of the connection.
DNS Remote Endpoint Port	The port number of the connection.
DNS Local Endpoint IP	The local IP of the connection.
DNS Local Endpoint Port	The port number of the connection.
DNS Payload	The payload of DNS packets that contains DNS queries and responses.

Open events

Field name	Description
Filepath	Path of the file that is opened in this event.
Flags	Describes the file access mode, such as read-only, write-only, and read-write.

Load module event

Field name	Description
Module Name	Name of the module loaded into the kernel.

Mprotect events

Field name	Description
Address Range	The address range for which the access protections were modified.
Memory Regions	Specifies the Region of a process's address space such as stack and heap.
Flags	Represents options that control the behavior of this event.

Mount events

Field name	Description
Mount Target	The path where the mount source is mounted.
Mount Source	The path on the host that is mounted at the mount target.

Field name	Description
Filesystem Type	Represents the type of mounted fileSystem.
Flags	Represents options that control the behavior of this event.

Link events

Field name	Description
Link Path	Path where the hard link gets created.
Target Path	Path of the file at which the hard link points.

Symlink events

Field name	Description
Link Path	Path where the symbolic link is created.
Target Path	Path of the file at which the symbolic link points.

Dup events

Field name	Description
Old File Descriptor	A file descriptor that represents an open file object.
New File Descriptor	A new file descriptor that is a duplicate of the old file descripto r. Both the old and new file descriptors represent the same open file object.
Dup Remote Endpoint IP	The remote IP address of the network socket represented by the old file descriptor. Only applicable when the old file descriptor represents a network socket.

Field name	Description
Dup Remote Endpoint Port	The remote port of the network socket represented by the old file descriptor. Only applicable when the old file descriptor represents a network socket.
Dup Local Endpoint IP	The local IP address of the network socket represented by the old file descriptor. Only applicable when the old file descriptor represents a network socket.
Dup Local Endpoint Port	The local port of the network socket represented by the old file descriptor. Only applicable when the old file descriptor represents a network socket.

Memory map event

Field name	Description
Filepath	Path of the file to which the memory is mapped.

Socket events

Field name	Description
Address family	Represents the communication protocol associated with the address. For example, the address family `AF_INET` is used for IP version of 4 protocol.
Socket Type	Type of socket to indicate communication semantics. For example, `SOCK_RAW`.
Protocol number	Specifies a particular protocol within the address family. Usually there is a single protocol in address families. For example, the address family `AF_INET` only has the IP protocol.

Connect events

Field name	Description
Address family	Represents the communication protocol associated with the address. For example, the address family `AF_INET` is used for IP v4 protocol.
Socket Type	Type of socket to indicate communication semantics. For example, `SOCK_RAW`.
Protocol Number	Specifies a particular protocol within the address family. Usually there is a single protocol in address families. For example, the address family `AF_INET` only has the IP protocol.
Filepath	Path of the socket file if the address family is `AF_UNIX`.
Remote Endpoint IP	The remote IP of the connection.
Remote Endpoint Port	The port number of the connection.
Local Endpoint IP	The local IP of the connection.
Local Endpoint Port	The port number of the connection.

Process VM Readv events

Field name	Description
Flags	Represents options that control the behavior of this event.
Target PID	Process ID of the process from which memory is being read.
Target Process UUID	The unique ID of the target process.
Target Executable Path	The absolute path of the target process executable file.

Process VM Writev events

Field name	Description
Flags	Represents options that control the behavior of this event.
Target PID	Process ID of the process to which memory is being written.
Target Process UUID	The unique ID of the target process.
Target Executable Path	The absolute path of the target process executable file.

Ptrace events

Field name	Description
Target PID	Process ID of the target process.
Target Process UUID	The unique ID of the target process.
Target Executable Path	The absolute path of the target process executable file.
Flags	Represents options that control the behavior of this event.

Bind events

Field name	Description
Address Family	Represents the communication protocol associated with the address. For example, the address family `AF_INET` is used for IP v4 protocol.
Socket type	Type of socket to indicate communication semantics. For example, `SOCK_RAW`.
Protocol number	The layer 4 protocol number such as 17 for UDP and 6 for TCP.

Field name	Description
Local endpoint IP	The local IP of the connection.
Local endpoint port	The port number of the connection.

Listen events

Field name	Description
Address Family	Represents the communication protocol associated with the address. For example, the address family `AF_INET` is used for IP v4 protocol.
Socket type	Type of socket to indicate communication semantics. For example, `SOCK_RAW`.
Protocol number	The layer 4 protocol number such as 17 for UDP and 6 for TCP.
Local endpoint IP	The local IP of the connection.
Local endpoint port	The port number of the connection.

Rename events

Field name	Description
Filepath	Path where the file that is renamed.
Target	The new path of the file.

Set UID events

Field name	Description
New EUID	The new effective user ID of the process.
New UID	The new user ID of the process.

Chmod events

Field name	Description
Filepath	Path of the file that invokes this event.
Filemode	The updated access permissions for the associated file.

Amazon ECR repository hosting GuardDuty agent

The following sections list the Amazon Elastic Container Registry (Amazon ECR) repositories where GuardDuty hosts the security agent that gets deployed on your Amazon EKS and Amazon ECS clusters.

Contents

Repository for GuardDuty agent on Amazon EKS clusters

The following table shows the Amazon ECR repositories that hosts the Amazon EKS add-on agent for GuardDuty (`aws-guardduty-agent`) for each AWS Region.

AWS Region	Amazon ECR repository URI
US West (Oregon)	`039403964562.dkr.ecr.us-west-2.amazonaws.com`

AWS Region	Amazon ECR repository URI
Europe (Paris)	`113643092156.dkr.ecr.eu-west-3.amazonaws.com`
Asia Pacific (Mumbai)	`610108029387.dkr.ecr.ap-south-1.amazonaws.com`
Asia Pacific (Hyderabad)	`618745550137.dkr.ecr.ap-south-2.amazonaws.com`
Canada (Central)	`001188825231.dkr.ecr.ca-central-1.amazonaws.com`
Middle East (UAE)	`601769779514.dkr.ecr.me-central-1.amazonaws.com`
Europe (London)	`109118265657.dkr.ecr.eu-west-2.amazonaws.com`
Europe (Ireland)	`373421517865.dkr.ecr.us-west-1.amazonaws.com`
US East (N. Virginia)	`031903291036.dkr.ecr.us-east-1.amazonaws.com`
US East (Ohio)	`591382732059.dkr.ecr.us-east-2.amazonaws.com`
Europe (Ireland)	`673884943994.dkr.ecr.eu-west-1.amazonaws.com`
South America (São Paulo)	`941219317354.dkr.ecr.sa-east-1.amazonaws.com`
Europe (Stockholm)	`366771026645.dkr.ecr.eu-north-1.amazonaws.com`
Europe (Frankfurt)	`409493279830.dkr.ecr.eu-central-1.amazonaws.com`
Europe (Zurich)	`718440343717.dkr.ecr.eu-central-2.amazonaws.com`
Asia Pacific (Singapore)	`584580519942.dkr.ecr.ap-southeast-1.amazonaws.com`

AWS Region	Amazon ECR repository URI
Asia Pacific (Sydney)	`011662287384.dkr.ecr.ap-southeast-2.amazonaws.com`
Asia Pacific (Jakarta)	`617474730032.dkr.ecr.ap-southeast-3.amazonaws.com`
Asia Pacific (Tokyo)	`781592569369.dkr.ecr.ap-northeast-1.amazonaws.com`
Asia Pacific (Seoul)	`732248494576.dkr.ecr.ap-northeast-2.amazonaws.com`
Asia Pacific (Osaka)	`810724417379.dkr.ecr.ap-northeast-3.amazonaws.com`
Asia Pacific (Hong Kong)	`790429075973.dkr.ecr.ap-east-1.amazonaws.com`
Middle East (Bahrain)	`541829937850.dkr.ecr.me-south-1.amazonaws.com`
Europe (Milan)	`528450769569.dkr.ecr.eu-south-1.amazonaws.com`
Europe (Spain)	`531047660167.dkr.ecr.eu-south-2.amazonaws.com`
Africa (Cape Town)	`379032919888.dkr.ecr.af-south-1.amazonaws.com`
Asia Pacific (Melbourne)	`750462861327.dkr.ecr.ap-southeast-4.amazonaws.com`
Israel (Tel Aviv)	`292660727137.dkr.ecr.il-central-1.amazonaws.com`

Repository for GuardDuty agent on AWS Fargate (Amazon ECS only)

The following table shows the Amazon ECR repositories that hosts the GuardDuty agent for AWS Fargate (Amazon ECS only) for each AWS Region.

AWS Region	Amazon ECR repository URI
US West (Oregon)	`733349766148.dkr.ecr.us-west-2.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Paris)	`665651866788.dkr.ecr.eu-west-3.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Mumbai)	`251508486986.dkr.ecr.ap-south-1.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Hyderabad)	`950823858135.dkr.ecr.ap-south-2.amazonaws.com/aws-guardduty-agent-fargate`
Canada (Central)	`354763396469.dkr.ecr.ca-central-1.amazonaws.com/aws-guardduty-agent-fargate`
Middle East (UAE)	`000014521398.dkr.ecr.me-central-1.amazonaws.com/aws-guardduty-agent-fargate`
Europe (London)	`892757235363.dkr.ecr.eu-west-2.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Ireland)	`684579721401.dkr.ecr.us-west-1.amazonaws.com/aws-guardduty-agent-fargate`
US East (N. Virginia)	`593207742271.dkr.ecr.us-east-1.amazonaws.com/aws-guardduty-agent-fargate`
US East (Ohio)	`307168627858.dkr.ecr.us-east-2.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Ireland)	`694911143906.dkr.ecr.eu-west-1.amazonaws.com/aws-guardduty-agent-fargate`
South America (São Paulo)	`758426053663.dkr.ecr.sa-east-1.amazonaws.com/aws-guardduty-agent-fargate`

AWS Region	Amazon ECR repository URI
Europe (Stockholm)	`591436053604.dkr.ecr.eu-north-1.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Frankfurt)	`323658145986.dkr.ecr.eu-central-1.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Zurich)	`529164026651.dkr.ecr.eu-central-2.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Singapore)	`174946120834.dkr.ecr.ap-southeast-1.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Sydney)	`005257825471.dkr.ecr.ap-southeast-2.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Jakarta)	`510637619217.dkr.ecr.ap-southeast-3.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Tokyo)	`533107202818.dkr.ecr.ap-northeast-1.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Seoul)	`914738172881.dkr.ecr.ap-northeast-2.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Osaka)	`273192626886.dkr.ecr.ap-northeast-3.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Hong Kong)	`258348409381.dkr.ecr.ap-east-1.amazonaws.com/aws-guardduty-agent-fargate`
Middle East (Bahrain)	`536382113932.dkr.ecr.me-south-1.amazonaws.com/aws-guardduty-agent-fargate`
Europe (Milan)	`266869475730.dkr.ecr.eu-south-1.amazonaws.com/aws-guardduty-agent-fargate`

AWS Region	Amazon ECR repository URI
Europe (Spain)	`919611009337.dkr.ecr.eu-south-2.amazonaws.com/aws-guardduty-agent-fargate`
Africa (Cape Town)	`197869348890.dkr.ecr.af-south-1.amazonaws.com/aws-guardduty-agent-fargate`
Asia Pacific (Melbourne)	`251357961535.dkr.ecr.ap-southeast-4.amazonaws.com/aws-guardduty-agent-fargate`
Israel (Tel Aviv)	`870907303882.dkr.ecr.il-central-1.amazonaws.com/aws-guardduty-agent-fargate`

GuardDuty agent release history

The following sections provide the release version for GuardDuty agent that gets deployed on Amazon EC2 instances, Amazon ECS clusters, and Amazon EKS clusters

GuardDuty security agent for Amazon EC2 instances

Agent version	Release notes	Availability date
v1.1.0	Supports GuardDuty automated agent configuration in Runtime Monitoring for Amazon EC2 instances. Supports new security signals and findings released with the announcement of general availability of Runtime Monitoring for EC2 instances. General performance improvement.	March 26, 2024

Agent version	Release notes	Availability date
v1.0.2	Supports the latest Amazon ECS AMIs.	February 2, 2024
v1.0.1	General performance tuning and enhancements Agent versions released prior to v1.0.2 are incompatible with Amazon ECS AMIs launched after January 31, 2024.	January 23, 2024
v1.0.0	Initial release of the RPM installation. Agent versions released prior to v1.0.2 are incompatible with Amazon ECS AMIs launched after January 31, 2024.	November 26, 2023

The public key, signature of x86_64 RPM, signature of arm64 RPM, and the corresponding access link to the RPM scripts hosted in Amazon S3 buckets can be formed from the following templates. Replace the value of the AWS Region, AWS account ID, and the GuardDuty agent version to access the RPM scripts. The following templates include the latest agent version for Amazon EC2 instances.

- **Public key**:

  ```
  s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/publickey.pem
  ```

- **GuardDuty security agent RPM signature**:

  Signature of x86_64 RPM

  ```
  s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/x86_64/amazon-guardduty-agent-1.1.0.x86_64.sig
  ```

Signature of arm64 RPM

```
s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/arm64/amazon-guardduty-agent-1.1.0.arm64.sig
```

- **Access links to the RPM scripts in Amazon S3 bucket**:

  Access link for x86_64 RPM

  ```
  s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/x86_64/amazon-guardduty-agent-1.1.0.x86_64.rpm
  ```

  Access link for arm64 RPM

  ```
  s3://694911143906-eu-west-1-guardduty-agent-rpm-artifacts/1.1.0/arm64/amazon-guardduty-agent-1.1.0.arm64.rpm
  ```

AWS Region	Region name	AWS account ID
`eu-west-1`	Europe (Ireland)	694911143906
`us-east-1`	US East (N. Virginia)	593207742271
`us-east-2`	US East (Ohio)	733349766148
`eu-west-3`	Europe (Paris)	665651866788
`us-east-2`	US East (Ohio)	307168627858
`eu-central-1`	Europe (Frankfurt)	323658145986
`ap-northeast-2`	Asia Pacific (Seoul)	914738172881
`eu-north-1`	Europe (Stockholm)	591436053604
`ap-east-1`	Asia Pacific (Hong Kong)	258348409381
`me-south-1`	Middle East (Bahrain)	536382113932

eu-west-2	Europe (London)	892757235363
ap-northeast-1	Asia Pacific (Tokyo)	533107202818
ap-southeast-1	Asia Pacific (Singapore)	174946120834
ap-south-1	Asia Pacific (Mumbai)	251508486986
ap-southeast-3	Asia Pacific (Jakarta)	510637619217
sa-east-1	South America (São Paulo)	758426053663
ap-northeast-3	Asia Pacific (Osaka)	273192626886
eu-south-1	Europe (Milan)	266869475730
af-south-1	Africa (Cape Town)	197869348890
ap-southeast-2	Asia Pacific (Sydney)	005257825471
me-central-1	Middle East (UAE)	000014521398
us-west-1	US West (N. California)	684579721401
ca-central-1	Canada (Central)	354763396469
ap-south-2	Asia Pacific (Hyderabad)	950823858135
eu-south-2	Europe (Spain)	919611009337
eu-central-2	Europe (Zurich)	529164026651
ap-southeast-4	Asia Pacific (Melbourne)	251357961535
il-central-1	Israel (Tel Aviv)	870907303882

GuardDuty security agent for AWS Fargate (Amazon ECS only)

The following table shows the release version history for the GuardDuty security agent for Fargate (Amazon ECS only).

Agent version	Container image	Release notes	Availability date
v1.0.1	**x86_64 (AMD64)**: `sha256:9f8cd438fb66f62d09bfc641286439f7ed5177988a314a6021ef4ff880642e68` **Graviton (ARM64)**: `sha256:82c66bb615bd0d1e96db77b1f1fb51dc03220caa593b1962249571bf7147d1b7`	General performance tuning and enhancements	January 26, 2024
v1.0.0	**x86_64 (AMD64)**: `sha256:359b8b014e5076c625daa1056090e522631587a7afa3b2e055edda6bd1141017` **Graviton (ARM64)**: `sha256:b9438690fa8a86067180a11658bec0f4f838ae3fbd225d04b9306250648b3984`	Initial release of GuardDuty security agent for AWS Fargate (Amazon ECS only).	November 26, 2023

GuardDuty security agent for Amazon EKS clusters

The following table shows the release version history of Amazon EKS add-on GuardDuty agent.

Agent version	Container image	Release notes	Availability date	End of standard support[1]
v1.5.0	**x86_64 (AMD64)**: `sha256:66d491927763742660fa`	• General performan	March 07, 2024	–

Agent version	Container image	Release notes	Availability date	End of standard support[1]
	a87cc2c39bb97b7873039157ae8b90bc999cb73d0b9c **Graviton (ARM64)**: sha256:537a330b2dd82357024fb6daeb8761034b7defd43b10dffe0792c9e6d0778b40	ce tuning and enhancements. - Security enhancements including new event types under Collected runtime event types. - Performance enhancements around CPU usage.		
v1.4.1	**x86_64 (AMD64)**: sha256:66d491927763742660faa87cc2c39bb97b7873039157ae8b90bc999cb73d0b9c **Graviton (ARM64)**: sha256:537a330b2dd82357024fb6daeb8761034b7defd43b10dffe0792c9e6d0778b40	General performance tuning and enhancements.	January 16, 2024	–

Agent version	Container image	Release notes	Availability date	End of standard support[1]
v1.4.0	**x86_64 (AMD64)**: `sha256:848ce13d9430bad554ac23d4699551505326ada2a88e1a721fe9f86b56b52c0f` **Graviton (ARM64)**: `sha256:0c650aeafeeb5f2bcb8b989ac849bedc1fae1a4de1cf6306ffdd9c6aebe67f8e`	Manifest mount point support better data collection AppArmor configuration in manifest Collect command line argument General performance tuning and enhancements	December 21, 2023	–
v1.3.1	**x86_64 (AMD64)**: `sha256:55578fcb7b73097ade5c8404390ef16cf76a7b568490abaae01ac75992b3ea29` **Graviton (ARM64)**: `sha256:e3ce8d66ac2121f8d476eb58f8bc50ab51336647615eb7cf514c21421cb818fd`	Important security patches and updates.	October 23, 2023	–

Agent version	Container image	Release notes	Availability date	End of standard support[1]
v1.3.0	**x86_64 (AMD64)**: `sha256:6dace2337dfbb7609811be89fb4b23ae0b865f1027ad78fbe69530bfbd46c694` **Graviton (ARM64)**: `sha256:4928a7c6ef40e77c8ec95841323bb9a110db31f12c0ee7ab965e08b43efd01bb`	Supports Ubuntu platform Supports Kubernetes version 1.28 General performance enhancements and stability improvement.	October 05, 2023	–

Agent version	Container image	Release notes	Availability date	End of standard support[1]
v1.2.0	**x86_64 (AMD64)**: `sha256:d610413d662ec042057f05d6942496d7f2c08e9f5a077ea307ffdb5d3f11bcc3` **Graviton (ARM64)**: `sha256:174d7ab28b2f95e5309da80d95b88ad26f602dfe72c2b351a0ef9297a1412bfa`	In addition to AMD64-based instances, v1.2.0 now also supports ARM64-based instances. Added and verified support for Bottlerocket Supports Kubernetes version 1.27 General performance enhancements and stability improvements.	June 16, 2023	–

Agent version	Container image	Release notes	Availability date	End of standard support[1]
v1.1.0	`sha256:b19ba3a3c1a508d153263ae2fda891a7928b5ca9b3a5692db6c101829303281c`	In addition to Kubernetes versions supported by GuardDuty security agent, this agent release also supports Kubernetes version 1.26. General performance enhancements and stability improvements.	May 2, 2023	May 14, 2024
v1.0.0	`sha256:e38bdd2b1323e89113f1a31bd4bc8e5a8098525dd98e6981a28b9906b1e4411e`	Initial release of Amazon EKS add-on agent.	March 30, 2023	May 14, 2024

- [1] For information about updating your current agent version that is approaching to an end of standard support, see Updating security agent manually.

Amazon S3 Protection in Amazon GuardDuty

S3 Protection helps Amazon GuardDuty monitor AWS CloudTrail data events for Amazon Simple Storage Service (Amazon S3) that include object-level API operations to identify potential security risks for data within your Amazon S3 buckets.

GuardDuty monitors both AWS CloudTrail management events and AWS CloudTrail S3 data events to identify potential threats in your Amazon S3 resources. Both the data sources monitor different kinds of activities. Examples of CloudTrail management events for S3 include operations that list or configure Amazon S3 buckets, such as `ListBuckets`, `DeleteBuckets`, and `PutBucketReplication`. Examples of CloudTrail data events for S3 include object-level API operations, such as `GetObject`, `ListObjects`, `DeleteObject`, and `PutObject`.

When you enable Amazon GuardDuty for an AWS account, GuardDuty starts monitoring CloudTrail management events. You don't need to manually enable or configure S3 data event logging in AWS CloudTrail. You can enable the S3 Protection feature (that monitors CloudTrail data events for S3) for any account in any AWS Region where this feature is available within Amazon GuardDuty, at any time. An AWS account that has already enabled GuardDuty, can enable S3 Protection for the first time with a 30-day free trial period. For an AWS account that enables GuardDuty for the first time, S3 Protection is already enabled and included in this 30-day free trial. For more information, see Estimating GuardDuty cost.

We recommend you enable S3 Protection in GuardDuty. If this feature is not enabled, GuardDuty will not be able to fully monitor your Amazon S3 buckets or generate findings for suspicious access to the data stored in your S3 buckets.

How GuardDuty uses S3 data events

When you enable S3 data events (S3 Protection), GuardDuty begins to analyze S3 data events from all of your S3 buckets, and monitors them for malicious and suspicious activity. For more information, see AWS CloudTrail data events for S3.

When an unauthenticated user accesses an S3 object, it means that the S3 object is publicly accessible. Therefore, GuardDuty doesn't process such requests. GuardDuty processes the requests made to the S3 objects by using valid IAM (AWS Identity and Access Management) or AWS STS (AWS Security Token Service) credentials.

When GuardDuty detects a potential 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 buckets, see GuardDuty S3 finding types.

If you disable S3 Protection, GuardDuty stops S3 data event monitoring of the 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. For more information, see Configuring S3 Protection in multiple-account environments.

To enable or disable S3 Protection

Choose your preferred access method to configure S3 Protection for a standalone account.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **S3 Protection**.
3. The **S3 Protection** page provides the current status of S3 Protection for your account. Choose **Enable** or **Disable** to enable or disable S3 Protection at any point in time.
4. Choose **Confirm** to confirm your selection.

API/CLI

1. Run updateDetector by using your valid detector ID for the current Region and passing the `features` object name as S3_DATA_EVENTS set to ENABLED or DISABLED to enable or disable S3 Protection, respectively.

   > **Note**
   >
   > To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

2. Alternatively, you can use AWS Command Line Interface. To enable S3 Protection, run the following command and make sure to use your own valid detector ID.

```
aws guardduty update-detector --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
features '[{"Name" : "S3_DATA_EVENTS", "Status" : "ENABLED"}]'
```

To disable S3 Protection, replace `ENABLED` with `DISABLED` in the example.

Configuring S3 Protection in multiple-account environments

In a multi-account environment, only the delegated GuardDuty administrator account has the option to configure (enable or disable) S3 Protection for the member accounts in their AWS organization. The GuardDuty member accounts can't modify this configuration from their accounts. The delegated GuardDuty administrator account manages their member accounts using AWS Organizations. The delegated GuardDuty administrator account can choose to have S3 Protection automatically enabled on all accounts, only new accounts, or no accounts in the organization. For more information, see Managing accounts with AWS Organizations.

Configuring S3 Protection for delegated GuardDuty administrator account

Choose your preferred access method to configure S3 Protection for the delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the management account credentials.
2. In the navigation pane, choose **S3 Protection**.
3. On the **S3 Protection** page, choose **Edit**.
4. Do one of the following:

   Using Enable for all accounts

   - Choose **Enable for all accounts**. This will enable the protection plan for all the active GuardDuty accounts in your AWS organization, including the new accounts that join the organization.
   - Choose **Save**.

Using Configure accounts manually

- To enable the protection plan only for the delegated GuardDuty administrator account account, choose **Configure accounts manually**.
- Choose **Enable** under the **delegated GuardDuty administrator account (this account)** section.
- Choose **Save**.

API/CLI

Run updateDetector by using the detector ID of the delegated GuardDuty administrator account for the current Region and passing the `features` object `name` as `S3_DATA_EVENTS` and `status` as `ENABLED` or `DISABLED`.

Alternatively, you can configure S3 Protection by using AWS Command Line Interface. Run the following command, and make sure to replace *12abc34d567e8fa901bc2d34e56789f0* with the detector ID of the delegated GuardDuty administrator account for the current Region and *555555555555* with the AWS account ID of the delegated GuardDuty administrator account.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --account-ids 555555555555 --features '[{"Name": "RDS_LOGIN_EVENTS", "Status":
 "ENABLED"}]'
```

Auto-enable S3 Protection for all member accounts in the organization

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using your administrator account account.
2. Do one of the following:

Using the S3 Protection page

1. In the navigation pane, choose **S3 Protection**.
2. Choose **Enable for all accounts**. This action automatically enables S3 Protection for both existing and new accounts in the organization.
3. Choose **Save**.

> **Note**
>
> It may take up to 24 hours to update the configuration for the member accounts.

Using the Accounts page

1. In the navigation pane, choose **Accounts**.
2. On the **Accounts** page, choose **Auto-enable** preferences before **Add accounts by invitation**.
3. In the **Manage auto-enable preferences** window, choose **Enable for all accounts** under **S3 Protection**.
4. Choose **Save**.

If you can't use the **Enable for all accounts** option, see Selectively enable or disable S3 Protection in member accounts.

API/CLI

- To selectively enable or disable S3 Protection for your member accounts, invoke the updateMemberDetectors API operation using your own *`detector ID`*.
- The following example shows how you can enable S3 Protection for a single member account. Make sure to replace *`12abc34d567e8fa901bc2d34e56789f0`* with the `detector-id` of the delegated GuardDuty administrator account, and *`111122223333`*. To disable S3 Protection, replace `ENABLED` with `DISABLED`.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
 '[{"name": "S3_DATA_EVENTS", "status": "ENABLED"}]'
```

> **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.

Enable S3 Protection for all existing active member accounts

Choose your preferred access method to enable S3 Protection for all the existing active member accounts in your organization.

Console

1. Sign in to the AWS Management Console and open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Sign in using the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **S3 Protection**.
3. On the **S3 Protection** page, you can view the current status of the configuration. Under the **Active member accounts** section, choose **Actions**.
4. From the **Actions** dropdown menu, choose **Enable for all existing active member accounts**.
5. Choose **Confirm**.

API/CLI

- To selectively enable or disable S3 Protection for your member accounts, invoke the updateMemberDetectors API operation using your own *`detector ID`*.
- The following example shows how you can enable S3 Protection for a single member account. Make sure to replace *`12abc34d567e8fa901bc2d34e56789f0`* with the `detector-id`

of the delegated GuardDuty administrator account, and *111122223333*. To disable S3 Protection, replace ENABLED with DISABLED.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-member-detectors --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 111122223333 --features
 '[{"name": "S3_DATA_EVENTS", "status": "ENABLED"}]'
```

> **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.

Auto-enable S3 Protection for new member accounts

Choose your preferred access method to enable S3 Protection for new accounts that join your organization.

Console

The delegated GuardDuty administrator account can enable for new member accounts in an organization through the console, using either the **S3 Protection** or **Accounts** page.

To auto-enable S3 Protection for new member accounts

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. Do one of the following:

   - Using the **S3 Protection** page:

     1. In the navigation pane, choose **S3 Protection**.
     2. On the **S3 Protection** page, choose **Edit**.
     3. Choose **Configure accounts manually**.

4. Select **Automatically enable for new member accounts**. This step ensures that whenever a new account joins your organization, S3 Protection will be automatically enabled for their account. Only the organization delegated GuardDuty administrator account can modify this configuration.
5. Choose **Save**.

- Using the **Accounts** page:

  1. In the navigation pane, choose **Accounts**.
  2. On the **Accounts** page, choose **Auto-enable** preferences.
  3. In the **Manage auto-enable preferences** window, select **Enable for new accounts** under **S3 Protection**.
  4. Choose **Save**.

API/CLI

- To selectively enable or disable S3 Protection for your member accounts, invoke the UpdateOrganizationConfiguration 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, see Selectively enable or disable RDS Protection for member accounts. Set the preferences to automatically enable or disable the protection plan in that Region for new accounts (NEW) that join the organization, all the accounts (ALL), or none of the accounts (NONE) in the organization. For more information, see autoEnableOrganizationMembers. Based on your preference, you may need to replace NEW with ALL or NONE.

  To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty update-organization-configuration --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable --features '[{"Name":
   "S3_DATA_EVENTS", "autoEnable": "NEW"}]'
  ```

  > **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.

Selectively enable or disable S3 Protection in member accounts

Choose your preferred access method to selectively enable or disable S3 Protection for member accounts.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the delegated GuardDuty administrator account credentials.

2. In the navigation pane, choose **Accounts**.

   On the **Accounts** page, review the **S3 Protection** column for the status of your member account.

3. **To selectively enable or disable S3 Protection**

   Select the account for which you want to configure S3 Protection. You can select multiple accounts at a time. In the **Edit Protection Plans** dropdown menu, choose **S3Pro**, and then choose the appropriate option.

API/CLI

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`.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
 aws guardduty update-member-detectors --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --account-ids 123456789012 --features
 '[{"Name" : "S3_DATA_EVENTS", "Status" : "ENABLED"}]'
```

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 want 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 AWS accounts that join GuardDuty for the first time.

If you are a GuardDuty administrator account enabling GuardDuty for the first time on a new account and do not want S3 Protection enabled by default, you can disable it by modifying the createDetector API operation with the optional `features` object. The following example uses the AWS CLI to enable a new GuardDuty detector with the S3 Protection disabled.

```
aws guardduty create-detector --enable --features '[{"Name" : "S3_DATA_EVENTS",
"Status" : "DISABLED"}]'
```

Feature in S3 Protection

AWS CloudTrail data events for S3

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 data events for S3 that GuardDuty can monitor:

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

When you enable GuardDuty for the first time, S3 Protection is enabled by default and is also included in the 30-day free trial period. However, this feature is optional and you can choose to enable or disable it for any account or Region at any time. For more information about configuring Amazon S3 as a feature, see GuardDuty S3 Protection.

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 AWS CLI or API operations. For an overview of the ways you can manage findings see Managing Amazon GuardDuty findings.

Topics:

Finding details

Learn about the types of data available within GuardDuty findings.

Sample findings

Learn how to generate sample findings to test or better understand GuardDuty.

GuardDuty finding format

Understand the format of GuardDuty finding types and the different threat purposes tracked by GuardDuty.

Finding types

View and search all available GuardDuty finding by type. Each finding type entry includes an explanation of that finding as well as tips and suggestions for remediation.

Finding details

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

There are two primary details that determine what kind of information is available for any finding. The first is the resource type, which can be `Instance`, `AccessKey`, `S3Bucket`, `Kubernetes cluster`, `ECS cluster`, `Container`, `RDSDBInstance`, or `Lambda`. 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 overview

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

- **Account ID –** The ID of the AWS account in which the activity took place that prompted GuardDuty to generate this finding.
- **Count –** The number of times GuardDuty has aggregated an activity matching this pattern to this finding ID.
- **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.

- **Finding ID –** A unique identifier for this finding type and set of parameters. New occurrences of activity matching this pattern will be aggregated to the same ID.
- **Finding type –** A formatted string representing the type of activity that triggered the finding. For more information, see GuardDuty finding format.
- **Region –** The AWS Region in which the finding was generated. For more information about supported Regions, see Regions and endpoints
- **Resource ID –** The ID of the AWS resource against which the activity took place that prompted GuardDuty to generate this finding.
- **Scan ID –** Applicable to findings when GuardDuty Malware Protection is enabled, this is an identifier of the malware scan that runs on the EBS volumes attached to the potentially compromised EC2 instance or container workload. For more information, see Malware Protection finding details.
- **Severity –** A finding's assigned severity level of either High, Medium, or Low. For more information, see Severity levels for GuardDuty findings.
- **Updated at –** The last time this finding was updated with new activity matching the pattern that prompted GuardDuty to generate this finding.

Resource

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

Resource role – The role of the AWS resource that initiated 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 performed the suspicious activity.

Resource type – The type of the affected resource. If multiple resources were involved, a finding can include multiple resources types. The resource types are **Instance**, **AccessKey**, **S3Bucket**, **KubernetesCluster**, **ECSCluster**, **Container**, **RDSDBInstance**, and **Lambda**. 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 stopped 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 that the instance was launched.
- **Outpost ARN –** The Amazon Resource Name (ARN) of 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`.

AccessKey

Access Key details:

- **Access key ID –** The 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.

S3Bucket

Amazon 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 –** The 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**.

EKSCluster

Kubernetes cluster details:

- **Name –** The name of the Kubernetes cluster.
- **ARN –** The ARN that identifies the cluster.
- **Created At –** The time and date when this cluster was created.

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

- **VPC ID –** The ID of the VPC that is associated to your cluster.
- **Status –** The current status of the cluster.
- **Tags –** The metadata that you apply to the cluster to help you to categorize and organize them. Each tag consists of a key and an optional value, listed in the format `key:value`. You get to define both key and value.

  Cluster tags do not propagate to any other resource associated with the cluster.

Kubernetes workload details:

- **Type –** The type of Kubernetes workload, such as pod, deployment, and job.
- **Name –** The name of the Kubernetes workload.
- **Uid –** The unique ID of the Kubernetes workload.
- **Created at –** The time and date when this workload was created.
- **Labels –** The key-value pairs attached to the Kubernetes workload.
- **Containers –** The details of the container running as a part of Kubernetes workload.
- **Namespace –** The workload belongs to this Kubernetes namespace.
- **Volumes –** The volumes used by the Kubernetes workload.
  - **Host path –** Represents a preexisting file or directory on the host machine that the volume maps to.
  - **Name –** The name of the volume.

- **pod security context –** Defines the privilege and acess control settings for all containers in a pod.
- **Host network –** Set to `true` if the pods are included in the Kubernetes workload.

Kubernetes user details:

- **Groups –** Kubernetes RBAC (role-access based control) groups of the user involved in the activity that generated the finding.
- **ID –** Unique ID of the Kubernetes user.
- **Username –** Name of the Kubernetes user involved in the activity that generated the finding.
- **Session name –** Entity that assumed the IAM role with Kubernetes RBAC permissions.

ECSCluster

ECS cluster details:

- **ARN –** The ARN that identifies the cluster.
- **Name –** The name of the cluster.
- **Status –** The current status of the cluster.
- **Active services count –** The number of services that are running on the cluster in an `ACTIVE` state. You can view these services with ListServices
- **Registered container instances count –** The number of container instances registered into the cluster. This includes container instances in both `ACTIVE` and `DRAINING` status.
- **Running tasks count –** The number of tasks in the cluster that are in the `RUNNING` state.
- **Tags –** The metadata that you apply to the cluster to help you to categorize and organize them. Each tag consists of a key and an optional value, listed in the format `key:value`. You get to define both key and value.
- **Containers –** The details about the container that's associated with the task:
  - **Container name –** The name of the container.
  - **Container image –** The image of the container.
- **Task details –** The details of a task in a cluster.
  - **ARN –** The Amazon Resource Name (ARN) of the task.
  - **Definition ARN –** The Amazon Resource Name (ARN) of the task definition that creates the task.

- **Version –** The version counter for the task.
- **Task created at –** The Unix timestamp when the task was created.
- **Task started at –** The Unix timestamp when the task started.
- **Task started by –** The tag specified when a task is started.

Container

Container details:

- **Container runtime –** The container runtime (such as `docker` or `containerd`) used to run the container.
- **ID –** The container instance ID or full ARN entries for the container instance.
- **Name –** The name of the container.

  When available, this field displays the value of the label `io.kubenetes.container.name`.
- **Image –** The image of the container instance.
- **Volume mounts –** List of container volume mounts. A container can mount a volume under its file system.
- **Security context –** The container security context defines privilege and access control settings for a container.
- **Process details –** Describes the details of the process that is associated to the finding.

RDSDBInstance

RDSDBInstance details:

> **Note**
>
> This resource is available in RDS Protection findings related to the database instance.

- **Database Instance ID –** The identifier associated to the database instance that was involved in the GuardDuty finding.
- **Engine –** The database engine name of the database instance involved in the finding. Possible values are Aurora MySQL-Compatible or Aurora PostgreSQL-Compatible.

- **Engine version –** The version of the database engine that was involved in the GuardDuty finding.
- **Database cluster ID –** The identifier of the database cluster that contains the database instance ID involved in the GuardDuty finding.
- **Database instance ARN –** The ARN that identifies the database instance involved in the GuardDuty finding.

Lambda

Lambda function details

- **Function name –** The name of the Lambda function involved in the finding.
- **Function version –** The version of the Lambda function involved in the finding.
- **Function description –** A description of the Lambda function involved in the finding.
- **Function ARN –** The Amazon Resource Name (ARN) of the Lambda function involved in the finding.
- **Revision ID –** The revision ID of the Lambda function version.
- **Role –** The execution role of the Lambda function involved in the finding.
- **VPC configuration –** The Amazon VPC configuration, including the VPC ID, security group, and subnet IDs associated with your Lambda function.
- **VPC ID –** The ID of the Amazon VPC that is associated with the Lambda function involved in the finding.
- **Subnet IDs –** The ID of the subnets that are associated with your Lambda function.
- **Security Group –** The security group attached to the involved Lambda function. This includes the security group name and group ID.
- **Tags –** A list of tags attached to this resource, listed in the format of `key:value` pair.

RDS database (DB) user details

> **Note**
>
> This section is applicable to findings when you enable the RDS Protection feature in GuardDuty. For more information, see GuardDuty RDS Protection.

The GuardDuty finding provides the following user and authentication details of the potentially compromised database.

- **User –** The user name used to make the anomalous login attempt.
- **Application –** The application name used to make the anomalous login attempt.
- **Database –** The name of the database instance involved in the anomalous login attempt.
- **SSL –** The version of the Secure Socket Layer (SSL) used for the network.
- **Auth method –** The authentication method used by the user involved in the finding.

Runtime Monitoring finding details

> **Note**
>
> These details may be available only if GuardDuty generates one of the Runtime Monitoring finding types.

This section contains the runtime details such as process details and any required context. Process details describe information about the observed process and runtime context describes any additional information about the potentially suspicious activity.

Process details

- **Name –** The name of the process.
- **Executable path –** The absolute path of the process executable file.
- **Executable SHA-256 –** The SHA256 hash of the process executable.
- **Namespace PID –** The process ID of the process in a secondary PID namespace other than the host level PID namespace. For processes inside a container, it is the process ID observed inside the container.
- **Present working directory –** The present working directory of the process.
- **Process ID –** The ID assigned to the process by operating system.
- **startTime –** The time when the process started. This is in UTC date string format (`2023-03-22T19:37:20.168Z`).
- **UUID –** The unique ID assigned to the process by GuardDuty.

- **Parent UUID –** The unique ID of the parent process. This ID is assigned to the parent process by GuardDuty.
- **User –** The user that executed the process.
- **User ID –** The ID of the user that executed the process.
- **Effective user ID –** The effective user ID of the process at the time of the event.
- **Lineage –** Information about the ancestors of the process.
  - **Process ID –** The ID assigned to the process by operating system.
  - **UUID –** The unique ID assigned to the process by GuardDuty.
  - **Executable path –** The absolute path of the process executable file.
  - **Effective user ID –** The effective user ID of the process at the time of the event.
  - **Parent UUID –** The unique ID of the parent process. This ID is assigned to the parent process by GuardDuty.
  - **Start Time –** The time when the process started.
  - **Namespace PID –** The process ID of the process in a secondary PID namespace other than the host level PID namespace. For processes inside a container, it is the process ID observed inside the container.
  - **User ID –** The user ID of the user that executed the process.
  - **Name –** Name of the process.

Runtime context

From the following fields, a generated finding may include only those fields that are relevant to the finding type.

- **Mount Source –** The path on the host that is mounted by the container.
- **Mount Target –** The path in the container that is mapped to the host directory.
- **Filesystem Type –** Represents the type of the mounted filesystem.
- **Flags –** Represents options that control the behavior of the event involved in this finding.
- **Modifying Process –** Information about the process that created or modified a binary, script, or a library, inside a container at runtime.
- **Modified At –** The timestamp at which the process created or modified a binary, script, or library inside a container at runtime. This field is in the UTC date string format (`2023-03-22T19:37:20.168Z`).

- **Library Path –** The path to the new library that was loaded.
- **LD Preload Value –** The value of the `LD_PRELOAD` environment variable.
- **Socket Path –** The path to the Docker socket that was accessed.
- **Runc Binary Path –** The path to the `runc` binary.
- **Release Agent Path –** The path to the `cgroup` release agent file.
- **Command Line Example –** The example of the command line involved in the potentially suspicious activity.
- **Tool Category –** Category that the tool belongs to. Some of the examples are Backdoor Tool, Pentest Tool, Network Scanner, and Network Sniffer.
- **Tool Name –** The name of the potentially suspicous tool.
- **Script Path –** The path to the executed script that generated the finding.
- **Threat File Path –** The suspicious path for which the threat intelligence details were found.
- **Service Name –** The name of the security service that has been disabled.

EBS volumes scan details

Note

This section is applicable to findings when you turn on the GuardDuty-initiated malware scan in GuardDuty Malware Protection.

The EBS volumes scan provides details about the EBS volume attached to the potentially compromised EC2 instance or container workload.

- **Scan ID –** The identifier of the malware scan.
- **Scan started at –** The date and time when the malware scan started.
- **Scan completed at –** The date and time when the malware scan completed.
- **Trigger Finding ID –** The finding ID of the GuardDuty finding that initiated this malware scan.
- **Sources –** The possible values are `Bitdefender` and `AWS`.
- **Scan detections –** The complete view of details and results for each malware scan.
    - **Scanned item count –** The total number of scanned files. It provides details such as `totalGb`, `files`, and `volumes`.

- **Threats detected item count** – The total number of malicious `files` detected during the scan.
- **Highest severity threat details** – The details of the highest severity threat detected during the scan and the number of malicious files. It provides details such as `severity`, `threatName`, and `count`.
- **Threats detected by Name** – The container element grouping threats of all severity levels. It provides details such as `itemCount`, `uniqueThreatNameCount`, `shortened`, and `threatNames`.

Malware Protection finding details

> **Note**
>
> This section is applicable to findings when you turn on the GuardDuty-initiated malware scan in GuardDuty Malware Protection.

When the Malware Protection scan detects malware, you can view the scan details by selecting the corresponding finding on the **Findings** page in the https://console.aws.amazon.com/guardduty/ console. The severity of your Malware Protection finding depends on the severity of the GuardDuty finding.

> **Note**
>
> The `GuardDutyFindingDetected` tag specifies that the snapshots contains malware.

The following information is available under the **Threats detected** section in the details panel.

- **Name** – The name of the threat, obtained by grouping the files by detection.
- **Severity** – The severity of the threat detected.
- **Hash** – The SHA-256 of the file.
- **File path** – The location of the malicious file in the EBS volume.
- **File name** – The name of the file in which the threat was detected.
- **Volume ARN** – The ARN of the scanned EBS volumes.

The following information is available under the **Malware scan details** section in the details panel.

- **Scan ID –** The scan ID of the malware scan.
- **Scan started at –** The date and time when the scan started.
- **Scan completed at –** The date and time when the scan completed.
- **Files scanned –** The total number of scanned files and directories.
- **Total GB scanned –** The amount of storage scanned during the process.
- **Trigger finding ID –** The finding ID of the GuardDuty finding that initiated this malware scan.
- The following information is available under the **Volume details** section in the details panel.
  - **Volume ARN –** The Amazon Resource Name (ARN) of the volume.
  - **SnapshotARN –** The ARN of the snapshot of the EBS volume.
  - **Status –** The scan status of the volume, such as `Running`, `Skipped`, and `Completed`.
  - **Encryption type –** The type of encryption used to encrypt the volume. For example, `CMCMK`.
  - **Device name –** The name of the device. For example, `/dev/xvda`.

Action

A finding's **Action** gives details about 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**, **AWS_API_CALL**, or **RDS_LOGIN_ATTEMPT**. 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 address 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 address 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.
- **RDS_LOGIN_ATTEMPT –** Indicates that a login attempt was made to the potentially compromised database from a remote IP address.
  - **IP address –** The remote IP address that was used to make the potentially suspicious login attempt.

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 about 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:

- **Affiliated –** Details about whether the AWS account of the remote API caller is related to your GuardDuty environment. If this value is `true`, the API caller is affiliated to your account in some manner; if `false`, the API caller is from outside your environment.
- **Remote Account ID –** The account ID that owns the outbound IP address that was used to access the resource at the final network.
- **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.

- **Domain with suffix** – The second- and top-level domain involved in an activity that potentially prompted GuardDuty to generate the finding. For a list of top-level and second-level domains, see public suffix list.

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, location, time, bucket, login behavior, or ASN Org.
- **Unusual protocol** – The network connection protocol involved in the activity that prompted GuardDuty to generate the finding.
- **Agent details** – Details about the security agent that is currently deployed on the EKS cluster in your AWS account. This is only applicable to EKS Runtime Monitoring finding types.
  - **Agent version** – The version of the GuardDuty security agent.
  - **Agent Id** – The unique identifier of the GuardDuty security agent.

Evidence

Findings based on threat intelligence have an **Evidence** section that includes the following information:

- **Threat intelligence details** – The name of the threat list on which the recognized `Threat name` appears.
- **Threat name** – The name of the malware family or other identifier that is associated with the threat.
- **Threat file SHA256** – SHA256 of the file that generated the finding.

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, and the 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`, and `operation not permitted`.
  - APIs are categorized by their associated service.

> **Note**
>
> For more context, choose **Historical APIs** to view the details about 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 about the profiled behavior for your account. The information tracked in this panel includes:
  - **ASN Org –** The ASN Org that 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`.
  - **Bucket –** The name of the S3 bucket that is being accessed.
- **Unusual Behavior (User Identity) –** This section gives additional details about the profiled behavior for the **User Identity** involved with the finding. When a behavior isn't identified as historical, this means the GuardDuty ML model hasn't 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`.
  - **Bucket –** The name of the S3 bucket that is being accessed.
- **Unusual Behavior (Bucket) –** This section gives additional details about the profiled behavior for the S3 bucket associated with the finding. When a behavior isn't identified as historical, this means the GuardDuty ML model hasn't previously seen API calls made to this bucket in this way within the training period. The information tracked in this section 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`.

  > **Note**
  >
  > For more context on historical behaviors, choose **Historical behavior** in either **Unusual behavior (Account)**, **User ID**, or **Bucket** section to view details about the expected behavior in your 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.

- **Unusual Behavior (Database)** – This section provides additional details about the profiled behavior for the database instance associated with the finding. When a behavior isn't identified as historical, it means that the GuardDuty ML model hasn't previously seen a login attempt made to this database instance in this way within the training period. The information tracked for this section in the finding panel includes:
  - **User name** – The user name used to make the anomalous login attempt.
  - **ASN Org** – The ASN Org that the anomalous login attempt was made from.
  - **Application name** – The application name used to make the anomalous login attempt.
  - **Database name** – The name of the database instance involved in the anomalous login attempt.

  > **Note**
  >
  > The **Historical behavior** section provides more context on the previously observed **User names**, **ASN Orgs**, **Application names**, and **Database names** for the associated database. Each unique value has an associated count representing the number of times this value was observed in a successful login event.

- **Unusual behavior (Account Kubernetes cluster, Kubernetes namespace, and Kubernetes username)** – This section provides additional details about the profiled behavior for the Kubernetes cluster and namespace associated with the finding. When a behavior isn't identified as historical, it means that the GuardDuty ML model hasn't previously observed this account, cluster, namespace, or username in this way. The information tracked for this section in the finding panel includes:

- **Username –** The user that called the Kubernetes API associated with the finding.
- **Impersonated Username –** The user being impersonated by `username`.
- **Namespace –** The Kubernetes namespace within the Amazon EKS cluster where the action occurred.
- **User Agent –** The user agent associated with the Kubernetes API call. The user agent is the method used to make the call such as `kubectl`.
- **API –** The Kubernetes API called by `username` within the Amazon EKS cluster.
- **ASN Information –** The ASN information, such as Organization and ISP, associated with the IP address of the user making this call.
- **Day of week –** The day of the week when the Kubernetes API call was made.
- **Permission**[1] – The Kubernetes verb and resource being checked for access to indicate whether or not the `username` can use the Kubernetes API.
- **Service Account Name**[1] – The service account associated with the Kubernetes workload that provides an identity to the workload.
- **Registry[1]** – The container registry associated with the container image that is deployed in the Kubernetes workload.
- **Image[1]** – The container image, without the associated tags and digest, that is deployed in the Kubernetes workload.
- **Image Prefix Config[1]** – The image prefix with the container and workload security configuration enabled, such as `hostNetwork` or `privileged`, for the container using the image.
- **Subject Name[1]** – The subjects, such as a `user`, `group`, or `serviceAccountName` that is bound to a reference role in a `RoleBinding` or `ClusterRoleBinding`.
- **Role Name[1]** – The name of the role that is involved in creation or modification of roles or the `roleBinding` API.

S3 volume-based anomalies

This section details the contextual information for S3 volume-based anomalies. The volume-based finding (Exfiltration:S3/AnomalousBehavior) monitors for unusual numbers of S3 API calls made to the S3 buckets by users, indicating potential data exfiltration. The following S3 API calls are monitored for volume-based anomaly detection.

- `GetObject`
- `CopyObject.Read`
- `SelectObjectContent`

The following metrics would help to build a baseline of usual behavior when an IAM entity accesses an S3 bucket. To detect data exfiltration, volume-based anomaly detection finding evaluates all the activities against the usual behavioral baseline. Choose **Historical behavior** in the **Unusual behavior (User Identity)**, **Observed Volume (User Identity)**, and **Observed Volume (Bucket)** sections to view the following metrics, respectively.

- Number of `s3-api-name` API calls invoked by the IAM user or IAM role (depends on which one was issued) associated with the affected S3 bucket over the past 24 hours.
- Number of `s3-api-name` API calls invoked by the IAM user or IAM role (depends on which one was issued) associated with all S3 buckets over the past 24 hours.
- Number of `s3-api-name` API calls across all IAM user or IAM role (depends on which one was issued) associated with the affected S3 bucket over the past 24 hours.

RDS login activity-based anomalies

This section details the count of login attempts performed by the unusual actor and is grouped by the result of the login attempts. The RDS Protection finding types identify anomalous behavior by monitoring the login events for unusual patterns of `successfulLoginCount`, `failedLoginCount`, and `incompleteConnectionCount`.

- **successfulLoginCount** – This counter represents the sum of successful connections (correct combination of login attributes) made to the database instance by the unusual actor. Login attributes include user name, password, and database name.
- **failedLoginCount** – This counter represents the sum of failed (unsuccessful) login attempts made to establish a connection to the database instance. This indicates that one or more attributes of the login combination, such as user name, password, or database name were incorrect.
- **incompleteConnectionCount** – This counter represents the number of connection attempts that can't be classified as successful or failed. These connections are closed before the database provides a response. For example, port scanning where the database port is connected but no

piece of information is sent to the database, or the connection was aborted before the login completed in a successful or failed attempt.

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 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, S3, IAM, and EKS 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.

Execution

This value indicates that GuardDuty has detected that an adversary may try to run malicious code to explore the network or steal data. 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 table.

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

Generating sample findings through the GuardDuty console or API

Choose your preferred access method to generate sample findings.

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/CLI

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 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.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty create-sample-findings --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --finding-types Backdoor:EC2/DenialOfService.Tcp
```

The title of sample findings generated through these methods always begins with **[SAMPLE]** in the console. 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.

   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 1.0 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 between 9.0 and 10.0 are reserved for future use.

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

<table>
<tr><th>Severity level</th><th>Value range</th></tr>
<tr><td><b>High</b></td><td>7.0 - 8.9</td></tr>
<tr><td colspan="2">A High severity level indicates that the resource in question (an EC2 instance or a set of IAM user sign-in 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 for more details.</td></tr>
<tr><td><b>Medium</b></td><td>4.0 - 6.9</td></tr>
<tr><td colspan="2">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 to secure the resource.</td></tr>
</table>

Severity level	Value range
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	1.0 - 3.9
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 behavior 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 `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.
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.
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

In some cases, GuardDuty becomes aware that certain findings are false positives after they have been generated. GuardDuty provides a **Confidence** field in the finding's JSON, and sets its value to zero. This way GuardDuty lets you know that you can safely ignore such findings.

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.

For information about finding types which are now retired, see Retired finding types.

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 Foundational data sources.

> **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 potentially compromised Amazon EC2 instance.

Topics

- Backdoor:EC2/DenialOfService.UdpOnTcpPorts
- Backdoor:EC2/DenialOfService.UnusualProtocol
- Backdoor:EC2/Spambot
- Behavior:EC2/NetworkPortUnusual
- Behavior:EC2/TrafficVolumeUnusual
- CryptoCurrency:EC2/BitcoinTool.B
- CryptoCurrency:EC2/BitcoinTool.B!DNS
- DefenseEvasion:EC2/UnusualDNSResolver
- DefenseEvasion:EC2/UnusualDoHActivity
- DefenseEvasion:EC2/UnusualDoTActivity
- Impact:EC2/AbusedDomainRequest.Reputation
- Impact:EC2/BitcoinDomainRequest.Reputation
- Impact:EC2/MaliciousDomainRequest.Reputation
- Impact:EC2/PortSweep
- Impact:EC2/SuspiciousDomainRequest.Reputation
- Impact:EC2/WinRMBruteForce
- Recon:EC2/PortProbeEMRUnprotectedPort
- Recon:EC2/PortProbeUnprotectedPort
- Recon:EC2/Portscan
- Trojan:EC2/BlackholeTraffic
- Trojan:EC2/BlackholeTraffic!DNS
- Trojan:EC2/DGADomainRequest.B
- Trojan:EC2/DGADomainRequest.C!DNS
- Trojan:EC2/DNSDataExfiltration
- Trojan:EC2/DriveBySourceTraffic!DNS
- Trojan:EC2/DropPoint
- Trojan:EC2/DropPoint!DNS
- Trojan:EC2/PhishingDomainRequest!DNS
- UnauthorizedAccess:EC2/MaliciousIPCaller.Custom
- UnauthorizedAccess:EC2/MetadataDNSRebind

- UnauthorizedAccess:EC2/RDPBruteForce
- UnauthorizedAccess:EC2/SSHBruteForce
- UnauthorizedAccess:EC2/TorClient
- UnauthorizedAccess:EC2/TorRelay

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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.

If this activity is unexpected, your instance is likely compromised, see Remediating a potentially compromised Amazon EC2 instance.

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.

If this activity is unexpected, your instance is likely compromised, see Remediating a potentially compromised Amazon EC2 instance.

DefenseEvasion:EC2/UnusualDNSResolver

An Amazon EC2 instance is communicating with an unusual public DNS resolver.

Default severity: Medium

- **Data source:** VPC flow logs

This finding informs you that the listed Amazon EC2 instance in your AWS environment is behaving in a way that deviates from the baseline behavior. This EC2 instance has no recent history of communicating with this public DNS resolver. The **Unusual** field in the finding details panel in the GuardDuty console can provide information about the queried DNS resolver.

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

DefenseEvasion:EC2/UnusualDoHActivity

An Amazon EC2 instance is performing an unusual DNS over HTTPS (DoH) communication.

Default severity: Medium

- **Data source:** VPC flow logs

This finding informs you that the listed Amazon EC2 instance within your AWS environment is behaving in a way that deviates from the established baseline. This EC2 instance doesn't have any recent history of DNS over HTTPS (DoH) communications with this public DoH server. The **Unusual** field in the finding details can provide information about the queried DoH server.

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

DefenseEvasion:EC2/UnusualDoTActivity

An Amazon EC2 instance is performing an unusual DNS over TLS (DoT) communication.

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 doesn't have any recent history of DNS over TLS (DoT) communications with this public DoT server. The **Unusual** field in the finding details panel can provide information about the queried DoT server.

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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.

If this activity is unexpected, your instance is likely compromised, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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. In the finding details panel in your GuardDuty console, only the most recent remote IP address gets displayed

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 a 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. This finding also informs that known scanners on the Internet are actively probing this port. 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 that is being probed, is used by Elasticsearch (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.

GuardDuty doesn't generate this finding for ports 443 and 80.

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.

If this activity is unexpected, your instance is likely compromised, see Remediating a potentially compromised Amazon EC2 instance.

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.

If this activity is unexpected, your instance is likely compromised, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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, 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.

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 potentially compromised Amazon EC2 instance.

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.

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 potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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, see Foundational data sources.

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. For information about remediating findings, see Remediating potentially compromised AWS credentials.

Topics

- UnauthorizedAccess:IAMUser/TorIPCaller

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, usernames, 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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

PenTest:IAMUser/KaliLinux

An API was invoked from a Kali Linux 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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

Policy:IAMUser/RootCredentialUsage

An API was invoked using root user sign-in credentials.

Default severity: Low

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

This finding informs you that the root user sign-in 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 user sign-in 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 user sign-in 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 about S3 threat detection, see S3 protection.

Remediation recommendations:

If this activity is unexpected, your credentials may be compromised. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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.

> **Note**
>
> If GuardDuty observes continued activity from a remote account, its machine learning (ML) model will identify this as an expected behavior. Therefore, GuardDuty will stop generating this finding for activity from that remote account. GuardDuty will continue to generate findings for new behavior from other remote accounts and will re-evaluate learned remote accounts as the behavior changes over time.

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 potentially compromised Amazon EC2 instance 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.

> **Note**
>
> If GuardDuty observes continued activity from a remote account, its machine learning (ML) model will identify this as an expected behavior. Therefore, GuardDuty will stop generating this finding for activity from that remote account. GuardDuty will continue to generate findings for new behavior from other remote accounts and will re-evaluate learned remote accounts as the behavior changes over time.

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.

> **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 potentially compromised AWS credentials.

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, or 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. For more information, see Remediating potentially compromised AWS credentials.

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, or 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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

Kubernetes audit logs 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 EKS Audit Log Monitoring findings.

> **Note**
>
> If the activity because of which these findings get generated is expected, consider adding Suppression rules to prevent future alerts.

Topics

- Execution:Kubernetes/AnomalousBehavior.ExecInPod
- PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer
- Persistence:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount
- Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed
- PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated
- Discovery:Kubernetes/AnomalousBehavior.PermissionChecked

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 Security best practices for Amazon EKS in the *Amazon EKS User 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

- **Feature:** 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, usernames, 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, usernames, 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 and revoke the permissions, if needed, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, usernames, 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, usernames, and access keys for your Kubernetes cluster. 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

DefenseEvasion:Kubernetes/MaliciousIPCaller

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

Default severity: High

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

DefenseEvasion:Kubernetes/SuccessfulAnonymousAccess

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

Default severity: High

- **Feature:** 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

For more information, see Remediating EKS Audit Log Monitoring findings.

DefenseEvasion:Kubernetes/TorIPCaller

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

Default severity: High

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Discovery:Kubernetes/MaliciousIPCaller

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

Default severity: Medium

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Discovery:Kubernetes/MaliciousIPCaller.Custom

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Discovery:Kubernetes/SuccessfulAnonymousAccess

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

Default severity: Medium

- **Feature:** 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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 APIand revoke the permissions, if needed, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Execution:Kubernetes/ExecInKubeSystemPod

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

Default severity: Medium

- **Feature:** Kubernetes audit logs

This finding 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Impact:Kubernetes/SuccessfulAnonymousAccess

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

Default severity: High

- **Feature:** 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Persistence:Kubernetes/ContainerWithSensitiveMount

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

Default severity: Medium

- **Feature:** 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

For more information, see Remediating EKS Audit Log Monitoring findings.

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

- **Feature:** 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, by following the instructions in Security best practices for

Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Policy:Kubernetes/AdminAccessToDefaultServiceAccount

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

Default severity: High

- **Feature:** 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Policy:Kubernetes/AnonymousAccessGranted

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

Default severity: High

- **Feature:** 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. For more information, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*. 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. For more information, see Remediating EKS Audit Log Monitoring findings.

Policy:Kubernetes/ExposedDashboard

The dashboard for a Kubernetes cluster was exposed to the internet

Default severity: Medium

- **Feature:** 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.

For more information, see Remediating EKS Audit Log Monitoring findings.

Policy:Kubernetes/KubeflowDashboardExposed

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

Default severity: Medium

- **Feature:** 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.

For more information, see Remediating EKS Audit Log Monitoring findings.

PrivilegeEscalation:Kubernetes/PrivilegedContainer

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

Default severity: Medium

- **Feature:** 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. For more information, see Remediating EKS Audit Log Monitoring findings.

CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed

A Kubernetes API commonly used to access secrets was invoked in an anomalous way.

Default severity: Medium

- **Feature:** Kubernetes audit logs

This finding informs you that an anomalous API operation to retrieve sensitive cluster secrets was invoked by a Kubernetes user in your cluster. The observed API is commonly associated with credential access tactics that can lead to privileged escalation and further access within your cluster. If this behavior is not expected, it may indicate either a configuration mistake or that your AWS credentials are compromised.

The observed API was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API activity within your EKS cluster and identifies anomalous events that are associated with techniques used by unauthorized users. The ML model tracks multiple factors of the API operation such as the user making the request, the location the request was made from, user agent used, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

Examine the permissions granted to the Kubernetes user in your cluster and ensure that all these permissions are needed. If the permissions were granted mistakenly or maliciously, revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated

A RoleBinding or ClusterRoleBinding to an overly permissive role or sensitive namespace was created or modified in your Kubernetes cluster.

Default severity: Medium*

> **Note**
>
> This finding's default severity is Medium. However, if a RoleBinding or ClusterRoleBinding involves the ClusterRoles `admin` or `cluster-admin`, the severity is High.

- **Feature:** Kubernetes audit logs

This finding informs you that a user in your Kubernetes cluster created a `RoleBinding` or `ClusterRoleBinding` to bind a user to a role with admin permissions or sensitive namespaces. If this behavior is not expected, it may indicate either a configuration mistake or that your AWS credentials are compromised.

The observed API was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API activity within your EKS cluster. This ML model also identifies anomalous events that are associated with the techniques used by an unauthorized user. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

Examine the permissions granted to the Kubernetes user. These permissions are defined in the role and subjects involved in `RoleBinding` and `ClusterRoleBinding`. If the permissions were granted mistakenly or maliciously, revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

Execution:Kubernetes/AnomalousBehavior.ExecInPod

A command was executed inside a pod in an anomalous way.

Default severity: Medium

- **Feature:** Kubernetes audit logs

This finding informs you that a command was executed in a pod using the Kubernetes exec API. The Kubernetes exec API allows running arbitrary commands in a pod. If this behavior is not expected for the user, namespace, or pod, it may indicate either a configuration mistake or that your AWS credentials are compromised.

The observed API was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API activity within your EKS cluster. This ML model also identifies anomalous events that are associated with the techniques used by an unauthorized user. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

If the execution of this command is unexpected, the credentials of the user identity used to execute the command may have been compromised. Revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer

A workload was launched with a privileged container in an anomalous way.

Default severity: High

- **Feature:** Kubernetes audit logs

This finding informs you that a workload was launched with a privileged container in your Amazon EKS cluster. A privileged container has root level access to the host. Unauthorized users can launch privileged containers as a privilege escalation tactic to first gain access to the host and then compromise it.

The observed container creation or modification was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API and container image activity within your EKS cluster. This ML model also identifies anomalous events that are associated with the techniques used by an unauthorized user. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, container images observed in your account, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

If this container launch is unexpected, the credentials of the user identity used to launch the container may have been compromised. Revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

If this container launch is expected, it is recommended that you use a suppression rule with a filter criteria based on the `resource.KubernetesDetails.KubernetesWorkloadDetails.containers.imagePrefix` field. In the filter criteria, the `imagePrefix` field must have the same value as the `imagePrefix` field specified in the finding. For more information, see Suppression rules.

Persistence:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount

A workload was deployed in an anomalous way, with a sensitive host path mounted inside the workload.

Default severity: High

- **Feature:** Kubernetes audit logs

This finding informs you that a workload was launched with a container that included a sensitive host path in the `volumeMounts` section. This potentially makes the sensitive host path accessible and writable from inside the container. This technique is commonly used by unauthorized users to gain access to the host's file system.

The observed container creation or modification was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API and container image activity within your EKS cluster. This ML model also identifies anomalous events that are associated with the techniques used by an unauthorized user. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, container images observed in your account, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

If this container launch is unexpected, the credentials of the user identity used to launch the container may have been compromised. Revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

If this container launch is expected, it is recommended that you use a suppression rule with a filter criteria based on the `resource.KubernetesDetails.KubernetesWorkloadDetails.containers.imagePrefix` field. In the filter criteria, the `imagePrefix` field must have the same value as the `imagePrefix` field specified in the finding. For more information, see Suppression rules.

Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed

A workload was launched in an anomalous way.

Default severity: Low*

> **Note**
>
> The default severity is Low. However, if the workload contains a potentially suspicious image name, such as a known pentest tool, or a container running a potentially suspicious

command at launch, such as reverse shell commands, then the severity of this finding type will be considered as Medium.

- **Feature:** Kubernetes audit logs

This finding informs you that a Kubernetes workload was created or modified in an anomalous way, such as an API activity, new container images, or risky workload configuration, within your Amazon EKS cluster. Unauthorized users can launch containers as a tactic to execute arbitrary code to first gain access to the host and then compromise it.

The observed container creation or modification was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API and container image activity within your EKS cluster. This ML model also identifies anomalous events that are associated with the techniques used by an unauthorized user. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, container images observed in your account, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

If this container launch is unexpected, the credentials of the user identity used to launch the container may have been compromised. Revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

If this container launch is expected, it is recommended that you use a suppression rule with a filter criteria based on the `resource.KubernetesDetails.KubernetesWorkloadDetails.containers.imagePrefix` field. In the filter criteria, the `imagePrefix` field must have the same value as the `imagePrefix` field specified in the finding. For more information, see Suppression rules.

PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated

A highly permissive Role or ClusterRole was created or modified in an anomalous way.

Default severity: Low

- **Feature:** Kubernetes audit logs

This finding informs you that an anomalous API operation to create a `Role` or `ClusterRole` with excessive permissions was called by a Kubernetes user in your Amazon EKS cluster. Actors can use role creation with powerful permissions to avoid using built-in admin-like roles and avoid detection. The excessive permissions can lead to privileged escalation, remote code execution, and potentially control over a namespace or cluster. If this behavior is not expected, it may indicate either a configuration mistake or that your credentials are compromised.

The observed API was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API activity within your Amazon EKS cluster and identifies anomalous events that are associated with the techniques used by unauthorized users. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, the user agent used, container images observed in your account, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

Examine the permissions defined in `Role` or `ClusterRole` to ensure that all the permissions are needed and follow least privilege principles. If the permissions were granted mistakenly or maliciously, revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

Discovery:Kubernetes/AnomalousBehavior.PermissionChecked

A user checked their access permission in an anomalous way.

Default severity: Low

- **Feature:** Kubernetes audit logs

This finding informs you that a user in your Kubernetes cluster successfully checked whether or not the known powerful permissions that can lead to privileged escalation and remote code execution, are allowed. For example, a common command used to check permissions for a user is `kubectl auth can-i`. If this behavior is not expected, it may indicate either a configuration mistake or that your credentials have been compromised.

The observed API was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all user API activity within your Amazon EKS cluster and identifies anomalous events that are associated with the techniques used by unauthorized users. The ML model also tracks multiple factors of the API operation, such as the user making the request, the location the request was made from, permission being checked, and the namespace that the user operated. You can find the details of the API request that are unusual, in the finding details panel in the GuardDuty console.

Remediation recommendations:

Examine the permissions granted to the Kubernetes user to ensure that all the permissions are needed. If the permissions were granted mistakenly or maliciously, revoke user access and reverse any changes made by an unauthorized user to your cluster. For more information, see Remediating EKS Audit Log Monitoring findings.

If your AWS credentials are compromised, see Remediating potentially compromised AWS credentials.

Lambda Protection finding types

This section describes the finding types that are specific to your AWS Lambda resources and have the `resourceType` listed as `Lambda`. For all Lambda findings, we recommend that you examine the resource in question and determine if it is behaving in an expected manner. If the activity is authorized, you can use Suppression rules or Trusted IP and threat lists to prevent false positive notifications for that resource.

If the activity is unexpected, the security best practice is to assume that Lambda has been potentially compromised and follow the remediation recommendations.

Topics

Backdoor:Lambda/C&CActivity.B

A Lambda function is querying an IP address that is associated with a known command and control server.

Default severity: High

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a listed Lambda function within your AWS environment is querying an IP address that is associated with a known command and control (C&C) server. The Lambda function associated to the generated finding is potentially compromised. C&C 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 is 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.

Remediation recommendations:

If this activity is unexpected, your Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

CryptoCurrency:Lambda/BitcoinTool.B

A Lambda function is querying an IP address that is associated with a cryptocurrency-related activity.

Default severity: High

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that the listed Lambda function in your AWS environment is querying an IP address that is associated with a Bitcoin or other cryptocurrency-related activity. Threat actors may seek to take control over Lambda functions in order to maliciously repurpose them for unauthorized cryptocurrency mining.

Remediation recommendations:

If you use this Lambda function to mine or manage cryptocurrency, or this function is otherwise involved in a blockchain activity, it is potentially an 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 criterion should use the finding type attribute with a value of CryptoCurrency:Lambda/BitcoinTool.B. The second filter criterion should be the Lambda function name of the function involved in blockchain activity. For information about creating suppression rules, see Suppression rules.

If this activity is unexpected, your Lambda function is potentially compromised. For more information, see Remediating a potentially compromised Lambda function.

Trojan:Lambda/BlackholeTraffic

A Lambda function is attempting to communicate with an IP address of a remote host that is a known black hole.

Default severity: Medium

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a listed Lambda function within your AWS environment is trying to communicate with an IP address of a black hole (or a 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. The listed Lambda function is potentially compromised.

Remediation recommendations:

If this activity is unexpected, your Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

Trojan:Lambda/DropPoint

A Lambda function 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

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a listed Lambda function within 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 Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

UnauthorizedAccess:Lambda/MaliciousIPCaller.Custom

A Lambda function is making connections to an IP address on a custom threat list.

Default severity: Medium

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a Lambda function 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 the uploaded threat lists. You can view the details of the threat list in the finding details on the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

UnauthorizedAccess:Lambda/TorClient

A Lambda function is making connections to a Tor Guard or an Authority node.

Default severity: High

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a Lambda function 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 node act as initial gateways into a Tor network. This traffic can indicate that this Lambda function has been potentially compromised. It is now acting as a client on a Tor network.

Remediation recommendations:

If this activity is unexpected, your Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

UnauthorizedAccess:Lambda/TorRelay

A Lambda function is making connections to a Tor network as a Tor relay.

Default severity: High

- **Feature:** Lambda Network Activity Monitoring

This finding informs you that a Lambda function 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 enables anonymous communication by forwarding the client's potentially illicit traffic from one Tor relay to another.

Remediation recommendations:

If this activity is unexpected, your Lambda function may be compromised. For more information, see Remediating a potentially compromised Lambda function.

Malware Protection finding types

GuardDuty Malware Protection provides a single Malware Protection finding for all threats detected during the scan of an EC2 instance or a container workload. The finding includes the total number of detections made during the scan, and based on the severity, provides details for the top 32 threats that it detects. Unlike other GuardDuty findings, Malware Protection findings are not updated when the same EC2 instance or container workload is scanned again.

A new Malware Protection finding is generated for each scan that detects malware. Malware Protection findings include information about the corresponding scan that produced the finding as well as the GuardDuty finding that initiated this scan. This makes it easier to correlate the suspicious behavior with the detected malware.

> **Note**
>
> When GuardDuty detects malicious activity on a container workload, Malware Protection doesn't generate an EC2 level finding.

The following findings are specific to GuardDuty Malware Protection.

Topics

Execution:EC2/MaliciousFile

A malicious file has been detected on an EC2 instance.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more malicious files on the listed EC2 instance within your AWS environment. This listed instance might be compromised. For more information, see **Threats detected** section in the findings' details.

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

Execution:ECS/MaliciousFile

A malicious file has been detected on an ECS cluster.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more malicious files on a container workload that belongs to an ECS cluster. For more information, see **Threats detected** section in the findings' details.

Remediation recommendations:

If this activity is unexpected, your container belonging to the ECS cluster may be compromised. For more information, see Remediating a potentially compromised ECS cluster.

Execution:Kubernetes/MaliciousFile

A malicious file has been detected on an Kubernetes cluster.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more malicious files on a container workload that belongs to a Kubernetes cluster. If this is an EKS managed cluster, the findings details will provide additional information about the impacted EKS resource. For more information, see **Threats detected** section in the findings' details.

Remediation recommendations:

If this activity is unexpected, your container workload may be compromised. For more information, see Remediating EKS Audit Log Monitoring findings.

Execution:Container/MaliciousFile

A malicious file has been detected on a standalone container.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more malicious files on a container workload and no cluster information has been identified. For more information, see **Threats detected** section in the findings' details.

Remediation recommendations:

If this activity is unexpected, your container workload may be compromised. For more information, see Remediating a potentially compromised standalone container.

Execution:EC2/SuspiciousFile

A suspicious file has been detected on an EC2 instance.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more suspicious files on an EC2 instance. For more information, see **Threats detected** section in the findings' details.

`SuspiciousFile` type detections indicate that potentially unwanted programs such as adware, spyware, or dual use tools are present on an impacted resource. These programs could have a negative impact on your resource, or be used by attackers for malicious purposes. For example, networking tools can be used legitimately or maliciously by adversaries as hack tools to try and compromise resources.

When a suspicious file has been detected, evaluate whether you expect to see the detected file in your AWS environment. If the file is unexpected, follow the remediation recommendations provided in the next section.

Remediation recommendations:

If this activity is unexpected, your instance may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

Execution:ECS/SuspiciousFile

A suspicious file has been detected on an ECS cluster.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more suspicious files on a container that belongs to an ECS cluster. For more information, see **Threats detected** section in the findings' details.

`SuspiciousFile` type detections indicate that potentially unwanted programs such as adware, spyware, or dual use tools are present on an impacted resource. These programs could have a negative impact on your resource, or be used by attackers for malicious purposes. For example, networking tools can be used legitimately or maliciously by adversaries as hack tools to try and compromise resources.

When a suspicious file has been detected, evaluate whether you expect to see the detected file in your AWS environment. If the file is unexpected, follow the remediation recommendations provided in the next section.

Remediation recommendations:

If this activity is unexpected, your container belonging to the ECS cluster may be compromised. For more information, see Remediating a potentially compromised ECS cluster.

Execution:Kubernetes/SuspiciousFile

A suspicious file has been detected on a Kubernetes cluster.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more suspicious files on a container that belongs to a Kubernetes cluster. If this is an EKS managed cluster, the findings' details will provide additional information about the impacted EKS. For more information, see **Threats detected** section in the findings' details.

`SuspiciousFile` type detections indicate that potentially unwanted programs such as adware, spyware, or dual use tools are present on an impacted resource. These programs could have a negative impact on your resource, or be used by attackers for malicious purposes. For example, networking tools can be used legitimately or maliciously by adversaries as hack tools to try and compromise resources.

When a suspicious file has been detected, evaluate whether you expect to see the detected file in your AWS environment. If the file is unexpected, follow the remediation recommendations provided in the next section.

Remediation recommendations:

If this activity is unexpected, your container workload may be compromised. For more information, see Remediating EKS Audit Log Monitoring findings.

Execution:Container/SuspiciousFile

A suspicious file has been detected on a standalone container.

Default severity: Varies depending on the detected threat.

- **Feature:** EBS Malware Protection

This finding indicates that the GuardDuty Malware Protection scan has detected one or more suspicious files on a container with no cluster information. For more information, see **Threats detected** section in the findings' details.

`SuspiciousFile` type detections indicate that potentially unwanted programs such as adware, spyware, or dual use tools are present on an impacted resource. These programs could have a negative impact on your resource, or be used by attackers for malicious purposes. For example, networking tools can be used legitimately or maliciously by adversaries as hack tools to try and compromise resources.

When a suspicious file has been detected, evaluate whether you expect to see the detected file in your AWS environment. If the file is unexpected, follow the remediation recommendations provided in the next section.

Remediation recommendations:

If this activity is unexpected, your container workload may be compromised. For more information, see Remediating a potentially compromised standalone container.

GuardDuty RDS Protection finding types

GuardDuty RDS Protection detects anomalous login behavior on your database instance. The following findings are specific to the Supported Amazon Aurora databases and will have a **Resource Type** of `RDSDBInstance`. The severity and details of the findings will differ based on the finding type.

Topics

CredentialAccess:RDS/AnomalousBehavior.SuccessfulLogin

A user successfully logged into an RDS database in your account in an anomalous way.

Default severity: Variable

> **Note**
>
> Depending on the anomalous behavior associated with this finding, the default severity can Low, Medium, and High.
>
> - **Low –** If the user name associated with this finding logged in from an IP address that is associated with a private network.
> - **Medium –** If the user name associated with this finding logged in from a public IP address.
> - **High –** If there is a consistent pattern of failed login attempts from public IP addresses indicative of overly permissive access policies.

- **Feature:** RDS login activity monitoring

This finding informs you that an anomalous successful login was observed on an RDS database in your AWS environment. This may indicate that a previous unseen user logged into an RDS database for the first time. A common scenario is an internal user logging into a database that is accessed programmatically by applications and not by individual users.

This successful login was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all database login events in your Supported Amazon Aurora databases and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the RDS login activity such as the user that made the request, the location the request was made from, and the specific database connection details that were used. For information about the login events that are potentially unusual, see RDS login activity-based anomalies.

Remediation recommendations:

If this activity is unexpected for the associated database, it is recommended to change the password of the associated database user, and review available audit logs for activity performed by the anomalous user. Medium and high severity findings may indicate that there is an overly permissive access policy to the database, and user credentials may have been exposed or compromised. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with successful login events.

CredentialAccess:RDS/AnomalousBehavior.FailedLogin

One or more unusual failed login attempts were observed on an RDS database in your account.

Default severity: Low

- **Feature:** RDS login activity monitoring

This finding informs you that one or more anomalous failed logins were observed on an RDS database in your AWS environment. A failed login attempts from public IP addresses may indicate that the RDS database in your account has been subject to an attempted brute force attack by a potentially malicious actor.

These failed logins were identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all database login events in your Supported Amazon Aurora databases and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the RDS login activity such as the user that made the request, the location the request was made from, and the specific database connection details that were used. For information about the RDS login activity that are potentially unusual, see RDS login activity-based anomalies.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that the database is publicly exposed or there is an overly permissive access policy to the database. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with failed login events.

CredentialAccess:RDS/AnomalousBehavior.SuccessfulBruteForce

A user successfully logged into an RDS database in your account from a public IP address in an anomalous way after a consistent pattern of unusual failed login attempts.

Default severity: High

- **Feature:** RDS login activity monitoring

This finding informs you that an anomalous login indicative of a successful brute force was observed on an RDS database in your AWS environment. Prior to an anomalous successful login, a consistent pattern of unusual failed login attempts was observed. This indicates that the user and password associated with the RDS database in your account may have been compromised, and the RDS database may have been accessed by a potentially malicious actor.

This successful brute force login was identified as anomalous by the GuardDuty anomaly detection machine learning (ML) model. The ML model evaluates all database login events in your Supported Amazon Aurora databases and identifies anomalous events that are associated with techniques used by adversaries. The ML model tracks various factors of the RDS login activity such as the user that made the request, the location the request was made from, and the specific database connection details that were used. For information about the RDS login activity that are potentially unusual, see RDS login activity-based anomalies.

Remediation recommendations:

This activity indicates that database credentials may have been exposed or compromised. It is recommended to change the password of the associated database user, and review available audit logs for activity performed by the potentially compromised user. A consistent pattern of unusual failed login attempts indicate an overly permissive access policy to the database or the database may have also been public exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with successful login events.

CredentialAccess:RDS/MaliciousIPCaller.SuccessfulLogin

A user successfully logged into an RDS database in your account from a known malicious IP address.

Default severity: High

- **Feature:** RDS login activity monitoring

This finding informs you that a successful RDS login activity occurred from an IP address that is associated with a known malicious activity in your AWS environment. This indicates that the user and password associated with the RDS database in your account may have been compromised, and the RDS database may have been accessed by a potentially malicious actor.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that the user credentials may have been exposed or compromised. It is recommended to change the password of the associated database user, and review the available audit logs for activity performed by the compromised user. This activity may also indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with successful login events.

CredentialAccess:RDS/MaliciousIPCaller.FailedLogin

An IP address that is associated with a known malicious activity unsuccessfully attempted to log in to an RDS database in your account.

Default severity: Medium

- **Feature:** RDS login activity monitoring

This finding informs you that an IP address associated with known malicious activity attempted to log in to an RDS database in your AWS environment, but failed to provide the correct user name or password. This indicates that a potentially malicious actor may be attempting to compromise the RDS database in your account.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with failed login events.

Discovery:RDS/MaliciousIPCaller

An IP address that is associated with a known malicious activity probed an RDS database in your account; no authentication attempt was made.

Default severity: Medium

- **Feature:** RDS login activity monitoring

This finding informs you that an IP address associated with known a malicious activity probed an RDS database in your AWS environment, though no login attempt was made. This may indicate that a potentially malicious actor is attempting to scan for a publicly accessible infrastructure.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with failed login events.

CredentialAccess:RDS/TorIPCaller.SuccessfulLogin

A user successfully logged into an RDS database in your account from a Tor exit node IP address.

Default severity: High

- **Feature:** RDS login activity monitoring

This finding informs you that a user successfully logged in to an RDS database in your AWS environment, from a Tor exit node IP address. Tor is a 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 the RDS resources in your account, with the intent of hiding the anonymous user's true identity.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that the user credentials may have been exposed or compromised. It is recommended to change the password of the associated database user, and review the available audit logs for activity performed by the compromised user. This activity may also indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with successful login events.

CredentialAccess:RDS/TorIPCaller.FailedLogin

A Tor IP address attempted to unsuccessfully log in to an RDS database in your account.

Default severity: Medium

- **Feature:** RDS login activity monitoring

This finding informs you that a Tor exit node IP address attempted to log in to an RDS database in your AWS environment, but failed to provide the correct user name or password. Tor is a 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 the RDS resources in your account, with the intent of hiding the anonymous user's true identity.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with failed login events.

Discovery:RDS/TorIPCaller

A Tor exit node IP address probed an RDS database in your account, no authentication attempt was made.

Default severity: Medium

- **Feature:** RDS login activity monitoring

This finding informs you that a Tor exit node IP address probed an RDS database in your AWS environment, though no login attempt was made. This may indicate that a potentially malicious actor is attempting to scan for publicly accessible infrastructure. Tor is a software for enabling anonymous communication. It encrypts and randmonly 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 the RDS resources in your account, with the intent of hiding the potentially malicious actor's true identity.

Remediation recommendations:

If this activity is unexpected for the associated database, it may indicate that there is an overly permissive access policy to the database or the database is publicly exposed. It is recommended to place the database in a private VPC, and limit the security group rules to allow traffic only from the necessary sources. For more information, see Remediating potentially compromised database with failed login events.

Runtime Monitoring finding types

Amazon GuardDuty generates the following Runtime Monitoring findings to indicate potential threats based on the operating system-level behavior from Amazon EC2 hosts and containers in your Amazon EKS clusters, Fargate and Amazon ECS workloads, and Amazon EC2 instances.

> **Note**
>
> Runtime Monitoring finding types are based on the runtime logs collected from hosts. The logs contain fields such as file paths that may be controlled by a malicious actor. These fields are also included in GuardDuty findings to provide runtime context. When processing Runtime Monitoring findings outside of GuardDuty console, you must sanitize

finding fields. For example, you can HTML encode finding fields when displaying them on a webpage.

Topics

- DefenseEvasion:Runtime/FilelessExecution
- Impact:Runtime/CryptoMinerExecuted
- Execution:Runtime/NewLibraryLoaded
- PrivilegeEscalation:Runtime/ContainerMountsHostDirectory
- PrivilegeEscalation:Runtime/UserfaultfdUsage
- Execution:Runtime/SuspiciousTool
- Execution:Runtime/SuspiciousCommand
- DefenseEvasion:Runtime/SuspiciousCommand
- DefenseEvasion:Runtime/PtraceAntiDebugging
- Execution:Runtime/MaliciousFileExecuted

CryptoCurrency:Runtime/BitcoinTool.B

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or a container in your AWS environment is querying an IP Address that is associated with a cryptocurrency-related activity. Threat actors may seek to take control over compute resources to maliciously repurpose them for unauthorized cryptocurrency mining.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If you use this EC2 instance or a container to mine or manage cryptocurrency, or either of these is otherwise involved in blockchain activity, the CryptoCurrency:Runtime/BitcoinTool.B 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 filter criterion should use the **Finding type** attribute with a value of `CryptoCurrency:Runtime/BitcoinTool.B`. The second filter criterion should be the **Instance ID** of the instance or the **Container Image ID** of the container involved in cryptocurrency or blockchain-related activity. For more information, see Suppression rules.

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Backdoor:Runtime/C&CActivity.B

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or a container within your AWS environment is querying an IP associated with a known command and control (C&C) server. The listed instance or container might be potentially compromised. Command and control servers are computers that issue commands to members of a botnet.

A botnet is a collection of internet-connected devices that 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 the fields of the associated finding will include the following values:

- `service.additionalInfo.threatListName = Amazon`
- `service.additionalInfo.threatName = Log4j Related`

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

UnauthorizedAccess:Runtime/TorRelay

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that an EC2 instance or a container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

UnauthorizedAccess:Runtime/TorClient

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that an EC2 instance or a container 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 or the container has been potentially 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/BlackholeTraffic

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

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you the listed EC2 instance or a container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/DropPoint

An Amazon EC2 instance or a container 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

- **Feature:** Runtime Monitoring

This finding informs you that an EC2 instance or a container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

CryptoCurrency:Runtime/BitcoinTool.B!DNS

An Amazon EC2 instance or a container is querying a domain name that is associated with a cryptocurrency activity.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or a container in your AWS environment is querying a domain name that is associated with Bitcoin or other cryptocurrency-related activity. Threat actors may seek to take control over the compute resources in order to maliciously repurpose them for unauthorized cryptocurrency mining.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If you use this EC2 instance or container to mine or manage cryptocurrency, or either of these is otherwise involved in blockchain activity, the CryptoCurrency:Runtime/BitcoinTool.B!DNS finding could be an 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 criterion. The first criteria should use the **Finding type** attribute with a value of `CryptoCurrency:Runtime/BitcoinTool.B!DNS`. The second filter criteria should be the **Instance ID** of the instance or the **Container Image ID** of the container involved in cryptocurrency or blockchain activity. For more information, see Suppression Rules.

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Backdoor:Runtime/C&CActivity.B!DNS

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container within your AWS environment is querying a domain name associated with a known command and control (C&C) server. The listed EC2 instance or the container 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`.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/BlackholeTraffic!DNS

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

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you the listed EC2 instance or the container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/DropPoint!DNS

An Amazon EC2 instance or a container 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

- **Feature:** Runtime Monitoring

This finding informs you that an EC2 instance or a container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/DGADomainRequest.C!DNS

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container in your AWS environment is trying to query domain generation algorithm (DGA) domains. Your resource might have been 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 threat intelligence feeds.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/DriveBySourceTraffic!DNS

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container 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 initiate an automatic installation of a virus, spyware, or malware.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Trojan:Runtime/PhishingDomainRequest!DNS

An Amazon EC2 instance or a container is querying domains involved in phishing attacks.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that there is an EC2 instance or a container 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 or the container might 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 or the container might be compromised.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Impact:Runtime/AbusedDomainRequest.Reputation

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

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container 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 or the container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Impact:Runtime/BitcoinDomainRequest.Reputation

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container within your AWS environment is querying a low reputation domain name associated with Bitcoin or other cryptocurrency-related activity. Threat actors may seek to take control over compute resources to maliciously repurpose them for unauthorized cryptocurrency mining.

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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If you use this EC2 instance or the container to mine or manage cryptocurrency, or if these resources are 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 filter criterion should use the **Finding type** attribute with a value of `Impact:Runtime/BitcoinDomainRequest.Reputation`. The second filter criterion should be the **Instance ID** of the instance or the **Container Image ID** of the container is involved in cryptocurrency or blockchain–related activity. For more information, see Suppression rules.

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Impact:Runtime/MaliciousDomainRequest.Reputation

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

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Impact:Runtime/SuspiciousDomainRequest.Reputation

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

Default severity: Low

- **Feature:** Runtime Monitoring

This finding informs you that the listed EC2 instance or the container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

UnauthorizedAccess:Runtime/MetadataDNSRebind

An Amazon EC2 instance or a container is performing DNS lookups that resolve to the instance metadata service.

Default severity: High

- **Feature:** Runtime Monitoring

Note

Presently, this finding type is only supported for AMD64 architecture.

This finding informs you that an EC2 instance or a container 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.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

In response to this finding, you should evaluate if there is a vulnerable application running on the EC2 instance or on the container, or if someone used a browser to access the domain identified in the finding. If the root cause is a vulnerable application, fix the vulnerability. If someone browsed the identified domain, block the domain or prevent users from accessing it. If you determine this finding was related to either case above, 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 filter criterion should use the **Finding type** attribute with a value of `UnauthorizedAccess:Runtime/MetaDataDNSRebind`. The second filter criterion should be **DNS request domain** or the **Container Image ID** of the container. The **DNS request domain** value should match the domain you have mapped to the metadata IP address (`169.254.169.254`). For information about creating suppression rules, see Suppression rules.

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Execution:Runtime/NewBinaryExecuted

A newly created or recently modified binary file in a container has been executed.

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you that a newly created or a recently modified binary file in a container was executed. It is the best practice to keep containers immutable at runtime, and binary files, scripts, or libraries should not be created or modified during the lifetime of the container. This behavior indicates that a malicious actor that has gained access to the container, has downloaded, and executed malware or other software as part of the potential compromise. Although this type of activity could be an indication of a compromise, it is also a common usage pattern. Therefore, GuardDuty uses mechanisms to identify suspicious instances of this activity and generates this finding type only for suspicious instances.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

PrivilegeEscalation:Runtime/DockerSocketAccessed

A process inside a container is communicating with Docker daemon using Docker socket.

Default severity: Medium

- **Feature:** Runtime Monitoring

The Docker socket is a Unix Domain Socket that Docker daemon (`dockerd`) uses to communicate with its clients. A client can perform various actions, such as creating containers by communicating with Docker daemon through the Docker socket. It is suspicious for a container process to access the Docker socket. A container process can escape the container and get a host-level access by communicating with the Docket socket and creating a privileged container.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

PrivilegeEscalation:Runtime/RuncContainerEscape

An attempt to gain host access of a container was detected.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that the host runC binary file has been potentially overwritten. runC is the low-level container runtime that high-level container runtimes, such as Docker and Containerd, use to spawn and run containers. runC is always executed with root privileges because it needs to perform a low-level task of creating a container. A well-known vulnerability[1] in the past allowed malicious containers to override the host's runC binary file and gained root-level access to the host when the modified runC binary was executed.

This finding may also indicate that a malicious actor has potentially executed a command in one of the following two types of containers:

- A new container with an attacker-controlled image.
- An existing container that was previously accessible to the attacker with write permissions.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

- 1. CVE-2019-5736 Detail

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

PrivilegeEscalation:Runtime/CGroupsReleaseAgentModified

A container escape through runC was detected in an Amazon EKS cluster.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that an attempt to modify a control group (cgroup) release agent file has been detected. Linux uses control groups (cgroups) to limit, account for, and isolate the resource usage of a collection of processes. Each cgroup has a release agent file (`release_agent`), a script that Linux executes when any process inside the cgroup terminates. The release agent file is always executed at the host level. A threat actor inside a container can escape to the host by writing arbitrary commands to the release agent file that belongs to a cgroup. When a process inside that cgroup terminates, the commands written by the actor get executed.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

DefenseEvasion:Runtime/ProcessInjection.Proc

A process injection using proc filesystem was detected in a container or an Amazon EC2 instance.

Default severity: High

- **Feature:** Runtime Monitoring

Process injection is a technique that threat actors use to inject code into processes to evade defenses and potentially elevate privileges. The proc filesystem (procfs) is a special filesystem in Linux that presents the virtual memory of process as a file. The path of that file is `/proc/PID/mem`, where `PID` is the unique ID of the process. A threat actor can write to this file to inject code into the process. This finding identifies potential attempts to write to this file.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource type might have been compromised. For more information, see [Remediating Runtime Monitoring findings](#).

DefenseEvasion:Runtime/ProcessInjection.Ptrace

A process injection using ptrace system call was detected in a container or an Amazon EC2 instance.

Default severity: Medium

- **Feature:** Runtime Monitoring

Process injection is a technique that threat actors use to inject code into processes to evade defenses and potentially elevate privileges. A process can use ptrace system call to inject code into another process. This finding identifies a potential attempt to inject code into a process using the ptrace system call.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource type might have been compromised. For more information, see [Remediating Runtime Monitoring findings](#).

DefenseEvasion:Runtime/ProcessInjection.VirtualMemoryWrite

A process injection through a direct write to virtual memory was detected in a container or an Amazon EC2 instance.

Default severity: High

- **Feature:** Runtime Monitoring

Process injection is a technique that threat actors use to inject code into processes to evade defenses and potentially elevate privileges. A process can use a system call such as `process_vm_writev` to directly inject code into another process's virtual memory. This finding identifies a potential attempt to inject code into a process using a system call for writing to the virtual memory of the process.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource type might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Execution:Runtime/ReverseShell

A process in a container or an Amazon EC2 instance has created a reverse shell.

Default severity: High

- **Feature:** Runtime Monitoring

A reverse shell is a shell session created on a connection that is initiated from the target host to the actor's host. This is opposite to a normal shell that is initiated from the actor's host to the target's host. Threat actors create a reverse shell to execute commands on the target after gaining initial access to the target. This finding identifies a potential attempt to create a reverse shell.

Remediation recommendations:

If this activity is unexpected, your resource type might have been compromised.

DefenseEvasion:Runtime/FilelessExecution

A process in a container or an Amazon EC2 instance is executing code from memory.

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you when a process is executed using an in-memory executable file on disk. This is a common defense evasion technique that avoids writing the malicious executable to the disk to evade file system scanning-based detection. Although this technique is used by malware, it also has some legitimate use cases. One of the examples is a just-in-time (JIT) compiler that writes compiled code to memory and executes it from memory.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Impact:Runtime/CryptoMinerExecuted

A container or an Amazon EC2 instance is executing a binary file that is associated with a cryptocurrency mining activity.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that a container or an EC2 instance in your AWS environment is executing a binary file that is associated with a cryptocurrency mining activity. Threat actors may seek to take control over compute resources to maliciously repurpose them for unauthorized cryptocurrency mining.

The runtime agent monitors events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the findings panel in the GuardDuty console.

Remediation recommendations:

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console and see Remediating Runtime Monitoring findings.

Execution:Runtime/NewLibraryLoaded

A newly created or recently modified library was loaded by a process inside a container.

Default severity: Medium

- **Feature:** Runtime Monitoring

This finding informs you that a library was created or modified inside a container during runtime and loaded by a process running inside the container. The best practice is to keep the containers immutable at the runtime, and not to create or modify the binary files, scripts, or libraries during the lifetime of the container. Loading of a newly created or modified library in a container may indicate suspicious activity. This behavior indicates that a malicious actor has potentially gained access to the container, has downloaded, and executed malware or other software as a part of the potential compromise. Although this type of activity could be an indication of a compromise, it is also a common usage pattern. Therefore, GuardDuty uses mechanisms to identify suspicious instances of this activity and generates this finding type only for suspicious instances.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

PrivilegeEscalation:Runtime/ContainerMountsHostDirectory

A process inside a container mounted a host filesystem at runtime.

Default severity: Medium

- **Feature:** Runtime Monitoring

Multiple container escape techniques involve mounting a host filesystem inside a container at runtime. This finding informs you that a process inside a container potentially attempted to mount a host filesystem, which may indicate an attempt to escape to the host.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

PrivilegeEscalation:Runtime/UserfaultfdUsage

A process used `userfaultfd` system calls to handle page faults in user space.

Default severity: Medium

- **Feature:** Runtime Monitoring

Typically, page faults are handled by the kernel in kernel space. However, `userfaultfd` system call allows a process to handle page faults on a filesystem in user space. This is a useful feature that enables implementation of user-space filesystems. On the other hand, it can also be used by a potentially malicious process to interrupt kernel from user space. Interrupting kernel by using `userfaultfd` system call is a common exploitation technique to extend race windows during exploitation of kernel race conditions. Use of `userfaultfd` may indicate suspicious activity on the Amazon Elastic Compute Cloud (Amazon EC2) instance.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Execution:Runtime/SuspiciousTool

A container or an Amazon EC2 instance is running a binary file or script that is frequently used in offensive security scenarios such as pentesting engagement.

Default severity: Variable

The severity of this finding can be either high or low, depending on whether the detected suspicious tool is considered to be dual-use or is it exclusively for offensive use.

- **Feature:** Runtime Monitoring

This finding informs you that a suspicious tool has been executed on an EC2 instance or container within your AWS environment. This includes tools used in pentesting engagements, also known as backdoor tools, network scanners, and network sniffers. All these tools can be used in benign contexts but are also frequently used by threat actors with malicious intent. Observing offensive security tools could indicate that the associated EC2 instance or container has been compromised.

GuardDuty examines related runtime activity and context so that it generates this finding only when the associated activity and context are potentially suspicious.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Execution:Runtime/SuspiciousCommand

A suspicious command has been executed on an Amazon EC2 instance or a container that is indicative of a compromise.

Default severity: Variable

Depending on the impact of the observed malicious pattern, the severity of this finding type could be either low, medium, or high.

- **Feature:** Runtime Monitoring

This finding informs you that a suspicious command has been executed and it indicates that an Amazon EC2 instance or a container in your AWS environment has been compromised. This might mean that either a file was downloaded from a suspicious source and then executed, or a running process displays a known malicious pattern in its command line. This further indicates that malware is running on the system.

GuardDuty examines related runtime activity and context so that it generates this finding only when the associated activity and context are potentially suspicious.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

DefenseEvasion:Runtime/SuspiciousCommand

A command has been executed on the listed Amazon EC2 instance or a container, it attempts to modify or disable a Linux defense mechanism, such as firewall or essential system services.

Default severity: Variable

Depending on which defense mechanism has been modified or disabled, the severity of this finding type can be either high, medium, or low.

- **Feature:** Runtime Monitoring

This finding informs you that a command that attempts to hide an attack from the local system's security services, has been executed. This includes actions such as disabling the Unix firewall, modifying local IP tables, removing crontab entries, disabling a local service, or taking over the `LDPreload` function. Any modification is highly suspicious and a potential indicator of compromise. Therefore, these mechanisms detect or prevent further compromise of the system.

GuardDuty examines related runtime activity and context so that it generates this finding only when the associated activity and context are potentially suspicious.

The runtime agent monitors events from multiple resources. To identify the potentially compromised resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

DefenseEvasion:Runtime/PtraceAntiDebugging

A process in a container or an Amazon EC2 instance has executed an anti-debugging measure using the ptrace system call.

Default severity: Low

- **Feature:** Runtime Monitoring

This finding shows that a process running on an Amazon EC2 instance or a container within your AWS environment has used the ptrace system call with the `PTRACE_TRACEME` option. This activity would cause an attached debugger to detach from the running process. If no debugger is attached, it has no effect. However, the activity in itself raises suspicion. This might indicate that malware is running on the system. Malware frequently uses anti-debugging techniques to evade analysis, and these techniques can be detected at runtime.

GuardDuty examines related runtime activity and context so that it generates this finding only when the associated activity and context are potentially suspicious.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

Execution:Runtime/MaliciousFileExecuted

A known malicious executable file has been executed on an Amazon EC2 instance or a container.

Default severity: High

- **Feature:** Runtime Monitoring

This finding informs you that a known malicious executable has been executed on Amazon EC2 instance or a container within your AWS environment. This is a strong indicator that the instance or container has been potentially compromised and that malware has been executed.

Malware frequently uses anti-debugging techniques to evade analysis, and these techniques can be detected at runtime.

GuardDuty examines related runtime activity and context so that it generates this finding only when the associated activity and context are potentially suspicious.

The runtime agent monitors events from multiple resources. To identify the affected resource, view **Resource type** in the findings details in the GuardDuty console.

Remediation recommendations:

If this activity is unexpected, your resource might have been compromised. For more information, see Remediating Runtime Monitoring findings.

GuardDuty S3 finding types

The following findings are specific to Amazon S3 resources and will have a **Resource Type** of `S3Bucket` if the data source is **CloudTrail data events for S3**, 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 Foundational data sources.

⚠ **Important**

Findings with a data source of **CloudTrail data events for S3** 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 about how to enable or disable S3 protection, see Amazon S3 Protection in Amazon GuardDuty

For all `S3Bucket` 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 potentially compromised S3 bucket.

Topics

- UnauthorizedAccess:S3/TorIPCaller

Discovery:S3/AnomalousBehavior

An API commonly used to discover S3 objects was invoked in an anomalous way.

Default severity: Low

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

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

This API was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all the API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. It tracks various factors of the API requests, such as the user who made the request, the location from which the request was made, the specific API that was requested, the bucket that was requested, and the number of API calls made. For more information on which factors of the API request are unusual for the user identity that invoked the request, see Finding details.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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:** CloudTrail data events for S3

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 about your AWS environment. Examples include `GetObjectAcl` and `ListObjects`.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see [Remediating a potentially compromised S3 bucket].

Discovery:S3/MaliciousIPCaller.Custom

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

Default severity: High

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see [Remediating a potentially compromised S3 bucket].

Discovery:S3/TorIPCaller

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

Default severity: Medium

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

Exfiltration:S3/AnomalousBehavior

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

Default severity: High

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

This finding informs you that an IAM entity is making API calls that involve an S3 bucket and this activity differs from that entity's established baseline. The API call used in this activity is associated with the exfiltration stage of an attack, wherein an attacker attempts to collect data. This activity is suspicious because the IAM entity invoked the API in an unusual way. For example, an IAM entity with no previous history invokes an S3 API, or an IAM entity invokes an S3 API from an unusual location.

This API was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all the API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. It tracks various factors of the API requests, such as the user who made the request, the location from which the request was made, the specific API that was requested, the bucket that was requested, and the number of API calls made. For more information on which factors of the API request are unusual for the user identity that invoked the request, see Finding details.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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:** CloudTrail data events for S3

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

Impact:S3/AnomalousBehavior.Delete

An IAM entity invoked an S3 API that attempts to delete data in a suspicious way.

Default severity: High

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

This finding informs you that an IAM entity in your AWS environment is making API calls that involve an S3 bucket, and this behavior differs from that entity's established baseline. The API call used in this activity is associated with an attack that attempts to delete data. This activity is suspicious because the IAM entity invoked the API in an unusual way. For example, an IAM entity with no previous history invokes an S3 API, or an IAM entity invokes an S3 API from an unusual location.

This API was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all the API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. It tracks various factors of the API requests, such as the user who made the request, the location from which the request was made, the specific API that was requested, the bucket that was requested, and the number of API calls made. For more information on which factors of the API request are unusual for the user identity that invoked the request, see Finding details.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

We recommend an audit of your S3 bucket's contents to determine if you the previous object version can or should be restored.

Impact:S3/AnomalousBehavior.Permission

An API commonly used to set the access control list (ACL) permissions was invoked in an anomalous way.

Default severity: High

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

This finding informs you that an IAM entity in your AWS environment has changed a bucket policy or ACL on the listed S3 buckets. This change may publicly expose your S3 buckets to all the authenticated AWS users.

This API was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all the API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. It tracks various factors of the API requests, such as the user who made the request, the location from which the request was made, the specific API that was requested, the bucket that was requested, and the number of API calls made. For more information on which factors of the API request are unusual for the user identity that invoked the request, see Finding details.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

We recommend an audit of your S3 bucket's contents to ensure that no objects were unexpectedly allowed to be accessed publicly.

Impact:S3/AnomalousBehavior.Write

An IAM entity invoked an S3 API that attempts to write data in a suspicious way.

Default severity: Medium

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

This finding informs you that an IAM entity in your AWS environment is making API calls that involve an S3 bucket, and this behavior differs from that entity's established baseline. The API call used in this activity is associated with an attack that attempts to write data. This activity is suspicious because the IAM entity invoked the API in an unusual way. For example, an IAM entity with no previous history invokes an S3 API, or an IAM entity invokes an S3 API from an unusual location.

This API was identified as anomalous by GuardDuty's anomaly detection machine learning (ML) model. The ML model evaluates all the API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. It tracks various factors of the API requests, such as the user who made the request, the location from which the request was made, the specific API that was requested, the bucket that was requested, and the number of API calls made. For more information on which factors of the API request are unusual for the user identity that invoked the request, see Finding details.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

We recommend an audit of your S3 bucket's contents to ensure that this API call didn't write malicious or unauthorized data.

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:** CloudTrail data events for S3

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` and `PutObjectAcl`.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

PenTest:S3/KaliLinux

An S3 API was invoked from a Kali Linux machine.

Default severity: Medium

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

PenTest:S3/ParrotLinux

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

Default severity: Medium

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

PenTest:S3/PentooLinux

An S3 API was invoked from a Pentoo Linux machine.

Default severity: Medium

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 may not reflect the current state of the bucket. This finding will not reflect any S3 Block Public Access settings that may have been enabled for your S3 bucket. In such cases, the `effectivePermission` value in the finding will be marked as `UNKNOWN`.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 may not reflect the current state of the bucket. This finding will not reflect any S3 Block Public Access settings that may have been enabled for your S3 bucket. In such cases, the `effectivePermission` value in the finding will be marked as `UNKNOWN`.

Remediation recommendations:

If this activity is unexpected for the associated principal, it may indicate that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

UnauthorizedAccess:S3/MaliciousIPCaller.Custom

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

Default severity: High

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

UnauthorizedAccess:S3/TorIPCaller

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

Default severity: High

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

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

Retired finding types

A finding is a notification that contains details about a potential security issue that GuardDuty discovers. For information about important changes to the GuardDuty finding types, including newly added or retired finding types, see Document history for Amazon GuardDuty.

The following finding types are retired and no longer generated by GuardDuty.

> ⚠ **Important**
>
> You can't reactivate retired GuardDuty finding types.

Topics

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:** CloudTrail data events for S3

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 that the credentials have been exposed or your S3 permissions are not restrictive enough. For more information, see Remediating a potentially compromised S3 bucket.

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 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. For more information, see Remediating potentially compromised AWS credentials.

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 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. For more information, see Remediating potentially compromised AWS credentials.

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 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. For more information, see Remediating potentially compromised AWS credentials.

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's sign-in credential was stolen and deleted 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. For more information, see Remediating potentially compromised AWS credentials.

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 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. For more information, see Remediating potentially compromised AWS credentials.

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 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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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. For more information, see Remediating potentially compromised AWS credentials.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

Backdoor:EC2/XORDDOS

An EC2 instance is attempting to communicate with an IP address that is associated with XOR DDoS 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 XOR DDoS 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 user 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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

Behavior:IAMUser/InstanceLaunchUnusual

A user launched an EC2 instance of an unusual type.

Default severity: High

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

Remediation recommendations:

If this activity is unexpected, your credentials may be compromised. For more information, see Remediating potentially compromised AWS credentials.

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 may be compromised. For more information, see Remediating a potentially compromised Amazon EC2 instance.

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. For more information, see Remediating potentially compromised AWS credentials.

Findings by resource type

The following pages are categorized by resource type associated to a GuardDuty finding:

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

Findings table

The following table shows all of the active finding types sorted by the foundational data source or feature, as applicable. Some of the following finding types may have a variable severity, indicated by an asterisk (*). For information about the variable severity of a finding type, view the detailed description of that finding type.

Finding type	Resource type	Foundational data source/Feature	Finding severity
Discovery:S3/AnomalousBehavior	Amazon S3	CloudTrail data events for S3	Low
Discovery:S3/MaliciousIPCaller	Amazon S3	CloudTrail data events for S3	High
Discovery:S3/MaliciousIPCaller.Custom	Amazon S3	CloudTrail data events for S3	High
Discovery:S3/TorIPCaller	Amazon S3	CloudTrail data events for S3	Medium
Exfiltration:S3/AnomalousBehavior	Amazon S3	CloudTrail data events for S3	High
Exfiltration:S3/MaliciousIPCaller	Amazon S3	CloudTrail data events for S3	High
Impact:S3/AnomalousBehavior.Delete	Amazon S3	CloudTrail data events for S3	High
Impact:S3/AnomalousBehavior.Permission	Amazon S3	CloudTrail data events for S3	High
Impact:S3/Anomalou	Amazon S3	CloudTrail data events for S3	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
sBehavior.Write			
Impact:S3/MaliciousIPCaller	Amazon S3	CloudTrail data events for S3	High
PenTest:S3/KaliLinux	Amazon S3	CloudTrail data events for S3	Medium
PenTest:S3/ParrotLinux	Amazon S3	CloudTrail data events for S3	Medium
PenTest:S3/PentooLinux	Amazon S3	CloudTrail data events for S3	Medium
UnauthorizedAccess:S3/TorIPCaller	Amazon S3	CloudTrail data events for S3	High
UnauthorizedAccess:S3/MaliciousIPCaller.Custom	Amazon S3	CloudTrail data events for S3	High
CredentialAccess:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Medium
DefenseEvasion:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
Discovery:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Low
Exfiltration:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	High
Impact:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	High
InitialAccess:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Medium
PenTest:IAMUser/KaliLinux	IAM	CloudTrail management event	Medium
PenTest:IAMUser/ParrotLinux	IAM	CloudTrail management event	Medium
PenTest:IAMUser/PentooLinux	IAM	CloudTrail management event	Medium
Persistence:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
Stealth:IAMUser/PasswordPolicyChange	IAM	CloudTrail management event	Low*
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS	IAM	CloudTrail management event	High*
Policy:S3/AccountBlockPublicAccessDisabled	Amazon S3	CloudTrail management event	Low
Policy:S3/BucketAnonymousAccessGranted	Amazon S3	CloudTrail management event	High
Policy:S3/BucketBlockPublicAccessDisabled	Amazon S3	CloudTrail management event	Low
Policy:S3/BucketPublicAccessGranted	Amazon S3	CloudTrail management event	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Privilege Escalation:IAMUser/AnomalousBehavior	IAM	CloudTrail management event	Medium
Recon:IAMUser/MaliciousIPCaller	IAM	CloudTrail management event	Medium
Recon:IAMUser/MaliciousIPCaller.Custom	IAM	CloudTrail management event	Medium
Recon:IAMUser/TorIPCaller	IAM	CloudTrail management event	Medium
Stealth:IAMUser/CloudTrailLoggingDisabled	IAM	CloudTrail management event	Low
Stealth:S3/ServerAccessLoggingDisabled	Amazon S3	CloudTrail management event	Low
UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B	IAM	CloudTrail management event	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
UnauthorizedAccess:IAMUser/MaliciousIPCaller	IAM	CloudTrail management event	Medium
UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom	IAM	CloudTrail management event	Medium
UnauthorizedAccess:IAMUser/TorIPCaller	IAM	CloudTrail management event	Medium
Policy:IAMUser/RootCredentialUsage	IAM	CloudTrail management events or CloudTrail data events for S3	Low
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS	IAM	CloudTrail management events or CloudTrail data events for S3	High
Backdoor:EC2/C&CActivity.B!DNS	Amazon EC2	DNS logs	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
CryptoCurrency:EC2/BitcoinTool.B!DNS	Amazon EC2	DNS logs	High
Impact:EC2/AbusedDomainRequest.Reputation	Amazon EC2	DNS logs	Medium
Impact:EC2/BitcoinDomainRequest.Reputation	Amazon EC2	DNS logs	High
Impact:EC2/MaliciousDomainRequest.Reputation	Amazon EC2	DNS logs	High
Impact:EC2/SuspiciousDomainRequest.Reputation	Amazon EC2	DNS logs	Low
Trojan:EC2/BlackholeTraffic!DNS	Amazon EC2	DNS logs	Medium
Trojan:EC2/DGADomainRequest.B	Amazon EC2	DNS logs	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Trojan:EC2/DGADomainRequest.C!DNS	Amazon EC2	DNS logs	High
Trojan:EC2/DNSDataExfiltration	Amazon EC2	DNS logs	High
Trojan:EC2/DriveBySourceTraffic!DNS	Amazon EC2	DNS logs	High
Trojan:EC2/DropPoint!DNS	Amazon EC2	DNS logs	Medium
Trojan:EC2/PhishingDomainRequest!DNS	Amazon EC2	DNS logs	High
UnauthorizedAccess:EC2/MetadataDNSRebind	Amazon EC2	DNS logs	High
Execution:Container/MaliciousFile	Container	EBS Malware Protection	Varies depending on the detected threat
Execution:Container/SuspiciousFile	Container	EBS Malware Protection	Varies depending on the detected threat

Finding type	Resource type	Foundational data source/Feature	Finding severity
Execution:EC2/MaliciousFile	EC2	EBS Malware Protection	Varies depending on the detected threat
Execution:EC2/SuspiciousFile	EC2	EBS Malware Protection	Varies depending on the detected threat
Execution:ECS/MaliciousFile	ECS	EBS Malware Protection	Varies depending on the detected threat
Execution:ECS/SuspiciousFile	ECS	EBS Malware Protection	Varies depending on the detected threat
Execution:Kubernetes/MaliciousFile	Kubernetes	EBS Malware Protection	Varies depending on the detected threat
Execution:Kubernetes/SuspiciousFile	Kubernetes	EBS Malware Protection	Varies depending on the detected threat
CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed	Kubernetes	Kubernetes audit logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
CredentialAccess:Kubernetes/MaliciousIPCaller	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/MaliciousIPCaller.Custom	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/SuccessfulAnonymousAccess	Kubernetes	Kubernetes audit logs	High
CredentialAccess:Kubernetes/TorIPCaller	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/MaliciousIPCaller	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/MaliciousIPCaller.Custom	Kubernetes	Kubernetes audit logs	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
DefenseEvasion:Kubernetes/SuccessfulAnonymousAccess	Kubernetes	Kubernetes audit logs	High
DefenseEvasion:Kubernetes/TorIPCaller	Kubernetes	Kubernetes audit logs	High
Discovery:Kubernetes/AnomalousBehavior.PermissionChecked	Kubernetes	Kubernetes audit logs	Low
Discovery:Kubernetes/MaliciousIPCaller	Kubernetes	Kubernetes audit logs	Medium
Discovery:Kubernetes/MaliciousIPCaller.Custom	Kubernetes	Kubernetes audit logs	Medium
Discovery:Kubernetes/SuccessfulAnonymousAccess	Kubernetes	Kubernetes audit logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
Discovery:Kubernetes/TorIPCaller	Kubernetes	Kubernetes audit logs	Medium
Execution:Kubernetes/ExecInKubeSystemPod	Kubernetes	Kubernetes audit logs	Medium
Execution:Kubernetes/AnomalousBehavior.ExecInPod	Kubernetes	Kubernetes audit logs	Medium
Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed	Kubernetes	Kubernetes audit logs	Low
Impact:Kubernetes/MaliciousIPCaller	Kubernetes	Kubernetes audit logs	High
Impact:Kubernetes/MaliciousIPCaller.Custom	Kubernetes	Kubernetes audit logs	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Impact:Kubernetes/SuccessfulAnonymousAccess	Kubernetes	Kubernetes audit logs	High
Impact:Kubernetes/TorIPCaller	Kubernetes	Kubernetes audit logs	High
Persistence:Kubernetes/ContainerWithSensitiveMount	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/MaliciousIPCaller	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/MaliciousIPCaller.Custom	Kubernetes	Kubernetes audit logs	Medium
Persistence:Kubernetes/SuccessfulAnonymousAccess	Kubernetes	Kubernetes audit logs	High
Persistence:Kubernetes/TorIPCaller	Kubernetes	Kubernetes audit logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
Policy:Kubernetes/AdminAccessToDefaultServiceAccount	Kubernetes	Kubernetes audit logs	High
Policy:Kubernetes/AnonymousAccessGranted	Kubernetes	Kubernetes audit logs	High
Policy:Kubernetes/KubeflowDashboardExposed	Kubernetes	Kubernetes audit logs	Medium
Policy:Kubernetes/ExposedDashboard	Kubernetes	Kubernetes audit logs	Medium
PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated	Kubernetes	Kubernetes audit logs	Medium*

Finding type	Resource type	Foundational data source/Feature	Finding severity
PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated	Kubernetes	Kubernetes audit logs	Low
Persistence:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount	Kubernetes	Kubernetes audit logs	High
PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer	Kubernetes	Kubernetes audit logs	High
PrivilegeEscalation:Kubernetes/PrivilegedContainer	Kubernetes	Kubernetes audit logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
Backdoor:Lambda/C&CActivity.B	Lambda	Lambda Network Activity Monitoring	High
CryptoCurrency:Lambda/BitcoinTool.B	Lambda	Lambda Network Activity Monitoring	High
Trojan:Lambda/BlackholeTraffic	Lambda	Lambda Network Activity Monitoring	Medium
Trojan:Lambda/DropPoint	Lambda	Lambda Network Activity Monitoring	Medium
UnauthorizedAccess:Lambda/MaliciousIPCaller.Custom	Lambda	Lambda Network Activity Monitoring	Medium
UnauthorizedAccess:Lambda/TorClient	Lambda	Lambda Network Activity Monitoring	High
UnauthorizedAccess:Lambda/TorRelay	Lambda	Lambda Network Activity Monitoring	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
CredentialAccess:RDS/AnomalousBehavior.FailedLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Low
CredentialAccess:RDS/AnomalousBehavior.SuccessfulBruteForce	Supported Amazon Aurora databases	RDS Login Activity Monitoring	High
CredentialAccess:RDS/AnomalousBehavior.SuccessfulLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Variable*
CredentialAccess:RDS/MaliciousIPCaller.FailedLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Medium
CredentialAccess:RDS/MaliciousIPCaller.SuccessfulLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	High
CredentialAccess:RDS/TorIPCaller.FailedLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
CredentialAccess:RDS/TorIPCaller.SuccessfulLogin	Supported Amazon Aurora databases	RDS Login Activity Monitoring	High
Discovery:RDS/MaliciousIPCaller	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Medium
Discovery:RDS/TorIPCaller	Supported Amazon Aurora databases	RDS Login Activity Monitoring	Medium
Backdoor:Runtime/C&CActivity.B	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Backdoor:Runtime/C&CActivity.B!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
CryptoCurrency:Runtime/BitcoinTool.B	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
CryptoCurrency:Runtime/BitcoinTool.B!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
DefenseEvasion:Runtime/FilelessExecution	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
DefenseEvasion:Runtime/ProcessInjection.Proc	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
DefenseEvasion:Runtime/ProcessInjection.Ptrace	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
DefenseEvasion:Runtime/ProcessInjection.VirtualMemoryWrite	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
DefenseEvasion:Runtime/PtraceAntiDebugging	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Low
DefenseEvasion:Runtime/SuspiciousCommand	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Execution:Runtime/MaliciousFileExecuted	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Execution:Runtime/NewBinaryExecuted	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Execution:Runtime/NewLibraryLoaded	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Execution:Runtime/SuspiciousCommand	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Variable
Execution:Runtime/SuspiciousTool	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Variable
Execution:Runtime/ReverseShell	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Impact:Runtime/AbusedDomainRequest.Reputation	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Impact:Runtime/BitcoinDomainRequest.Reputation	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Impact:Runtime/CryptoMinerExecuted	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Impact:Runtime/MaliciousDomainRequest.Reputation	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Impact:Runtime/SuspiciousDomainRequest.Reputation	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Low
PrivilegeEscalation:Runtime/CGroupsReleaseAgentModified	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
PrivilegeEscalation:Runtime/ContainerMountsHostDirectory	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
PrivilegeEscalation:Runtime/DockerSocketAccessed	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
PrivilegeEscalation:Runtime/RuncContainerEscape	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
PrivilegeEscalation:Runtime/UserfaultfdUsage	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Trojan:Runtime/BlackholeTraffic	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Trojan:Runtime/BlackholeTraffic!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Trojan:Runtime/DropPoint	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Trojan:Runtime/DGADomainRequest.C!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Trojan:Runtime/DriveBySourceTraffic!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Trojan:Runtime/DropPoint!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	Medium
Trojan:Runtime/PhishingDomainRequest!DNS	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
UnauthorizedAccess:Runtime/MetadataDNSRebind	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
UnauthorizedAccess:Runtime/TorClient	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
UnauthorizedAccess:Runtime/TorRelay	Instance, EKS cluster, ECS cluster, or container	Runtime Monitoring	High
Backdoor:EC2/C&CActivity.B	EC2	VPC flow logs	High
Backdoor:EC2/DenialOfService.Dns	EC2	VPC flow logs	High
Backdoor:EC2/DenialOfService.Tcp	EC2	VPC flow logs	High

Finding type	Resource type	Foundational data source/Feature	Finding severity
Backdoor:EC2/DenialOfService.Udp	EC2	VPC flow logs	High
Backdoor:EC2/DenialOfService.UdpOnTcpPorts	EC2	VPC flow logs	High
Backdoor:EC2/DenialOfService.UnusualProtocol	EC2	VPC flow logs	High
Backdoor:EC2/Spambot	EC2	VPC flow logs	Medium
Behavior:EC2/NetworkPortUnusual	EC2	VPC flow logs	Medium
Behavior:EC2/TrafficVolumeUnusual	EC2	VPC flow logs	Medium
CryptoCurrency:EC2/BitcoinTool.B	EC2	VPC flow logs	High
DefenseEvasion:EC2/UnusualDNSResolver	EC2	VPC flow logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
DefenseEvasion:EC2/UnusualDoHActivity	EC2	VPC flow logs	Medium
DefenseEvasion:EC2/UnusualDoTActivity	EC2	VPC flow logs	Medium
Impact:EC2/PortSweep	EC2	VPC flow logs	High
Impact:EC2/WinRMBruteForce	EC2	VPC flow logs	Low*
Recon:EC2/PortProbeEMRUnprotectedPort	EC2	VPC flow logs	High
Recon:EC2/PortProbeUnprotectedPort	EC2	VPC flow logs	Low*
Recon:EC2/Portscan	EC2	VPC flow logs	Medium
Trojan:EC2/BlackholeTraffic	EC2	VPC flow logs	Medium
Trojan:EC2/DropPoint	EC2	VPC flow logs	Medium

Finding type	Resource type	Foundational data source/Feature	Finding severity
UnauthorizedAccess:EC2/MaliciousIPCaller.Custom	EC2	VPC flow logs	Medium
UnauthorizedAccess:EC2/RDPBruteForce	EC2	VPC flow logs	Low*
UnauthorizedAccess:EC2/SSHBruteForce	EC2	VPC flow logs	Low*
UnauthorizedAccess:EC2/TorClient	EC2	VPC flow logs	High
UnauthorizedAccess:EC2/TorRelay	EC2	VPC flow logs	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:

Summary dashboard

Learn about the components of the summary dashboard available in the GuardDuty console.

Filtering findings

Learn how to filter GuardDuty findings based on criteria you specify.

Suppression rules

Learn how to automatically filter the findings GuardDuty alerts you to through suppression rules. Suppression rules automatically archive findings based on filters.

Working with trusted IP lists and threat lists

Customize the GuardDuty monitoring scope using IP Lists and Threat Lists based on publicly-routable IP addresses. Trusted IP lists prevent non-DNS findings from being generated from IP's you consider trusted, while Threat Intel Lists will cause GuardDuty to alert you of activity from user-defined IPs.

Exporting findings

Export the generated findings to an Amazon S3 bucket so that you can maintain records past the 90-day findings retention period in GuardDuty. Use this historical data to track potential suspicious activities in your account and evaluate whether the recommended remediation steps were successful.

Creating custom responses to GuardDuty findings with Amazon CloudWatch Events

Set up automatic notifications for GuardDuty findings through Amazon CloudWatch events. You can also automate other tasks through CloudWatch Events to help you respond to findings.

Understanding CloudWatch Logs and reasons for skipping resources during Malware Protection scan

Learn how you can audit the CloudWatch Logs for GuardDuty Malware Protection and what are the reasons because of which your impacted Amazon EC2 instance or Amazon EBS volumes may have been skipped during the scanning process.

Reporting false positives in GuardDuty Malware Protection

Learn about the false positive experience in GuardDuty Malware Protection and how you can report false positive threat detections.

Summary dashboard

The **Summary** dashboard provides an aggregated view of the GuardDuty findings generated in your AWS account in the current Region. Presently, the dashboard supports a volume of up to 5,000 findings. However, you can view the details of all the findings by using either the **Findings** page on the GuardDuty console, or GetFindings or ListFindings.

Note

The findings summary is only available through the GuardDuty console at https://console.aws.amazon.com/guardduty/.

The following sections will help you access the dashboard and understand its components.

Contents

Accessing the Summary dashboard

On the GuardDuty console, the **Summary** dashboard shows a consolidated view of up to last 5,000 GuardDuty findings generated in the current Region.

To access the Summary dashboard

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Summary**. When you open the console, GuardDuty shows the **Summary** dashboard.
3. By default, the summary gets displayed for the same day – **Today**. The GuardDuty console provides an option to view the summary for the **Last 2 days**, **Last 7 days**, and **Last 30 days**. To change the default time range, choose one of the options from the dropdown above the **Overview** pane.
4. **Filter the data**

    - The **Accounts with most findings**, **Resources with most findings**, and **Least occurring findings** widgets help you filter the data on the basis of the level of severity of the findings.
    - The **Resources with most findings** widget also helps you filter the data on the basis of your potentially impacted resource type.

      A member account can view the details of the potentially impacted resource that belongs to their own account. If you're a GuardDuty administrator account and want to view the details of the potentially impacted resource, open the GuardDuty console using the credentials of the associated member account.
5. **Protection plans coverage**

    The protection plans coverage provides the count of member accounts that have enabled GuardDuty in your organization. The statistics are visible only to the delegated GuardDuty administrator.

Understanding the Summary dashboard

The **Summary** dashboard shows the aggregated data in the following sections. Before you proceed to view and understand the summary, make sure to choose the desired AWS Region from the Region selector at the top of the console. Also, make sure to choose the desired time range from the dropdown menu provided above the **Overview** pane. If no findings were generated for the chosen parameters, no data will be available in any of the widgets.

Out of a volume of up to last 5,000 GuardDuty findings, the summary dashboard with **Accounts with most findings**, **Resources with most findings**, and **Least occurring findings** shows the data based off of the top 5 results. For a deeper analysis, see the **Findings** page in the GuardDuty console.

Overview

This section provides the following data:

- **Total findings**: Indicates the total number of findings generated in your account in the current Region.
- **High severity findings**: Indicates the number of GuardDuty findings that have a high severity level in the current Region.
- **Resources with findings**: Indicates the number of resources that are associated to a finding and have been potentially compromised.
- **Accounts with findings**: Indicates the number of accounts in which at least one finding was generated. If you're a standalone account, the value in this field is **1**.

For the time ranges **Last 7 days** and **Last 30 days**, the **Overview** pane may show the percentage difference in the findings generated week over week (WoW) or month over month (MoM), respectively. If no findings were generated in the week or the month before, then with no data to compare, the percentage difference may not be available.

If you're a GuardDuty administrator account, all of these fields provide the summarized data across all the member accounts in your organization.

Findings by severity

This section displays a bar chart with the total number of findings against the chosen time range. You can view the number of findings with low, medium, or high severity, generated on a specific date within the chosen time range.

Most common finding types

This section provides a pie chart illustration of the top five common finding types as observed from a volume of up to last 5,000 GuardDuty findings generated in the current Region. This pie chart displays the following data when hovered over each sector:

- **Findings count**: Indicates the number of times this finding has been generated in the chosen time range.

- **Severity**: Indicates the severity level of the finding – for example, Medium and High.
- **Percentage**: Indicates the share of this finding type in the pie chart.
- **Last generated**: Indicates how much time has passed since this finding type was last generated.

Accounts with most findings

This section provides the following data:

- **Account**: Indicates the AWS account ID where the finding was generated.
- **Finding count**: Indicates the number of times a finding was generated for this account ID.
- **Last generated**: Indicates how much time has passed since a finding type was last generated for this account ID.
- **High severity**: By default, the data is shown for the high severity finding types. Possible options for this field are **High severity**, **Medium severity**, and **All severity**.

Resources with findings

This section provides the following data:

- **Resource**: Indicates the potentially impacted resource type and if this resource belongs to your account, you can access the quick link to view the resource details. If you're a GuardDuty administrator account, you can view the details of the potentially impacted resource by accessing the GuardDuty console with the credentials of the member account to which this resource belongs.
- **Account**: Indicates the AWS account ID to which this resource belongs.
- **Finding count**: Indicates the number of times that this resource was associated to a finding.
- **Last generated**: Indicates how much time has passed since a finding type associated to this resource was last generated.
- **All resource types**: By default, the data is shown for all of the resource types. By using the dropdown, you can view the data for a specific resource type, such as **Instance**, **AccessKey**, **Lambda**, and others.
- **High severity**: By default, the data is shown for the high severity finding types. By using the dropdown, you can view the data for other severity levels. Possible options are **High severity**, **Medium severity**, and **All severity**.

Least occurring findings

This section provides the details of the finding types that are not generated often in your AWS environment. This insight can help you investigate and take action on an emergent threat pattern in your environment. The table shows the following data:

- **Finding type**: Indicates the finding type name.
- **Finding count**: Indicates the number of times that this finding type was generated in the chosen time range.
- **Last generated**: Indicates how much time has passed since this finding type was last generated.
- **High severity**: By default, the data is shown for the high severity finding types. Possible options for this field are **High severity**, **Medium severity**, and **All severity**.

Protection plans coverage

This section provides the number of active member accounts that belong to your organization and have enabled one or more features and additional features (as applicable) configuration in the current AWS Region.

Only a delegated GuardDuty administrator can view the statistics for the member accounts within their organization. If a feature is not configured, choose **Configure** under the **Actions** column.

When you create a new AWS organization, it might take up to 24 hours to generate the statistics for the entire organization.

Providing feedback on the Summary dashboard

GuardDuty encourages you to provide feedback on the Summary dashboard's usability, features, and performance. This will help us improve the dashboard.

To provide feedback on the Summary dashboard

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Summary**. When you open the GuardDuty console, it shows the **Summary** dashboard.
3. Choose **Feedback** at the top-right corner of the dashboard. This will open up a form. After you provide the feedback, choose **Submit**.

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.

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
Finding ID	id
Region	region
Severity	severity If you use `severity` with API, AWS CLI, or AWS CloudFormation, it will have a numeric value. For more information, see findingCriteria.
Finding type	type
Updated at	updatedAt
Access Key ID	resource.accessKeyDetails.accessKeyId
Principal ID	resource.accessKeyDetails.principalId

Console field name	JSON field name
Username	resource.accessKeyDetails.userName
User type	resource.accessKeyDetails.userType
IAM instance profile ID	resource.instanceDetails.iamInstanceProfile.id
Instance ID	resource.instanceDetails.instanceId
Instance image ID	resource.instanceDetails.imageId
Instance tag key	resource.instanceDetails.tags.key
Instance tag value	resource.instanceDetails.tags.value
IPv6 address	resource.instanceDetails.networkInterfaces.ipv6Addresses
Private IPv4 address	resource.instanceDetails.networkInterfaces.privateIpAddresses.privateIpAddress
Public DNS name	resource.instanceDetails.networkInterfaces.publicDnsName
Public IP	resource.instanceDetails.networkInterfaces.publicIp
Security group ID	resource.instanceDetails.networkInterfaces.securityGroups.groupId
Security group name	resource.instanceDetails.networkInterfaces.securityGroups.groupName
Subnet ID	resource.instanceDetails.networkInterfaces.subnetId
VPC ID	resource.instanceDetails.networkInterfaces.vpcId
Outpost ARN	resource.instanceDetails.outpostARN

Console field name	JSON field name
Resource type	resource.resourceType
Bucket permissions	resource.s3BucketDetails.publicAccess.effectivePermission
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.countryName
API caller IPv4 address	service.action.awsApiCallAction.remoteIpDetails.ipAddressV4
API caller IPv6 address	service.action.awsApiCallAction.remoteIpDetails.ipAddressV6
API caller ASN ID	service.action.awsApiCallAction.remoteIpDetails.organization.asn
API caller ASN name	service.action.awsApiCallAction.remoteIpDetails.organization.asnOrg

Console field name	JSON field name
API caller service name	service.action.awsApiCallAction.serviceName
DNS request domain	service.action.dnsRequestAction.domain
DNS request domain suffix	service.action.dnsRequestAction.domainWithSuffix
Network connection blocked	service.action.networkConnectionAction.blocked
Network connection direction	service.action.networkConnectionAction.connectionDirection
Network connection local port	service.action.networkConnectionAction.localPortDetails.port
Network connection protocol	service.action.networkConnectionAction.protocol
Network connection city	service.action.networkConnectionAction.remoteIpDetails.city.cityName
Network connection country	service.action.networkConnectionAction.remoteIpDetails.country.countryName
Network connection remote IPv4 address	service.action.networkConnectionAction.remoteIpDetails.ipAddressV4
Network connection remote IPv6 address	service.action.networkConnectionAction.remoteIpDetails.ipAddressV6
Network connection remote IP ASN ID	service.action.networkConnectionAction.remoteIpDetails.organization.asn
Network connection remote IP ASN name	service.action.networkConnectionAction.remoteIpDetails.organization.asnOrg

Console field name	JSON field name
Network connection remote port	service.action.networkConnectionAction.remotePortDetails.port
Remote account affiliated	service.action.awsApiCallAction.remoteAccountDetails.affiliated
Kubernetes API caller IPv4 address	service.action.kubernetesApiCallAction.remoteIpDetails.ipAddressV4
Kubernetes API caller IPv6 address	service.action.kubernetesApiCallAction.remoteIpDetails.ipAddressV6
Kubernetes namespace	service.action.kubernetesApiCallAction.namespace
Kubernetes API caller ASN ID	service.action.kubernetesApiCallAction.remoteIpDetails.organization.asn
Kubernetes API call request URI	service.action.kubernetesApiCallAction.requestUri
Kubernetes API status code	service.action.kubernetesApiCallAction.statusCode
Network connection local IPv4 address	service.action.networkConnectionAction.localIpDetails.ipAddressV4
Network connection local IPv6 address	service.action.networkConnectionAction.localIpDetails.ipAddressV6
Protocol	service.action.networkConnectionAction.protocol
API call service name	service.action.awsApiCallAction.serviceName
API caller account ID	service.action.awsApiCallAction.remoteAccountDetails.accountId

Console field name	JSON field name
Threat list name	service.additionalInfo.threatListName
Resource role	service.resourceRole
EKS cluster name	resource.eksClusterDetails.name
Kubernetes workload name	resource.kubernetesDetails.kubernetesWorkloadDetails.name
Kubernetes workload namespace	resource.kubernetesDetails.kubernetesWorkloadDetails.namespace
Kubernetes user name	resource.kubernetesDetails.kubernetesUserDetails.username
Kubernetes container image	resource.kubernetesDetails.kubernetesWorkloadDetails.containers.image
Kubernetes container image prefix	resource.kubernetesDetails.kubernetesWorkloadDetails.containers.imagePrefix
Scan ID	service.ebsVolumeScanDetails.scanId
Threat name	service.ebsVolumeScanDetails.scanDetections.threatDetectedByName.threatNames.name
Threat severity	service.ebsVolumeScanDetails.scanDetections.threatDetectedByName.threatNames.severity
File SHA	service.ebsVolumeScanDetails.scanDetections.threatDetectedByName.threatNames.filePaths.hash
ECS cluster name	resource.ecsClusterDetails.name

Console field name	JSON field name
ECS container image	resource.ecsClusterDetails.taskDetails.containers.image
ECS task definition ARN	resource.ecsClusterDetails.taskDetails.definitionArn
Standalone container image	resource.containerDetails.image
Database Instance Id	resource.rdsDbInstanceDetails.dbInstanceIdentifier
Database Cluster Id	resource.rdsDbInstanceDetails.dbClusterIdentifier
Database Engine	resource.rdsDbInstanceDetails.engine
Database user	resource.rdsDbUserDetails.user
Database instance tag key	resource.rdsDbInstanceDetails.tags.key
Database instance tag value	resource.rdsDbInstanceDetails.tags.value
Executable SHA-256	service.runtimeDetails.process.executableSha256
Process name	service.runtimeDetails.process.name
Executable path	service.runtimeDetails.process.executablePath
Lambda function name	resource.lambdaDetails.functionName
Lambda function ARN	resource.lambdaDetails.functionArn
Lambda function tag key	resource.lambdaDetails.tags.key
Lambda function tag value	resource.lambdaDetails.tags.value
DNS request domain	service.action.dnsRequestAction.domainWithSuffix

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. You can edit the suppression rules at any time.

Suppressed findings are not sent to AWS Security Hub, Amazon Simple Storage Service, Amazon Detective, or Amazon EventBridge, reducing finding noise level if you consume GuardDuty findings via Security Hub, a third-party SIEM, or other alerting and ticketing applications. If you've enabled GuardDuty Malware Protection, the suppressed GuardDuty findings won't initiate a malware scan.

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 – 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.OutsideAWS`. 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.OutsideAWS
   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 – 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 – 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 – 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
  ```

Recommended suppression rules for Runtime Monitoring findings

- PrivilegeEscalation:Runtime/DockerSocketAccessed gets generated when a process inside a container communicates with the Docker socket. There may be containers in your environment that may need to access the Docker socket for legitimate reasons. Access from such containers will generate PrivilegeEscalation:Runtime/DockerSocketAccessed finding. If this is a case in your AWS environment, we recommend that you set up a suppression rule for this finding type. The first criteria should use the **Finding type** field with value equal to `PrivilegeEscalation:Runtime/DockerSocketAccessed`. The second filter criteria is **Executable path** field with value equal to the process's `executablePath` in the generated finding. Alternatively, the second filter criteria can use **Executable SHA-256** field with value equal to the process's `executableSha256` in the generated finding.

- Kubernetes clusters run their own DNS servers as pods, such as `coredns`. Therefore, for each DNS lookup from a pod, GuardDuty captures two DNS events – one from the pod and the other from the server pod. This may generate duplicates for the following DNS findings:
  - Backdoor:Runtime/C&CActivity.B!DNS
  - CryptoCurrency:Runtime/BitcoinTool.B!DNS
  - Impact:Runtime/AbusedDomainRequest.Reputation
  - Impact:Runtime/BitcoinDomainRequest.Reputation
  - Impact:Runtime/MaliciousDomainRequest.Reputation
  - Impact:Runtime/SuspiciousDomainRequest.Reputation
  - Trojan:Runtime/BlackholeTraffic!DNS
  - Trojan:Runtime/DGADomainRequest.C!DNS
  - Trojan:Runtime/DriveBySourceTraffic!DNS
  - Trojan:Runtime/DropPoint!DNS
  - Trojan:Runtime/PhishingDomainRequest!DNS

  The duplicate findings will include pod, container, and process details that correspond to your DNS server pod. You may set up a suppression rule to suppress these duplicate findings using these fields. The first filter criteria should use the **Finding type** field with value equal to a DNS finding type from the list of findings provided earlier in this section. The second filter criteria could be either **Executable path** with value equal to your DNS server's `executablePath` or **Executable SHA-256** with value equal to your DNS server's `executableSHA256` in the generated finding. As an optional third filter criteria, you can use **Kubernetes container image** field with value equal to the container image of your DNS server pod in the generated finding.

Creating suppression rules

Choose your preferred access method to create a suppression rule for GuardDuty finding types.

Console

You can visualize, create, and manage suppression rules using the GuardDuty console. Suppression rules are generated in the same manner as filters, and your existing saved filters can be used as suppression rules. For more information about creating filters, see Filtering findings.

To create a suppression rule using the console:

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. On the **Findings** page, choose **Suppress findings** to open the suppression rule panel.
3. To open the filter criteria menu, enter the `filter criteria` in the **Add filter criteria**. You can choose a criterion from the list. Enter a valid value for the chosen criterion.

   > **Note**
   >
   > To determine the valid value, view the findings table and choose a finding that you want to suppress. Review it's details in the findings panel.

   You can add multiple filter criteria and ensure that only those findings appear in the table that you want to suppress.
4. Enter a **Name** and **Description** for the suppression rule. Valid characters include alphanumeric characters, period (.), dash (-), underscore (_), and whitespaces.
5. Choose **Save**.

You can also create a suppression rule from an existing saved filter. For more information about creating filters, see Filtering findings.

To create a suppression rule from a saved filter:

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. On the **Findings** page, choose **Suppress Findings** to open the suppression rule panel.
3. From the **Saved rules** drop down, choose a saved filter.
4. You can also add new filter criteria. If you don't need additional filter criteria, skip this step.

To open the filter criteria menu, enter the **`filter criteria`** in the **Add filter criteria**. You can choose a criterion from the list. Enter a valid value for the chosen criterion.

> **Note**
>
> To determine the valid value, view the findings table and choose a finding that you want to suppress. Review it's details in the findings panel.

5. Enter a **Name** and **Description** for the suppression rule. Valid characters include alphanumeric characters, period (.), dash (-), underscore (_), and whitespaces.

6. Choose **Save**.

API/CLI

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 low-severity findings that has a DNS request to the test.example.com domain. For medium-severity findings, the input list will be `["4", "5", "7"]`. For high-severity findings, the input list will be `["6", "7", "8"]`. You can also filter on the basis of any one value in the list.

```
{
    "Criterion": {
        "service.archived": {
            "Eq": [
                "false"
            ]
        },
        "service.action.dnsRequestAction.domain": {
            "Eq": [
                "test.example.com"
            ]
        },
        "severity": {
            "Eq": [
                "1",
                "2",
                "3"
            ]
        }
    }
}
```

For a list of JSON field names and their console equivalent see Filter attributes.

To test your filter criteria, use the same JSON criterion in the ListFindings API, and confirm that the correct findings have been selected. To test your filter criteria using AWS CLI follow the example using your own detectorId and .json file.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
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.

   To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   ```
   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 view the 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.

Deleting suppression rules

Choose your preferred access method to delete a suppression rule for GuardDuty finding types.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. On the **Findings** page, choose **Suppress Findings** to open the suppression rule panel.
3. From the **Saved rules** drop down, choose a saved filter.
4. Choose **Delete rule**.

API/CLI

Run the DeleteFilter API. Specify the filter name and the associated detector ID for the particular Region.

Alternatively, you can use the following AWS CLI example by replacing the values formatted in *red*:

```
aws guardduty delete-filter --region us-east-1 --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --filter-name filterName
```

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

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 IP list

Threat lists consist of known malicious IP addresses. This list can be supplied by third-party threat intelligence or created specifically for your organization. In addition to generating findings because of a potentially suspicious activity, GuardDuty also generates findings based on these 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 an activity that involves IP addresses and CIDR ranges in your threat lists; the findings are not generated based on the domain names. At any given point in time, you can have up to six uploaded threat lists per AWS account per each 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 account accounts can add and manage trusted IP lists and threat lists. Trusted IP lists and threat lists that are uploaded by the administrator account 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 account's threat lists and does not generate findings based on activity that involves IP addresses from the administrator account's trusted IP lists. For more information, see Managing multiple accounts in Amazon GuardDuty.

List formats

GuardDuty accepts lists in the following formats.

The maximum size of each file that hosts your trusted IP list or threat IP list is 35MB. In your trusted IP lists and threat IP lists, IP addresses and CIDR ranges must appear one per line. Only IPv4 addresses are accepted.

- Plaintext (TXT)

This format supports both CIDR block and individual IP addresses. The following sample list uses the Plaintext (TXT) format.

```
192.0.2.0/24
198.51.100.1
203.0.113.1
```

- Structured Threat Information Expression (STIX)

  This format supports both CIDR block and individual IP addresses. The following sample list uses the STIX format.

```
<?xml version="1.0" encoding="UTF-8"?>
<stix:STIX_Package
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    xmlns:ttp="http://stix.mitre.org/TTP-1"
    xmlns:cybox="http://cybox.mitre.org/cybox-2"
    xmlns:AddressObject="http://cybox.mitre.org/objects#AddressObject-2"
    xmlns:cyboxVocabs="http://cybox.mitre.org/default_vocabularies-2"
    xmlns:stixVocabs="http://stix.mitre.org/default_vocabularies-1"
    xmlns:example="http://example.com/"
    xsi:schemaLocation="
    http://stix.mitre.org/stix-1 http://stix.mitre.org/XMLSchema/core/1.2/
stix_core.xsd
    http://stix.mitre.org/Campaign-1 http://stix.mitre.org/XMLSchema/campaign/1.2/
campaign.xsd
    http://stix.mitre.org/Indicator-2 http://stix.mitre.org/XMLSchema/indicator/2.2/
indicator.xsd
    http://stix.mitre.org/TTP-2 http://stix.mitre.org/XMLSchema/ttp/1.2/ttp.xsd
    http://stix.mitre.org/default_vocabularies-1 http://stix.mitre.org/XMLSchema/
default_vocabularies/1.2.0/stix_default_vocabularies.xsd
    http://cybox.mitre.org/objects#AddressObject-2 http://cybox.mitre.org/XMLSchema/
objects/Address/2.1/Address_Object.xsd"
    id="example:STIXPackage-a78fc4e3-df94-42dd-a074-6de62babfe16"
    version="1.2">
    <stix:Observables cybox_major_version="1" cybox_minor_version="1">
        <cybox:Observable id="example:observable-80b26f43-
dc41-43ff-861d-19aff31e0236">
            <cybox:Object id="example:object-161a5438-1c26-4275-ba44-a35ba963c245">
```

```
                <cybox:Properties xsi:type="AddressObject:AddressObjectType"
 category="ipv4-addr">

 <AddressObject:Address_Valuecondition="InclusiveBetween">192.0.2.0##comma##192.0.2.255</
AddressObject:Address_Value>
                </cybox:Properties>
            </cybox:Object>
        </cybox:Observable>
        <cybox:Observable id="example:observable-b442b399-aea4-436f-bb34-
b9ef6c5ed8ab">
            <cybox:Object id="example:object-b422417f-bf78-4b34-ba2d-de4b09590a6d">
                <cybox:Properties xsi:type="AddressObject:AddressObjectType"
 category="ipv4-addr">
                    <AddressObject:Address_Value>198.51.100.1</
AddressObject:Address_Value>
                </cybox:Properties>
            </cybox:Object>
        </cybox:Observable>
        <cybox:Observable
 id="example:observable-1742fa06-8b5e-4449-9d89-6f9f32595784">
            <cybox:Object id="example:object-dc73b749-8a31-46be-803f-71df77565391">
                <cybox:Properties xsi:type="AddressObject:AddressObjectType"
 category="ipv4-addr">
                    <AddressObject:Address_Value>203.0.113.1</
AddressObject:Address_Value>
                </cybox:Properties>
            </cybox:Object>
        </cybox:Observable>
    </stix:Observables>
</stix:STIX_Package>
```

- Open Threat Exchange (OTX)^{TM} CSV

  This format supports both CIDR block and individual IP addresses. The following sample list uses the OTX^{TM} CSV format.

  ```
  Indicator type, Indicator, Description
  CIDR, 192.0.2.0/24, example
  IPv4, 198.51.100.1, example
  IPv4, 203.0.113.1, example
  ```

- FireEye^{TM} iSIGHT Threat Intelligence CSV

This format supports both CIDR block and individual IP addresses. The following sample list uses a `FireEye`™ CSV format.

```
reportId, title, threatScape, audience, intelligenceType, publishDate, reportLink,
 webLink, emailIdentifier, senderAddress, senderName, sourceDomain, sourceIp,
 subject, recipient, emailLanguage, fileName, fileSize, fuzzyHash, fileIdentifier,
 md5, sha1, sha256, description, fileType, packer, userAgent, registry,
 fileCompilationDateTime, filePath, asn, cidr, domain, domainTimeOfLookup,
 networkIdentifier, ip, port, protocol, registrantEmail, registrantName, networkType,
 url, malwareFamily, malwareFamilyId, actor, actorId, observationTime

01-00000001, Example, Test, Operational, threat, 1494944400,
 https://www.example.com/report/01-00000001, https://www.example.com/
report/01-00000001, , , , , , , , , , , , , , , , , , , , , , , , 192.0.2.0/24, , ,
 Related, , , , , , network, , Ursnif, 21a14673-0d94-46d3-89ab-8281a0466099, , ,
 1494944400

01-00000002, Example, Test, Operational, threat, 1494944400,
 https://www.example.com/report/01-00000002, https://www.example.com/
report/01-00000002, , , , , , , , , , , , , , , , , , , , , , , , , , , Related,
 198.51.100.1, , , , , network, , Ursnif,
 12ab7bc4-62ed-49fa-99e3-14b92afc41bf, , ,1494944400

01-00000003, Example, Test, Operational, threat, 1494944400,
 https://www.example.com/report/01-00000003, https://www.example.com/
report/01-00000003, , , , , , , , , , , , , , , , , , , , , , , , , , , Related,
 203.0.113.1, , , , , network, , Ursnif, 8a78c3db-7bcb-40bc-a080-75bd35a2572d, , ,
 1494944400
```

- Proofpoint™ ET Intelligence Feed CSV

  This format supports only individual IP addresses. The following sample list uses the `Proofpoint` CSV format. The `ports` parameter is optional. If you skip the port, ensure to leave a trailing comma (,) at the end.

```
ip, category, score, first_seen, last_seen, ports (|)
198.51.100.1, 1, 100, 2000-01-01, 2000-01-01,
203.0.113.1, 1, 100, 2000-01-01, 2000-01-01, 80
```

- AlienVault™ Reputation Feed

This format supports only individual IP addresses. The following sample list uses the `AlienVault` format.

```
198.51.100.1#4#2#Malicious Host#US##0.0,0.0#3
203.0.113.1#4#2#Malicious Host#US##0.0,0.0#3
```

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 attached AmazonGuardDutyFullAccess managed policy 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 adding, activating, deleting, and updating the location or name of the lists), make sure that the following actions are present in the permissions policy attached to a user, group, or role:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:PutRolePolicy",
        "iam:DeleteRolePolicy"
    ],
    "Resource": "arn:aws:iam::555555555555: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": "*"
}
```

Adding and activating a trusted IP list or a threat IP list

Choose one of the following access methods to add and activate a trusted IP list or a threat IP list.

Console

(Optional) step 1: Fetching location URL of your list

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**.
3. Choose the Amazon S3 bucket name that contains the specific list that you want to add.
4. Choose the object (list) name to view its details.
5. Under the **Properties** tab, copy the **S3 URI** for this object.

Step 2: Adding a trusted IP list or a threat list

> **Important**
>
> By default, at any given point in time, you can have only one trusted IP list. Similarly, you can have up to six threat lists.

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

2. In the navigation pane, choose **Lists**.
3. On the **List management** page, choose **Add a trusted IP list** or **Add a threat list**.
4. Based on your selection, a dialog box will appear. Do the following steps:

   a. For **List name**, enter a name for your list.

      List naming constraints – The name of your list can include lowercase letters, uppercase letters, numbers, dash (-), and underscore (_).

   b. For **Location**, provide the location where you have uploaded your list. If you don't already have it, see Step 1: Fetching location URL of your list.

      Format of location URL

      - 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

   c. Select the **I agree** check box.

   d. Choose **Add list**. By default, the **Status** of the added list is **Inactive**. For the list to be effective, you must activate the list.

Step 3: Activating a trusted IP list or a threat list

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Lists**.
3. On the **List management** page, select the list that you want to activate.
4. Choose **Actions**, and then choose **Activate**. It may take up to 15 min for the list to be effective.

API/CLI

For trusted IP lists

- Run CreateIPSet. Make sure to provide the `detectorId` of the member account for which you want to create this trusted IP list.

List naming constraints – The name of your list can include lowercase letters, uppercase letters, numbers, dash (-), and underscore (_).

- Alternatively, you can do this by running the following AWS Command Line Interface command and make sure to replace the `detector-id` with the detector ID of the member account for which you will update the trusted IP list.

```
aws guardduty create-ip-set --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --name AnyOrganization List --format Plaintext --location https://
s3.amazonaws.com/DOC-EXAMPLE-BUCKET2/DOC-EXAMPLE-SOURCE-FILE.format --
activate
```

For threat lists

- Run CreateThreatIntelSet. Make sure to provide the `detectorId` of the member account for which you want to create this threat list.
  - Alternatively, you can do this by running the following AWS Command Line Interface command. Make sure to provide the `detectorId` of the member account for which you want to create a threat list.

```
aws guardduty create-threat-intel-set --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --name AnyOrganization List --
format Plaintext --location https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET2/
DOC-EXAMPLE-SOURCE-FILE.format --activate
```

> **Note**
>
> After you activate or update any IP list, GuardDuty might take up to 15 minutes to sync the list.

Updating trusted IP lists and threat lists

You can update the name of a list or the IP addresses added to a list that has already been added and activated. If you update a list, you must activate it again for GuardDuty to use the latest version of the list.

Choose one of the access methods to update a trusted IP or threat list.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Lists**.
3. On the **List management** page, select the trusted IP set or a threat list that you want to update.
4. Choose **Actions**, and then choose **Edit**.
5. In the **Update list** dialog box, update the information as needed.

   List naming constraints – The name of your list can include lowercase letters, uppercase letters, numbers, dash (-), and underscore (_).
6. Choose the **I agree** check box, and then choose **Update list**. The value in the **Status** column will change to **Inactive**.
7. **Reactivating the updated list**

   a. On the **List management** page, select the list that you want to activate again.

   b. Choose **Actions**, and then choose **Activate**.

API/CLI

1. Run UpdateIPSet to update a trusted IP list.

   - Alternatively, you can run the following AWS CLI command to update a trusted IP list and make sure to replace the `detector-id` with the detector ID of the member account for which you will update the trusted IP list.

     ```
     aws guardduty update-ip-set --detector-id 12abc34d567e8fa901bc2d34e56789f0
      --name AnyOrganization List --ip-set-id d4b94fc952d6912b8f3060768example --
     activate
     ```
2. Run UpdateThreatIntelSet to update a threat list

   - Alternatively, you can run the following AWS CLI command to update a threat list and make sure to replace the `detector-id` with the detector ID of the member account for which you will update the threat list.

```
aws guardduty update-threatintel-set --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --name AnyOrganization List --threat-
intel-set-id d4b94fc952d6912b8f3060768example --activate
```

De-activating or deleting a trusted IP list or threat list

Choose one of the access methods to delete (by using the console) or deactivate (by using API/CLI) a trusted IP list, or a threat list.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Lists**.
3. On the **List management** page, select the list that you want to delete.
4. Choose **Actions**, and then choose **Delete**.
5. Confirm the action and choose **Delete**. The specific list will no longer be available in the table.

API/CLI

1. **For a trusted IP list**

   Run UpdateIPSet to update a trusted IP list.

   - Alternatively, you can run the following AWS CLI command to update a trusted IP list and make sure to replace the `detector-id` with the detector ID of the member account for which you will update the trusted IP list.

     To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

     ```
     aws guardduty update-ip-set --detector-id 12abc34d567e8fa901bc2d34e56789f0
      --name AnyOrganization List --ip-set-id d4b94fc952d6912b8f3060768example --
     no-activate
     ```

2. **For a threat list**

   Run UpdateThreatIntelSet to update a threat list

- Alternatively, you can run the following AWS CLI command to update a trusted IP list and make sure to replace the `detector-id` with the detector ID of the member account for which you will update the threat list.

  ```
  aws guardduty update-threatintel-set --detector-
  id 12abc34d567e8fa901bc2d34e56789f0 --name AnyOrganization List --threat-
  intel-set-id d4b94fc952d6912b8f3060768example --no-activate
  ```

Exporting findings

GuardDuty retains the generated findings for a period of 90 days. GuardDuty exports the active findings to Amazon EventBridge (EventBridge). You can optionally export the generated findings to an Amazon Simple Storage Service (Amazon S3) bucket. This will help you to track the historical data of potentially suspicious activities in your account and evaluate whether the recommended remediation steps were successful.

Any 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 the active findings are exported to EventBridge. The frequency that you select applies to the exporting of new occurrences of existing findings to EventBridge, your S3 bucket (when configured), and Detective (when integrated). For information about how GuardDuty aggregates multiple occurrences of existing findings, see GuardDuty finding aggregation.

When you configure settings to export findings to an Amazon S3 bucket, GuardDuty uses AWS Key Management Service (AWS KMS) to encrypt the findings data in your S3 bucket. This requires you to add permissions to your S3 bucket and the AWS KMS key so that GuardDuty can use them to export findings in your account.

Contents

Considerations

Before proceeding with the prerequisites and steps to export findings, consider the following key concepts:

- **Export settings are regional** – You need to configure export options in each Region where you use GuardDuty.
- **Exporting findings to Amazon S3 buckets in different AWS Regions (cross-Region)** – GuardDuty supports the following export settings:
  - Your Amazon S3 bucket or object, and AWS KMS key must belong to the same AWS Region.
  - For the findings generated in a commercial Region, you can choose to export these findings to an S3 bucket in any commercial Region. However, you can't export these findings to an S3 bucket in an opt-in Region.
  - For the findings generated in an opt-in Region, you can choose to export these findings to the same opt-in Region where they're generated or any commercial Region. However, you can't export findings from one opt-in Region to another opt-in Region.
- **Permissions to export findings** – To configure settings for exporting active findings, your S3 bucket must have permissions that allows GuardDuty to upload objects. You must also have an AWS KMS key that GuardDuty can use to encrypt findings.
- **Archived findings are not exported** – The default behavior is that the archived findings, including new instances of suppressed findings, are not exported.

  To export an archived finding, you must *un-archive* it. This will change its status to **Active**. Based on the export frequency, the finding will be exported to the configured S3 bucket.
- **GuardDuty administrator account can export findings generated in associated member accounts** – When you configure export findings in an administrator account, all the findings from the associated member accounts that are generated in the same Region are also exported to the same location that you configured for the administrator account account. For more information, see Understanding the relationship between GuardDuty administrator account and member accounts.

Step 1 – Permissions required to export findings

When you configure settings for exporting findings, you select an Amazon S3 bucket where you can store the findings and an AWS KMS key to use for data encryption. In addition to permissions for GuardDuty actions, you must also have permissions to the following actions to successfully configure settings to export findings:

- s3:GetBucketLocation
- s3:PutObject

Step 2 – Attaching policy to your KMS key

GuardDuty encrypts the findings data in your bucket by using AWS Key Management Service. To successfully configure the settings, you must first give GuardDuty permission to use a KMS key. You can grant the permissions by attaching the policy to your KMS key.

When you use a KMS key from another account, you need to apply the key policy by logging in to the AWS account that owns the key. When you configure the settings to export findings, you'll also need the key ARN from the account that owns the key.

To modify the KMS key policy for GuardDuty to encrypt your exported findings

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. Select an existing KMS key or perform the steps to Create a new key in the *AWS Key Management Service Developer Guide*, that you will use to encrypt the exported findings.

   > **Note**
   >
   > The AWS Region of your KMS key and the Amazon S3 bucket must be the same.

   You can use the same S3 bucket and KMS key pair to export the findings from any applicable Region. For more information, see Considerations for exporting findings across Regions.
4. In the **Key policy** section, choose **Edit**.

   If **Switch to policy view** is displayed, choose it to display the **Key policy**, and then choose **Edit**.
5. Copy the following policy block to your KMS key policy, to grant GuardDuty permission to use your key.

```
{
    "Sid": "AllowGuardDutyKey",
    "Effect": "Allow",
    "Principal": {
        "Service": "guardduty.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey",
    "Resource": "KMS key ARN",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "123456789012",
            "aws:SourceArn":
 "arn:aws:guardduty:Region2:123456789012:detector/SourceDetectorID"
        }
    }
}
```

6. Edit the policy by replacing the following values that are formatted in `red` in the policy example:

   1. Replace *`KMS key ARN`* with the Amazon Resource Name (ARN) of the KMS key. To locate the key ARN, see Finding the key ID and ARN in the *AWS Key Management Service Developer Guide*.
   2. Replace *`123456789012`* with the AWS account ID that owns the GuardDuty account exporting the findings.
   3. Replace *`Region2`* with the AWS Region where the GuardDuty findings are generated.
   4. Replace *`SourceDetectorID`* with the `detectorID` of the GuardDuty account in the specific Region where the findings generated.

      To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

Note

If you're using GuardDuty in an opt-in 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"`. For information about endpoints for each opt-in Region, see GuardDuty endpoints and quotas.

7. If you added the policy statement before the final statement, add a comma before adding this statement. Make sure that the JSON syntax of your KMS key policy is valid.

   Choose **Save**.
8. (Optional) copy the key ARN to a notepad for use in the later steps.

Step 3 – Attaching policy to Amazon S3 bucket

Add permissions to the Amazon S3 bucket to which you will export findings so that GuardDuty can upload objects to this S3 bucket. Independent of using an Amazon S3 bucket that belongs to either your account or in a different AWS account, you must add these permissions.

If at any point in time, you decide to export findings to a different S3 bucket, then to continue exporting findings, you must add permissions to that S3 bucket and configure the export findings settings again.

If you do not already have an Amazon S3 bucket where you want to export these findings, see Creating a bucket in the *Amazon S3 User Guide*.

To attach permissions to your S3 bucket policy

1. Perform the steps under To create or edit a bucket policy in the *Amazon S3 User Guide*, until the **Edit bucket policy** page appears.
2. The **example policy** shows how grant GuardDuty permission to export findings to your Amazon S3 bucket. If you change the path after you configure export findings, then you must modify the policy to grant permission to the new location.

   Copy the following **example policy** and paste it into the **Bucket policy editor**.

   If you added the policy statement before the final statement, add a comma before adding this statement. Make sure that the JSON syntax of your KMS key policy is valid.

   S3 bucket example policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGuardDutygetBucketLocation",
            "Effect": "Allow",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:GetBucketLocation",
            "Resource": "Amazon S3 bucket ARN",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012",
                    "aws:SourceArn":
 "arn:aws:guardduty:Region2:123456789012:detector/SourceDetectorID"

                }
            }
        },
        {
            "Sid": "AllowGuardDutyPutObject",
            "Effect": "Allow",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "Amazon S3 bucket ARN/[optional prefix]/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012",
                    "aws:SourceArn":
 "arn:aws:guardduty:Region2:123456789012:detector/SourceDetectorID"

                }
            }
        },
        {
            "Sid": "DenyUnencryptedUploadsThis is optional",
            "Effect": "Deny",
            "Principal": {
                "Service": "guardduty.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "Amazon S3 bucket ARN/[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": "Amazon S3 bucket ARN/[optional prefix]/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption-aws-kms-key-id": "KMS key ARN"
                }
            }
        },
        {
            "Sid": "DenyNon-HTTPS",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "Amazon S3 bucket ARN/[optional prefix]/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

3. Edit the policy by replacing the following values that are formatted in *red* in the policy example:

   1. Replace *Amazon S3 bucket ARN* with the Amazon Resource Name (ARN) of the Amazon S3 bucket. You can find the **Bucket ARN** on the **Edit bucket policy** page in the https://console.aws.amazon.com/s3/ console.

   2. Replace *123456789012* with the AWS account ID that owns the GuardDuty account exporting the findings.

   3. Replace *Region2* with the AWS Region where the GuardDuty findings are generated.

   4. Replace *SourceDetectorID* with the `detectorID` of the GuardDuty account in the specific Region where the findings generated.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

5. Replace *`[optional prefix]`* part of the *`S3 bucket ARN/[optional prefix]`* placeholder value with an optional folder location to which you want to export the findings. For more information about the use of prefixes, see Organizing objects using prefixes in the *Amazon S3 User Guide*.

   When you provide an optional folder location that doesn't exist already, GuardDuty will create that location only if the account associated with the S3 bucket is the same as the account exporting the findings. When you export findings to an S3 bucket that belongs to another account, the folder location must exist already.

6. Replace *`KMS key ARN`* with the Amazon Resource Name (ARN) of the KMS key associated with the encryption of the findings exported to the S3 bucket. To locate the key ARN, see Finding the key ID and ARN in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> If you're using GuardDuty in an opt-in 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"`. For information about endpoints for each opt-in Region, see GuardDuty endpoints and quotas.

4. Choose **Save**.

Step 4 - Exporting findings to an S3 bucket (Console)

GuardDuty permits you to export findings to an existing bucket in another AWS account.

When creating a new S3 bucket or choosing an existing bucket in your account, you can add an optional prefix. When configuring export findings, GuardDuty creates a new folder in the S3 bucket for your findings. The prefix will be appended to the default folder structure that GuardDuty created. For example, the format of the optional prefix `/AWSLogs/`*`123456789012`*`/GuardDuty/`*`Region`*.

⚠ **Important**

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

Before completing these steps, make sure you have attached the respective policies to your KMS key and existing S3 bucket.

To configure export findings

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 **Findings export options**, for **S3 bucket**, choose **Configure now** (or **Edit**, as needed).
4. For **S3 bucket ARN**, enter the **`bucket ARN`**. To find the bucket ARN, see Viewing the properties for an S3 bucket in the *Amazon S3 User Guide*. In the **Permissions** tab of the associated bucket's **Properties** page in the https://console.aws.amazon.com/guardduty/ console.
5. For **KMS key ARN**, enter the **`key ARN`**. To locate the key ARN, see Finding the key ID and ARN in the *AWS Key Management Service Developer Guide*.
6. **Attach policies**

   - Perform the steps to attach the S3 bucket policy. For more information, see Step 3 – Attaching policy to Amazon S3 bucket.

   - Perform the steps to attach the KMS key policy. For more information, see Step 2 – Attaching policy to your KMS key.
7. Choose **Save**.

Step 5 – Setting frequency to export 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. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. Choose **Settings**.
3. In the **Findings export options** section, choose **Frequency for updated findings**. This sets the frequency for exporting updated Active findings to both EventBridge and Amazon S3. You can choose from the following:
   - **Update EventBridge and S3 every 15 minutes**
   - **Update EventBridge and S3 every 1 hour**
   - **Update CWE and S3 every 6 hours (default)**
4. Choose **Save changes**.

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. For more information, see GuardDuty finding aggregation.

> **Note**
>
> If your account is a GuardDuty delegated administrator, the CloudWatch events are published to your account as well as to the member account where the finding was generated.

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).

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.

> **Note**
>
> By default, the frequency of notifications about the newly–generated findings is 5 minutes. This frequency cannot be updated.

Notifications for subsequent finding occurrences

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

Only an administrator account 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. The frequency value set by the administrator account account in its own account is imposed on GuardDuty functionality in all its member accounts. If a user from an administrator account account sets this frequency value to 1 hour, all member accounts will also have the 1 hour frequency of receiving notifications about the subsequent finding occurrences. For more information, see Managing multiple accounts in Amazon GuardDuty.

Note

As an administrator account, you can customize the default frequency of notifications about the subsequent finding occurrences. Possible values are 15 minutes, 1 hour, or the default 6 hours. For information about setting the frequency for these notifications, see Step 5 – Setting frequency to export updated active findings.

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, see Getting started in the *Amazon Simple Notification Service Developer 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%3D<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.

Understanding CloudWatch Logs and reasons for skipping resources during Malware Protection scan

GuardDuty Malware Protection publishes events to your Amazon CloudWatch log group **/aws/guardduty/malware-scan-events**. For each of the events related to the malware scan, you can monitor the status and scan result of your impacted resources. Certain Amazon EC2 resources and Amazon EBS volumes may have been skipped during the Malware Protection scan.

Auditing CloudWatch Logs in GuardDuty Malware Protection

There are three types of scan events supported in the **/aws/guardduty/malware-scan-events** CloudWatch log group.

Malware Protection scan event name	Explanation
`EC2_SCAN_STARTED`	Created when an GuardDuty Malware Protection is initiating the process of malware scan, such as preparing to take a snapshot of an EBS volume.
`EC2_SCAN_COMPLETED`	Created when GuardDuty Malware Protectio n scan completes for at least one of the EBS volumes of the impacted resource. This event also includes the `snapshotId` that belongs to the scanned EBS volume. After the scan completes, the scan result will either be `CLEAN`, `THREATS_FOUND` , or `NOT_SCANNED` .
`EC2_SCAN_SKIPPED`	Created when GuardDuty Malware Protection scan skips all the EBS volumes of the impacted resource. To identify the skip reason, select

Malware Protection scan event name	Explanation
	the corresponding event, and view the details. For more information on skip reasons, see Reasons for skipping resource during malware scan below.

Note

If you're using an AWS Organizations, CloudWatch log events from member accounts in Organizations get published to both administrator account and member account's log group.

Choose your preferred access method to view and query CloudWatch events.

Console

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, under **Logs**, choose **Log groups**. Choose the **/aws/guardduty/malware-scan-events** log group to view the scan events for GuardDuty Malware Protection.

   To run a query, choose **Log Insights**.

   For information about running a query, see Analyzing log data with CloudWatch Logs Insights in the *Amazon CloudWatch User Guide*.
3. Choose **Scan ID** to monitor the details of the impacted resource and malware findings. For example, you can run the following query to filter the CloudWatch log events by using `scanId`. Make sure to use your own valid *`scan-id`*.

   ```
   fields @timestamp, @message, scanRequestDetails.scanId as scanId
   | filter scanId like "77a6f6115da4bd95f4e4ca398492bcc0"
   | sort @timestamp asc
   ```

API/CLI

- To work with log groups, see Search log entries using the AWS CLI in the *Amazon CloudWatch User Guide*.

  Choose the **/aws/guardduty/malware-scan-events** log group to view the scan events for GuardDuty Malware Protection.
- To view and filter log events, see GetLogEvents and FilterLogEvents, respectively, in the *Amazon CloudWatch API Reference*.

GuardDuty Malware Protection log retention

The default log retention period for **/aws/guardduty/malware-scan-events** log group is 90 days, after which the log events are deleted automatically. To change the log retention policy for your CloudWatch log group, see Change log data retention in CloudWatch Logs or PutRetentionPolicy.

Reasons for skipping resource during malware scan

In the events related to the malware scan, certain EC2 resources and EBS volumes may have been skipped during the scanning process. The following table lists the reasons why GuardDuty Malware Protection may not scan the resources. If applicable, use the proposed steps to resolve these issues, and scan these resources the next time GuardDuty Malware Protection initiates a malware scan. The other issues are used to inform you about the course of events and are non-actionable.

Reasons for skipping	Explanation	Proposed steps	
`RESOURCE_NOT_FOUND`	The `resourceArn` provided to the initiate the on-demand malware scan was not found in your AWS environment.	Validate the `resourceArn` of your Amazon EC2 instance or container workload, and try again.	
`ACCOUNT_INELIGIBLE`	The AWS account ID from which you tried initiating an On-demand malware	Verify that GuardDuty is enabled for this AWS account.	

Reasons for skipping	Explanation	Proposed steps
	scan has not enabled GuardDuty.	When you enable GuardDuty in a new AWS Region it may take up to 20 minutes to sync.
`UNSUPPORTED_KEY_ENCRYPTION`	GuardDuty Malware Protection supports volumes that are both unencrypted and encrypted with customer managed key. It doesn't support scanning EBS volumes that are encrypted using Amazon EBS encryption. Presently, there is a regional difference where this skip reason is not applicable. For more information about these AWS Regions, see Region-specific feature availability.	Replace your encryption key with a customer managed key. For more information on the types of encryption that GuardDuty supports, see Supported Amazon EBS volumes for malware scan.

Reasons for skipping	Explanation	Proposed steps
`EXCLUDED_BY_SCAN_SETTINGS`	The EC2 instance or EBS volume was excluded during the malware scan. There are two possibilities - either the tag was added to the inclusion list but the resource isn't associated with this tag, the tag was added to the exclusion list and the resource is associated with this tag, or the `GuardDuty Excluded` tag is set to `true` for this resource.	Update your scan options or the tags associated to your Amazon EC2 resource. For more information, see Scan options with user-defined tags.
`UNSUPPORTED_VOLUME_SIZE`	The volume is greater than 1024 GB.	Not actionable.
`NO_VOLUMES_ATTACHED`	GuardDuty Malware Protection found the instance in your account but no EBS volume was attached to this instance to proceed with the scan.	Not actionable.
`UNABLE_TO_SCAN`	It is an internal service error.	Not actionable.

Reasons for skipping	Explanation	Proposed steps
`SNAPSHOT_NOT_FOUND`	The snapshots created from the EBS volumes and shared with the service account was not found, and GuardDuty Malware Protection couldn't proceed with the scan.	Check CloudTrail to ensure that the snapshots were not removed intentionally.
`SNAPSHOT_QUOTA_REACHED`	You have reached the maximum volume allowed for snapshots for each Region. This prevents not just retaining but also creating new snapshots.	You can either remove old snapshots or request for quota increase. You can view the default limit for Snapshots per Region and how to request quota increase under [Service quotas](#) in the *AWS General Reference Guide*.
`MAX_NUMBER_OF_ATTACHED_VOLUMES_REACHED`	More than 11 EBS volumes were attached to an EC2 instance. GuardDuty Malware Protection scanned the first 11 EBS volumes, obtained by sorting the `deviceName` alphabetically.	Not actionable.

Reasons for skipping	Explanation	Proposed steps	
UNSUPPORTED_PRODUCT_CODE_TYPE	GuardDuty doesn't support scanning of instances with `productCode` as `marketplace`. For more information, see Paid AMIs in the *Amazon EC2 User Guide for Linux Instances*. For information on `productCode`, see ProductCode in the *Amazon EC2 API Reference*.	Not actionable.	

Reporting false positives in GuardDuty Malware Protection

GuardDuty Malware Protection scans may identify a harmless file in your Amazon EC2 instance or container workload as being malicious or harmful. To improve your experience with Malware Protection and the GuardDuty service, you can report false positive results if you believe that a file identified as being malicious or harmful during a scan doesn't actually contain malware.

False positive file submission

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. When you identify what appears to be a false positive result, contact AWS Support to initiate the process of false positive file submission.
3. Choose **Malware Scans**.
4. Choose a scan to view its **Finding ID**.
5. Provide the **Finding ID**. You must also provide the SHA-256 hash of the file. This is required to ensure that GuardDuty Malware Protection has received the correct file.

6. The AWS Support team will provide you an Amazon Simple Storage Service (S3) URL that you can use to upload the file and SHA-256 hash. Inform the AWS Support team after you have successfully uploaded the file.

   > ⚠ **Warning**
   >
   > Do not directly provide the file or SHA-256 hash to AWS Support. You should only upload the file and hash to Amazon S3 through the provided URL. If you fail to upload the file and hash within seven days of receiving the URL, it will become invalid. If the URL becomes invalid, you'll have to reach out to AWS Support to receive a new URL.

   GuardDuty keeps your file for no more than 30 days. GuardDuty team members will analyze your submission and take appropriate steps to improve your experience with Malware Protection and the GuardDuty service.

Remediating security issues discovered by GuardDuty

Amazon GuardDuty generates findings that indicate potential security issues. In this release of GuardDuty, the potential security issues indicate either a compromised EC2 instance or container workload, 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.

Contents

Remediating a potentially compromised Amazon EC2 instance

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

1. **Identify the potentially compromised Amazon EC2 instance**

   Investigate the potentially compromised instance for malware and remove any discovered malware. You may use On-demand malware scan to identify malware in the potentially compromised EC2 instance, or check AWS Marketplace to see if there are helpful partner products to identify and remove malware.
2. **Isolate the potentially compromised Amazon EC2 instance**

   If possible, use the following steps to isolate the potentially compromised instance:

   1. Create a dedicated **Isolation** security group.
   2. Create a single rule of `0.0.0.0/0 (0-65535)` for all traffic in the outbound rules.

      When this rule applies, it will convert all the existing (and new) outbound traffic to untracked, blocking any established outbound sessions. For more information, see Untracked connections.
   3. Remove all the current security group associations from the potentially compromised instance.
   4. Associate the **Isolation** security group with this instance.

      After associating, delete the rule `0.0.0.0/0 (0-65535)` for all traffic from the outbound rules of the **Isolation** security group.

3. **Identify the source of the suspicious activity**

   If malware is detected, then based on the finding type in your account, identify and stop the potentially unauthorized activity on your EC2 instance. This may require actions such as closing any open ports, changing access policies, and upgrading applications to correct vulnerabilities.

   If you are unable to identify and stop unauthorized activity on your potentially compromised 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 Networking sections in Best practices for Amazon EC2
   - Amazon EC2 security groups for Linux instances and Amazon EC2 security groups for Windows instances
   - Security in Amazon EC2
   - Tips for securing your EC2 instances (Linux).
   - AWS security best practices
   - Infrastructure Domain Incidents on AWS

4. **Browse AWS re:Post**

   Browse AWS re:Post at https://forums.aws.amazon.com/index.jspa for further assistance.

5. **Submit a technical support request**

   If you are a premium support package subscriber, you can submit a technical support request.

Remediating a potentially compromised S3 bucket

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

1. **Identify the potentially compromised S3 resource.**

   A GuardDuty finding for S3 will list the associated S3 bucket, its Amazon Resource Name (ARN), and its 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 role, user, or account) and identifying details will be listed in the finding. Depending on the source type, Remote IP address or source domain info will be available and can help you evaluate whether the source was authorized. If the finding involved credentials from an Amazon 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 that their credentials have been potentially compromised? For more information, see Remediating potentially 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.

4. **Determine whether the S3 bucket contains sensitive data.**

Use Amazon Macie to determine whether the S3 bucket contains sensitive data, such as personally identifiable information (PII), financial data, or credentials. If automated sensitive data discovery is enabled for your Macie account, review the S3 bucket's details to gain a better understanding of your S3 bucket's contents. If this feature is disabled for your Macie account, we recommend turning it on to expedite your assessment. Alternatively, you can create and run a sensitive data discovery job to inspect the S3 bucket's objects for sensitive data. For more information, see Discovering sensitive data with Macie.

If the access was authorized, you can ignore the finding. The https://console.aws.amazon.com/guardduty/ console allows you to set up rules to entirely suppress individual findings so that they no longer appear. For more information, see Suppression rules.

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.

Recommendations based on specific S3 bucket access needs

The following list provides recommendations based on specific Amazon S3 bucket 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.

Recommendations based on specific S3 bucket access needs 614

- 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 more information about S3 security options, see S3 Security best practices.

Remediating a potentially compromised ECS cluster

Follow these recommended steps to remediate a potentially compromised Amazon ECS cluster in your AWS environment:

1. **Identify the potentially compromised ECS cluster.**

   The GuardDuty Malware Protection finding for ECS provides the **ECS cluster details** in the finding's details panel.
2. **Evaluate the source of malware**

   Evaluate if the detected malware was in the container's image. If malware was in the image, identify all other tasks which are running using this image. For information about running tasks, see ListTasks.
3. **Isolate the potentially impacted tasks**

   Isolate the impacted tasks by denying all ingress and egress traffic to the task. A deny all traffic rule may help you stop an attack that is already underway, by severing all the connections to the task.

If the access was authorized, you can ignore the finding. The https://console.aws.amazon.com/guardduty/ console allows you to set up rules to entirely suppress individual findings so that they no longer appear. For more information, see Suppression rules.

Remediating potentially compromised AWS credentials

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

1. **Identify the potentially compromised 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 role or user) and its identifying information will be listed in the **Resource** section of the finding 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 about 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 the 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 this activity is a legitimate use of the AWS credentials, you can ignore the GuardDuty finding. The https://console.aws.amazon.com/guardduty/ console allows you to set up rules to entirely suppress individual findings so that they no longer appear. For more information, see Suppression rules.

If you can't confirm if this activity is a legitimate use, it could be the result of a compromise to the particular access key - the IAM user's sign-in credentials, 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 this issue.

Remediating a potentially compromised standalone container

1. **Isolate the potentially compromised container**

   The following steps will help you identify identify the potentially malicious container workload:

   - Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
   - On the **Findings** page, choose the corresponding finding to view the findings panel.
   - In the findings panel, under the **Resource affected** section, you can view the container's **ID** and **Name**.

   Isolate this container from other container workloads.

2. **Pause the container**

   Suspend all the processes in your container.

   For information about freezing your container, see Pause a container.

   Stop the container

   If the step above fails, and the container doesn't pause, stop the container from running. If you've enabled the Snapshots retention feature, GuardDuty will retain the snapshots of your EBS volumes that contain malware.

   For information about stopping the container, see Stop a container.

3. **Evaluate the presence of malware**

   Evaluate if malware was in the container's image.

If the access was authorized, you can ignore the finding. The https://console.aws.amazon.com/guardduty/ console allows you to set up rules to entirely suppress individual findings so that they no longer appear. The GuardDuty console allows you to set up rules to entirely suppress individual findings so that they no longer appear. For more information, see Suppression rules.

Remediating EKS Audit Log Monitoring findings

Amazon GuardDuty generates findings that indicate potential Kubernetes security issues when EKS Audit Log Monitoring is enabled for your account. For more information, see EKS Audit Log Monitoring. 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.

If any of the EKS Audit Log Monitoring finding types were generated expectantly, you can consider adding Suppression rules to prevent future alerts.

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:

- Potential configuration issues
- Remediating potentially compromised Kubernetes users
- Remediating potentially compromised Kubernetes pods
- Remediating potentially compromised Kubernetes nodes
- Remediating potentially compromised container images

> **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 more information about removing these permissions, see Security best practices for Amazon EKS in the *Amazon EKS User Guide*.

Potential 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
- Policy:Kubernetes/ExposedDashboard
- Policy:Kubernetes/AdminAccessToDefaultServiceAccount
- Policy:Kubernetes/KubeflowDashboardExposed
- Any finding that ends in **SuccessfulAnonymousAccess**

Remediating potentially 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 role or user. See the following user types and their remediation guidance.

> **Note**
>
> You can use Amazon Detective to further investigate the IAM role or user identified in the finding. While viewing the finding details in GuardDuty console, choose **Investigate in Detective**. Then select 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 potentially compromised Amazon EC2 instance.
  - 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 potentially compromised Kubernetes pods

When GuardDuty specifies details of a pod or workload resource inside the `resource.kubernetesDetails.kubernetesWorkloadDetails` section, that pod or workload resource has been potentially 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 pods compromise

If the `type` field inside the `resource.kubernetesDetails.kubernetesWorkloadDetails` section is **pods**, the finding identifies a single pods. The name field is the `name` of the pods and `namespace` field is its namespace.

For information about identifying the worker node running the pods, see Identify the offending pods and worker node.

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.

For information about identifying all the pods of the workload resource and the nodes on which they are running, see Identify the offending pods and worker nodes using workload name.

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`.

For information about identyifying all the pods using the service account and the nodes on which they are running, see Identify the offending pods and worker nodes using service account name.

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

To remediate a potentially 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.

If the worker node has been assigned an IAM role that allows Pods to gain access to other AWS resources, remove those roles from the instance to prevent further damage from the attack. Similarly, if the Pod has been assigned an IAM role, evaluate whether you can safely remove the IAM policies from the role without impacting other workloads.

Remediating potentially compromised container images

When a GuardDuty finding indicates a pod compromise, the image used to launch the pod could be potentially 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 potentially compromised container image:

1. Stop using the image immediately and remove it from your image repository.
2. Identify all pods using the potentially compromised image.

   For more information, see Identify pods with potentially vulnerable or compromised container images and worker nodes.

3. Isolate the potentially compromised pods, rotate credentials, and gather data for analysis. For more information, see Amazon EKS best practices guide.
4. Delete all pods using the potentially compromised image.

Remediating potentially 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.` finding field set to `True`.

To remediate a potentially compromised node:

1. Isolate the pod, rotate its credentials, and gather data for forensic analysis.

   For more information, see Amazon EKS best practices guide.
2. Identify the service accounts used by all of the pods running on the potentially compromised node. Review their permissions and rotate the service accounts if needed.
3. Terminate the potentially compromised node.

Remediating Runtime Monitoring findings

When you enable Runtime Monitoring for your account, Amazon GuardDuty may generate Runtime Monitoring finding types that indicate potential security issues in your AWS environment. The potential security issues indicate either a compromised Amazon EC2 instance, container workload, an Amazon EKS cluster, or a set of compromised credentials in your AWS environment. The security agent monitors runtime events from multiple resource types. To identify the potentially compromised resource, view **Resource type** in the generated finding details in the GuardDuty console. The following section describes the recommended remediation steps for each resource type.

Instance

If the **Resource type** in the finding details is **Instance**, it indicates that either an EC2 instance or an EKS node is potentially compromised.

- To remediate a compromised EKS node, see Remediating potentially compromised Kubernetes nodes.
- To remediate a compromised EC2 instance, see Remediating a potentially compromised Amazon EC2 instance.

EKSCluster

If the **Resource type** in the finding details is **EKSCluster**, it indicates that either a pod or a container inside an EKS cluster is potentially compromised.

- To remediate a compromised pod, see Remediating potentially compromised Kubernetes pods.
- To remediate a compromised container image, see Remediating potentially compromised container images.

ECSCluster

If the **Resource type** in the finding details is **ECSCluster**, it indicates that either an ECS task or a container inside an ECS task is potentially compromised.

1. **Identify the affected ECS cluster**

   The GuardDuty Runtime Monitoring finding provides the ECS cluster details in the finding's details panel or in the `resource.ecsClusterDetails` section in the finding JSON.

2. **Identify the affected ECS task**

   The GuardDuty Runtime Monitoring finding provides the ECS task details in the finding's details panel or in the `resource.ecsClusterDetails.taskDetails` section in the finding JSON.

3. **Isolate the affected task**

Isolate the impacted task by denying all ingress and egress traffic to the task. A deny all traffic rule may help stop an attack that is already underway, by severing all connections to the task.

4. **Remediate the compromised task**

   a. Identify the vulnerability that compromised the task.

   b. Implement the fix for that vulnerability and start new a replacement task.

   c. Stop the vulnerable task.

Container

If the **Resource type** in the finding details is **Container**, it indicates that a standalone container is potentially compromised.

- To remediate, see Remediating a potentially compromised standalone container.
- If the finding is generated across multiple containers using the same container image, see Remediating potentially compromised container images.
- If the container has accessed the underlying EC2 host, its associated instance credentials may have been compromised. For more information, see Remediating potentially compromised AWS credentials.
- If a potentially malicious actor has accessed the underlying EKS node or an EC2 instance, see the recommended remediation under the *EKSCluster* and *Instance* tabs.

Remediating compromised container images

When a GuardDuty finding indicates a task compromise, the image used to launch the task could be malicious or compromised. GuardDuty findings identify the container image within the `resource.ecsClusterDetails.taskDetails.containers.image` field. You can determine whether or not 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 of the tasks that are using this image.
3. Stop all of the tasks that are using the compromised image. Update their task definitions so that they stop using the compromised image.

Remediating a potentially compromised database

GuardDuty generates RDS Protection finding types that indicate potentially suspicious and anomalous login behavior in your Supported databases after you enable GuardDuty RDS Protection. Using RDS login activity, GuardDuty analyzes and profiles threats by identifying unusual patterns in login attempts.

> **Note**
>
> You can access the full information about a finding type by selecting it from the Findings table.

Follow these recommended steps to remediate a potentially compromised Amazon Aurora database in your AWS environment.

Topics

Remediating potentially compromised database with successful login events

The following recommended steps can help you remediate a potentially compromised Aurora database that exhibits unusual behavior related to successful login events.

1. **Identify the affected database and user.**

   The generated GuardDuty finding provides the name of the affected database and the corresponding user details. For more information, see Finding details.
2. **Confirm whether this behavior is expected or unexpected.**

   The following list specifies potential scenarios that may have caused GuardDuty to generate a finding:

   - A user who logs in to their database after a long time has passed.

- A user who logs in to their database on an occasional basis, for example, a financial analyst who logs in each quarter.
- A potentially suspicious actor who is involved in a successful login attempt potentially compromises the database.

3. **Begin this step if the behavior is unexpected.**

    1. **Restrict database access**

        Restrict database access for the suspected accounts and the source of this login activity. For more information, see Remediating potentially compromised credentials and Restrict network access.

    2. **Assess the impact and determine what information was accessed.**

        - If available, review the audit logs to identify the pieces of information that might have been accessed. For more information, see Monitoring events, logs, and streams in an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.
        - Determine if any sensitive or protected information was accessed or modified.

Remediating potentially compromised database with failed login events

The following recommended steps can help you remediate a potentially compromised Aurora database that exhibits unusual behavior related to failed login events.

1. **Identify the affected database and user.**

    The generated GuardDuty finding provides the name of the affected database and the corresponding user details. For more information, see Finding details.

2. **Identify the source of the failed login attempts.**

    The generated GuardDuty finding provides the **IP address** and **ASN organization** (if it was a public connection) under the **Actor** section of the finding panel.

    An Autonomous System (AS) is a group of one or more IP prefixes (lists of IP addresses accessible on a network) run by one or more network operators that maintain a single, clearly-defined routing policy. Network operators need Autonomous System Numbers (ASNs) to control routing within their networks and to exchange routing information with other internet service providers (ISPs).

3. **Confirm that this behavior is unexpected.**

   Examine if this activity represents an attempt to gain additional unauthorized access to the database as follows:

   - If the source is internal, examine if an application is misconfigured and attempting a connection repeatedly.
   - If this is an external actor, examine whether the corresponding database is public facing or is misconfigured and thus allowing potential malicious actors to brute force common user names.

4. **Begin this step if the behavior is unexpected.**

   1. **Restrict database access**

      Restrict database access for the suspected accounts and the source of this login activity. For more information, see Remediating potentially compromised credentials and Restrict network access.

   2. **Perform root-cause analysis and determine the steps that potentially led to this activity.**

      Set up an alert to get notified when an activity modifies a networking policy and creates an insecure state. For more information, see Firewall policies in AWS Network Firewall in the *AWS Network Firewall Developer Guide*.

Remediating potentially compromised credentials

A GuardDuty finding may indicate that the user credentials for an affected database have been compromised when the user identified in the finding has performed an unexpected database operation. You can identify the user in the **RDS DB user details** section within the finding panel in the console, or within the `resource.rdsDbUserDetails` of the findings JSON. These user details include user name, application used, database accessed, SSL version, and authentication method.

- To revoke access or rotate passwords for specific users that are involved in the finding, see Security with Amazon Aurora MySQL, or Security with Amazon Aurora PostgreSQL in the *Amazon Aurora User Guide*.
- Use AWS Secrets Manager to securely store and automatically rotate the secrets for Amazon Relational Database Service(RDS) databases. For more information, see AWS Secrets Manager tutorials in the *AWS Secrets Manager User Guide*.

- Use IAM database authentication to manage database users' access without the need for passwords. For more information, see IAM database authentication in the *Amazon Aurora User Guide*.

  For more information, see Security best practices for Amazon Relational Database Service in the *Amazon RDS User Guide*.

Restrict network access

A GuardDuty finding may indicate that a database is accessible beyond your applications, or Virtual Private Cloud (VPC). If the remote IP address in the finding is an unexpected connection source, audit the security groups. A list of security groups attached to the database is available under **Security groups** in the https://console.aws.amazon.com/rds/ console, or in the `resource.rdsDbInstanceDetails.dbSecurityGroups` of the findings JSON. For more information on configuring security groups, see Controlling access with security groups in the *Amazon RDS User Guide*.

If you're using a firewall, restrict network access to the database by reconfiguring the Network Access Control Lists (NACLs). For more information, see Firewalls in AWS Network Firewall in the *AWS Network Firewall Developer Guide*.

Remediating a potentially compromised Lambda function

When GuardDuty generates a Lambda Protection finding and the activity is unexpected, your Lambda function may be compromised. We recommend completing the following steps to remediate a compromised Lambda function.

To remediate Lambda Protection findings

1. **Identify the potentially compromised Lambda function version**.

   A GuardDuty finding for Lambda Protection provides the name, Amazon Resource Name (ARN), function version, and revision ID associated with the Lambda function listed in the finding details.

2. **Identify the source of the potentially suspicious activity**.

   a. Review the code associated with the Lambda function version involved in the finding.

   b. Review the imported libraries and layers of the Lambda function version involved in the finding.

   c. If you have enabled Scanning AWS Lambda functions with Amazon Inspector, review the Amazon Inspector findings associated with the Lambda function involved in the finding.

   d. Review the AWS CloudTrail logs to identify the principal that caused the function update and ensure that the activity was authorized or expected.

3. **Remediate the potentially compromised Lambda function**.

   a. Disable the execution triggers of the Lambda function involved in the finding. For more information, see DeleteFunctionEventInvokeConfig.

   b. Review the Lambda code and update the libraries imports and Lambda function layers to remove the potentially suspicious libraries and layers.

   c. Mitigate Amazon Inspector findings related to the Lambda function involved in the finding.

Managing multiple accounts in Amazon GuardDuty

When your AWS environment has multiple accounts, you can manage them by designating one AWS account as your administrator account. You can then associate other AWS accounts with this administrator account as its member accounts. This designated GuardDuty administrator account can configure the protection plans Within GuardDuty there are two ways to associate accounts with a administrator account – create an organization by using AWS Organizations and both administrator account and one or more member accounts belong to this organization, or send an invitation to an AWS account 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 AWS 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.

Managing multiple accounts by invitation

If the accounts that you want to associate are not a part of your organization, you can specify an 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.

Understanding the relationship between GuardDuty administrator account 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 account and member accounts.

In this table:

- **Self –** An account can perform the listed action only for their own account.
- **Any –** An account can perform the listed action for any associated account.
- **All –** An account can perform the listed action and it applies to all the associated accounts. Usually, the account taking this action is a designated GuardDuty administrator account

Table cells with dash (—) indicate that the account can't perform the listed action.

Action	Through AWS Organizations		By invitation	
	Delegated GuardDuty administrator account	Associated member account	Delegated GuardDuty administrator account	Associated member account
Enable GuardDuty	Any	–	Self	Self
Enable GuardDuty automatically for the entire organization (`ALL`, `NEW`, `NONE`)	All	–	–	–
View all Organizations member accounts regardless of GuardDuty status	Any	–	–	–
Generate sample findings	Self	Self	Self	Self
View all GuardDuty findings	Any	Self	Any	Self
Archive GuardDuty findings	Any	–	Any	–

Apply suppression rules	All	–	All	–
Create trusted IP list or threat lists	All	–	All	–
Update trusted IP list or threat lists	All	–	All	–
Delete trusted IP list or threat lists	All	–	All	–
Set EventBridge notification frequency	All	–	All	Self
Set Amazon S3 location for exporting findings	All	–	All	Self
Enable one or more optional protection plans for the entire organization (`ALL`, `NEW`, `NONE`)	All	–	–	–
Enable any GuardDuty protection plan for individual accounts	Any	–	Any	Self
Disassociate a member account	Any	–	Any	–

Disassociate from an administrator account account	–	Self[#]	–	Self
Delete a disassociated member account	Any	–	Any	–
Suspend GuardDuty	Any[*]	–	Any[*]	–
Disable GuardDuty	Any[*]	–	Any[*]	–

- # Indicates that the account can take this action only if the delegated GuardDuty administrator account has not set up the auto-enable preference to `ALL` the organization members.
- * Indicates that this action must be taken for all associated accounts before being taken for this account. After you disassociate these accounts, you must delete them. For more information about performing these tasks in your organization, see Maintaining your organization within GuardDuty.

Managing GuardDuty accounts with AWS Organizations

When you use GuardDuty with an AWS organization, the management account of that organization can designate any account within the organization as the delegated GuardDuty administrator account. For this administrator account, GuardDuty gets enabled automatically only in the designated AWS Region. This account also has the permission to enable and manage GuardDuty for all of the accounts in the organization within that Region. The administrator account can view the members of and add members to this AWS organization.

If you have already set up a GuardDuty administrator account 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 delegated GuardDuty administrator account for your organization. If a delegated GuardDuty administrator account previously added members by invitation that are not part of the same organization, their **Type** remains **By Invitation**. In both the cases, the previously added accounts are member accounts that are associated with the organization's delegated GuardDuty administrator account.

You can continue to add accounts as members even if they are outside of your organization. For more information, see Adding and managing accounts by invitations or Designating a delegated GuardDuty administrator account and managing members by using the GuardDuty console.

Contents

Considerations and recommendations when designating a delegated GuardDuty administrator account

The following considerations and recommendations can help you understand how a delegated GuardDuty administrator account operates in GuardDuty:

A delegated GuardDuty administrator account can manage a maximum of 50,000 members.

> There is a limit of 50,000 member accounts per delegated GuardDuty administrator account. This includes member accounts that are added through AWS Organizations or those who accepted the GuardDuty administrator account's invitation to join their organization. However, there could be more than 50,000 accounts in your AWS organization.
>
> If you exceed the 50,000 member accounts limit, you will receive a notification from CloudWatch, AWS Health Dashboard, and an email to the designated delegated GuardDuty administrator account.

A delegated GuardDuty administrator account is Regional.

Unlike AWS Organizations, GuardDuty is a Regional service. The delegated GuardDuty administrator accounts and their member accounts must be added through AWS Organizations in each desired Region where you have GuardDuty enabled. If the organization management account designates a delegated GuardDuty administrator account in only US East (N. Virginia), then delegated GuardDuty administrator account will only manage member accounts added to the organization in that Region. For more information about feature parity in Regions where GuardDuty is available, see Regions and endpoints.

Special cases for opt-in Regions

- When a delegated GuardDuty administrator account opts out of an opt-in Region, even if your organization has the GuardDuty auto-enable configuration set to either new member accounts only (`NEW`) or all member accounts (`ALL`), GuardDuty cannot be enabled for any member account in the organization that currently has GuardDuty disabled. For information about the configuration of your member accounts, open **Accounts** in the GuardDuty console navigation pane or use the ListMembers API.
- When working with the GuardDuty auto-enable configuration set to `NEW`, ensure that the following sequence is met:

  1. The member accounts opt-in to an opt-in Region.
  2. Add the member accounts to your organization in AWS Organizations.

  If you change the order of these steps, the GuardDuty auto-enable setting with `NEW` **will not** work in the specific opt-in Region because the member account is no longer new to the organization. GuardDuty provides two alternate solutions:

  - Set the GuardDuty auto-enable configuration to `ALL`, that includes new and existing members accounts. In this case, the order of these steps is not relevant.
  - If a member account is already a part of your organization, manage the GuardDuty configuration for this account individually in the specific opt-in Region by using the GuardDuty console or the API.

Recommended for an AWS organization to have the same delegated GuardDuty administrator account across all the AWS Regions.

We recommend you designate the same delegated GuardDuty administrator account to your organization across all the AWS Regions where you have enabled GuardDuty. If you designate an account as a delegated GuardDuty administrator account in one Region, it is recommended that you use the same account as delegated GuardDuty administrator account in all other Regions.

You can designate a new delegated GuardDuty administrator account at any point in time. For more information about removing the existing delegated GuardDuty administrator account, see Changing the delegated GuardDuty administrator account.

Not recommended to set your organization's management account as the delegated GuardDuty administrator account.

Your organization's management account can be the delegated GuardDuty administrator account. However, the AWS security best practices follow the principle of least privilege and doesn't recommend this configuration.

Changing a delegated GuardDuty administrator account does not disable GuardDuty for member accounts.

If you remove a delegated GuardDuty administrator account, GuardDuty removes all the member accounts associated with this delegated GuardDuty administrator account. GuardDuty still remains enabled for all these member accounts.

Permissions required to designate a delegated GuardDuty administrator account

When delegating a delegated GuardDuty administrator account you must have permissions to enable GuardDuty as well as certain AWS Organizations API actions. You can add the following statement at 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:ListAccounts",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListAccounts"
    ],
    "Resource": "*"
}
```

Additionally, if you wish to designate your AWS Organizations management account as the GuardDuty delegated GuardDuty administrator account that entity will need `CreateServiceLinkedRole` permissions to initialize GuardDuty. To do this, add the following statement to the IAM policy and replace *111122223333* with the AWS account ID of your organization's management account:

```
{
 "Sid": "PermissionsToEnableGuardDuty"
 "Effect": "Allow",
 "Action": [
  "iam:CreateServiceLinkedRole"
 ],
 "Resource": "arn:aws:iam::111122223333:role/aws-service-role/guardduty.amazonaws.com/
AWSServiceRoleForAmazonGuardDuty",
 "Condition": {
  "StringLike": {
   "iam:AWSServiceName": "guardduty.amazonaws.com"
  }
 }
}
```

Designating a delegated GuardDuty administrator account and managing members by using the GuardDuty console

Contents

Step 1 – Designate a delegated GuardDuty administrator account for your organization

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   To log in, use the management account credentials for your AWS Organizations organization.

2. If you have already enabled GuardDuty for the management account, skip this step and follow the next step.

   If you haven't enabled GuardDuty yet, select **Get Started**, and then designate a delegated GuardDuty administrator account on the **Welcome to GuardDuty** page.

   > **Note**
   >
   > The management account must have the GuardDuty service-linked role (SLR) so that the delegated GuardDuty administrator account can enable and manage GuardDuty in that account. Once you enable GuardDuty in a Region for the management account, this SLR gets created automatically.

3. Do this step after you have enabled GuardDuty for the management account. In the navigation pane of the GuardDuty console, choose **Settings**. On the **Settings** page, enter the 12-digit AWS account ID of the account that you want to designate as the delegated GuardDuty administrator account for the organization.

   Make sure to enable GuardDuty for your newly designated delegated GuardDuty administrator account, otherwise it won't be able to take any action.

4. Choose **Delegate**.

5. (Recommended) Repeat the previous step to designate the delegated GuardDuty administrator account in each AWS Region where you have GuardDuty enabled.

Step 2 – Configuring auto-enable preferences for your organization

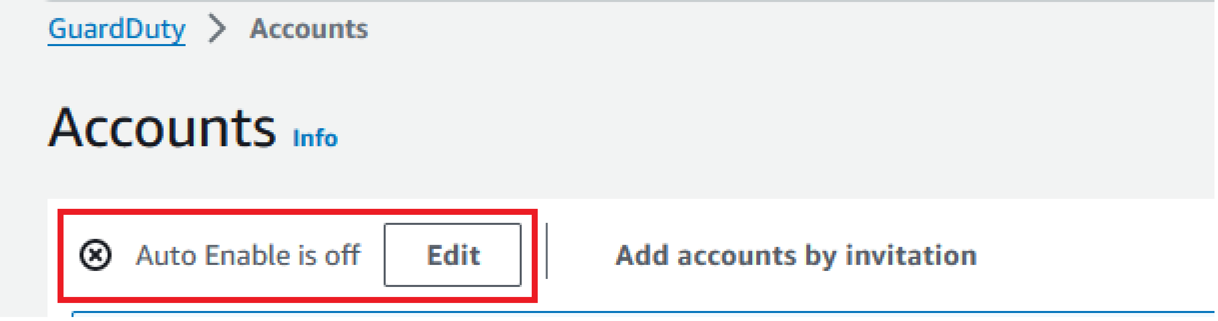
1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   To sign in, use the GuardDuty administrator account credentials.

2. In the navigation pane, choose **Accounts**.

The **Accounts** page provides configuration options to the GuardDuty administrator account to **Auto-enable** GuardDuty and the optional protection plans on behalf of the member accounts that belong to the organization.

3. To update the existing auto-enable settings, choose **Edit**.



This support is available to configure GuardDuty and all of the supported optional protection plans in your AWS Region. You can select one of the following configuration options for GuardDuty on behalf of your member accounts:

- **Enable for all accounts (ALL)** – Select to enable the corresponding option for all the accounts in an organization. This includes new accounts that join the organization and those accounts that may have been suspended or removed from the organization. This also includes the delegated GuardDuty administrator account.

> **Note**
>
> It may take up to 24 hours to update the configuration for all member accounts.

- **Auto-enable for new accounts (NEW)** – Select to enable GuardDuty or the optional protection plans for only new member accounts automatically when they join your organization.
- **Do not enable (NONE)** – Select to prevent enabling the corresponding option for new accounts in your organization. In this case, the GuardDuty administrator account will manage each account individually.

  When you update the auto-enable setting from `ALL` or `NEW` to `NONE`, this action doesn't disable the corresponding option for your existing accounts. This configuration will apply to the new accounts that join the organization. After you update the auto-enable settings, no new account will have the corresponding option as enabled.

> **Note**
>
> When a delegated GuardDuty administrator account opts out of an opt-in Region, even if your organization has the GuardDuty auto-enable configuration set to either new member accounts only (`NEW`) or all member accounts (`ALL`), GuardDuty cannot be enabled for any member account in the organization that currently has GuardDuty disabled. For information about the configuration of your member accounts, open **Accounts** in the GuardDuty console navigation pane or use the ListMembers API.

4. Choose **Save changes**.
5. (Optional) if you want to use the same preferences in each Region, update your preferences in each of the supported Regions separately.

   Some of the optional protection plans may not be available in all the AWS Regions where GuardDuty is available. For more information, see Regions and endpoints.

Step 3 – Add accounts as members to your organization

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   To log in, use the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **Accounts**.

   The accounts table displays all of the accounts that are added either **Via Organizations** (AWS Organizations) or **By Invitation**. If a member account is not associated with the organization's GuardDuty administrator account, the **Status** of this member account is **Not a member**.
3. Select one or multiple account IDs that you want to add as members. These account IDs must have the **Type** as **Via Organizations**.

   Accounts that are added through invitation are not a part of your organization. You can manage such accounts individually. For more information, see Managing accounts by invitation.
4. Choose the **Actions** dropdown and then choose **Add member**. After you add this account as a member, the auto-enable GuardDuty configuration will apply. Based on the settings in the section called "Step 1 – Designate a delegated GuardDuty administrator account for your organization", the GuardDuty configuration of these accounts may change.

5. You can select the down arrow of the **Status** column to sort the accounts by the **Not a member** status and then choose each account that doesn't have GuardDuty enabled in the current Region.

   If none of the accounts listed in the accounts table have been added as a member yet, you can enable GuardDuty in the current Region for all organization accounts. Choose **Enable** in the banner at the top of the page. This action automatically turns on the **Auto-enable** GuardDuty configuration so that GuardDuty gets enabled for any new account that joins the organization.
6. Choose **Confirm** to add the accounts as members. This action also enables GuardDuty for all of the selected accounts. The **Status** for the accounts will change to **Enabled**.
7. (Recommended) Repeat these steps in each AWS Region. This ensures that the delegated GuardDuty administrator account can manage findings and other configurations for member accounts in all the Regions where you have GuardDuty enabled.

   The auto-enable feature enables GuardDuty for all future members of your organization. This allows your delegated GuardDuty administrator account to manage any new members that are created within or get added to the organization. When the number of member accounts reaches the limit of 50,000, the Auto-enable feature is automatically turned off. If you remove a member account and the total number of members decreases to fewer than 50,000, the Auto-enable feature turns back on.

(Optional) step 4 – Configure protection plans for individual accounts

You can configure protection plans for individual accounts through the **Accounts** page.

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Use the delegated GuardDuty administrator account credentials.
2. In the navigation pane, choose **Accounts**.
3. Select one or more accounts for which you want to configure a protection plan. Repeat the following steps for each protection plan that you want to configure:

   a. Choose **Edit Protection Plans**.

   b. From the list of protection plans, choose one protection plan that you want to configure.

   c. Choose one of the actions that you want to perform for this protection plan, and then choose **Confirm**.

d. For the selected account, the column corresponding to the configured protection plan will show the updated configuration as **Enabled** or **Not enabled**.

Designating a GuardDuty delegated GuardDuty administrator account and managing members by using the API

Contents

Step 1 – Designate a delegated GuardDuty administrator account for your AWS organization

1. Run enableOrganizationAdminAccount using the credentials of the AWS account of the organization's management account.

   - Alternatively, you can use AWS Command Line Interface to do this. The following AWS CLI command designates a delegated GuardDuty administrator account for your current Region only. Run the following AWS CLI command and make sure to replace *111111111111* with the AWS account ID of the account you want to designate as a delegated GuardDuty administrator account:

     ```
     aws guardduty enable-organization-admin-account --admin-account-id 111111111111
     ```

     To designate the delegated GuardDuty administrator account for other Regions, specify the Region in the AWS CLI command. The following example demonstrates how to enable a delegated GuardDuty administrator account in US West (Oregon). Make sure to replace *us-west-2* with the Region for which you want to assign the GuardDuty delegated GuardDuty administrator account.

     ```
     aws guardduty enable-organization-admin-account --admin-account-id 111111111111
      --region us-west-2
     ```

     For information about the AWS Regions where GuardDuty is available, see Regions and endpoints.

If GuardDuty is not enabled for your delegated GuardDuty administrator account, it won't be able to take any action. If not already done so, make sure to enable GuardDuty for the newly designated delegated GuardDuty administrator account.

2. (Recommended) repeat the previous step to designate the delegated GuardDuty administrator account in each AWS Region where you have GuardDuty enabled.

Step 2 - Configuring auto-enable preferences for the organization

1. 1. Run UpdateOrganizationConfiguration by using the credentials of the delegated GuardDuty administrator account, to automatically configure GuardDuty and optional protection plans in that Region for your organization

   To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   > **Note**
   >
   > For information about the various auto-enable configurations, see autoEnableOrganizationMembers.

   2. To set auto-enable preferences for any of the supported optional protection plans in your Region, follow the steps provided in the corresponding documentation sections of each protection plan.

   3. You can validate the preferences for your organization in the current Region. Run describeOrganizationConfiguration. Make sure to specify the detector ID of the delegated GuardDuty administrator account.

   > **Note**
   >
   > It may take up to 24 hours to update the configuration for all the member accounts.

- 1. Alternatively, run the following AWS CLI command to set the preferences to automatically enable or disable GuardDuty in that Region for new accounts (`NEW`) that join the organization, all the accounts (`ALL`), or none of the accounts (`NONE`) in the organization. For more information, see autoEnableOrganizationMembers. Based on your preference, you may need to replace NEW with ALL or NONE. If you configure the protection plan with ALL, the protection plan will also be enabled for the delegated GuardDuty administrator account. Make sure to specify the detector ID of the delegated GuardDuty administrator account that manages the organization configuration.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty update-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0 --auto-enable-organization-members=NEW
```

2. You can validate the preferences for your organization in the current Region. Run the following AWS CLI command by using the detector ID of the delegated GuardDuty administrator account.

```
aws guardduty describe-organization-configuration --detector-
id 12abc34d567e8fa901bc2d34e56789f0
```

2. (Recommended) repeat the previous steps in each Region by using the delegated GuardDuty administrator account detector ID.

> **Note**
>
> When a delegated GuardDuty administrator account opts out of an opt-in Region, even if your organization has the GuardDuty auto-enable configuration set to either new member accounts only (NEW) or all member accounts (ALL), GuardDuty cannot be enabled for any member account in the organization that currently has GuardDuty disabled. For information about the configuration of your member accounts, open **Accounts** in the GuardDuty console navigation pane or use the ListMembers API.

Step 3 – Add accounts as members to your organization

- Run CreateMembers by using the credentials of the delegated GuardDuty administrator account designated in the previous step.

  You must specify the regional detector ID of the delegated GuardDuty administrator account and the account details (AWS account IDs and corresponding email addresses) of the accounts that you want to add as GuardDuty members. You can create one or more members with this API operation.

When you run CreateMembers in your organization, the auto-enable preferences for new members will apply as new member accounts join your organization. When you run CreateMembers with an existing member account, the organization configuration will also apply to the existing members. This might change the current configuration of the existing member accounts.

Run ListAccounts in the *AWS Organizations API Reference*, to view all the accounts in the AWS organization.

> ⚠ **Important**
>
> When you add an account as a GuardDuty member, it will automatically have GuardDuty enabled in that Region. There is an exception to the organization management account. Before the management account account gets added as a GuardDuty member, it must have GuardDuty enabled.

- Alternatively, you can use AWS Command Line Interface. Run the following AWS CLI command and make sure to use your own valid detector ID, AWS account ID, and the email address associated with the account ID.

  To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

  ```
  aws guardduty create-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
  account-details AccountId=111122223333,Email=guardduty-member-name@amazon.com
  ```

  You can view a list of all organization members by running the following AWS CLI command:

  ```
  aws organizations list-accounts
  ```

After you add this account as a member, the auto-enable GuardDuty configuration will apply.

Maintaining your organization within GuardDuty

As a delegated GuardDuty administrator account, you are responsible for maintaining the configuration of GuardDuty and its optional protection plans for all the accounts in your organization in each supported AWS Region. The following sections provide the options about maintaining the configuration status of GuardDuty or any of its optional protection plans:

To maintain the configuration status of your entire organization in each Region

- **Set auto-enable preferences for the entire organization by using GuardDuty console** – You can enable GuardDuty automatically for either all (`ALL`) the members in the organization or new (`NEW`) members joining the organization, or choose not to (`NONE`) auto-enable it any of the members in the organization.

  You can also configure the same or different settings for any of the protection plans within GuardDuty.

  It might take up to 24 hours to update the configuration for all member accounts in the organization.

- **Update auto-enable preferences by using API** – Run UpdateOrganizationConfiguration to automatically configure GuardDuty and its optional protection plans for the organization. When you run CreateMembers to add new member accounts in your organization, the configured settings will apply automatically. When you run CreateMembers with an existing member account, the organization configuration will also apply to the existing members. This might change the current configuration of the existing member accounts.

  To view all the accounts in your organization, run ListAccounts in the *AWS Organizations API Reference*.

To maintain the configuration status for member accounts individually in each Region

- To view all the accounts in your organization, run ListAccounts in the *AWS Organizations API Reference*.
- When you want selective member accounts to have a different configuration status, run UpdateMemberDetectors for each member account individually.

  You can use GuardDuty console to perform the same task by navigating to the **Accounts** page in the GuardDuty console.

For information about enabling protection plans for individual accounts by using either console or API, see the configuring page for the corresponding protection plan.

Changing the delegated GuardDuty administrator account

You can change the delegated GuardDuty administrator account for your organization in each Region and then delegate a new administrator in each Region. To maintain a security posture for your organization's member accounts in a Region, you must have a delegated GuardDuty administrator account in that Region.

Removing existing delegated GuardDuty administrator account

Step 1 - To remove existing delegated GuardDuty administrator account in each Region

1. As the existing delegated GuardDuty administrator account, list all the member accounts associated with your administrator account. Run ListMembers with `OnlyAssociated=false`.
2. If the auto-enable preference for GuardDuty or any of the optional protection plans is set to `ALL`, then run UpdateOrganizationConfiguration to update the organization configuration to either `NEW` or `NONE`. This action will prevent an error when you disassociate all the member accounts in the next step.
3. Run DisassociateMembers to disassociate all the member accounts that are associated with the administrator account.
4. Run DeleteMembers to delete the associations between the administrator account and member accounts.
5. As the organization management account, run DisableOrganizationAdminAccount to remove the existing delegated GuardDuty administrator account.
6. Repeat these steps in each AWS Region where you have this delegated GuardDuty administrator account.

Step 2 - To de-register existing delegated GuardDuty administrator account in AWS Organizations (One-time global action)

- Run DeregisterDelegatedAdministrator in the *AWS Organizations API Reference*, to de-register the existing delegated GuardDuty administrator account in AWS Organizations.

  Alternatively, you can run the following AWS CLI command:

```
aws organizations deregister-delegated-administrator --account-id 111122223333 --
service-principal guardduty.amazonaws.com
```

Make sure to replace *111122223333* with the existing delegated GuardDuty administrator account.

After you de-register the old delegated GuardDuty administrator account, you can add it as a member account to the new delegated GuardDuty administrator account.

Designating a new delegated GuardDuty administrator account in each Region

1. Designate a new delegated GuardDuty administrator account in each Region by using one of the following access methods:

   - **Using GuardDuty console** – Step 1 – Designate a delegated GuardDuty administrator account for your organization.
   - **Using GuardDuty API** – Step 1 – Designate a delegated GuardDuty administrator account for your AWS organization.

2. Run DescribeOrganizationConfiguration to view the current auto-enable configuration for your organization.

   > ⚠ **Important**
   >
   > Before you add any members to the new delegated GuardDuty administrator account, you must verify the auto-enable configuration for your organization. This configuration is specific to the new delegated GuardDuty administrator account and the selected Region, and doesn't relate to AWS Organizations. When you add (a new or an existing) organization member account under the new delegated GuardDuty administrator account, the auto-enable configuration of the new delegated GuardDuty administrator account will apply at the time of enabling GuardDuty or any of its optional protection plans.

   To change this organization configuration for the new delegated GuardDuty administrator account, use one of the following access methods:

- **Using GuardDuty console** – Step 2 – Configuring auto-enable preferences for your organization.
- **Using GuardDuty API** – Step 2 - Configuring auto-enable preferences for the organization.

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 an 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 account and member account at the same time.

When you accept an invitation from one account, you can't accept an invitation from another account. To accept an invitation from another account, you will first need to disassociate your account from the existing administrator account. Alternatively, the administrator account can also disassociate and remove your account from their organization.

Accounts associated by invitation have the same overall administrator account-to-member relationship as accounts associated by AWS Organizations, as described in Understanding the relationship between GuardDuty administrator account and member accounts. 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.

Adding and managing accounts by invitations

Choose one of the access methods to add and invite accounts to become GuardDuty member accounts as a GuardDuty administrator account.

Console

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 pane.
4. On the **Add member accounts** page, under **Enter account details**, enter the AWS account ID and email address associated with the account that you want to add.
5. To add another row to enter account details one at a time, choose **Add another account**. You can also choose **Upload .csv file with account details** to add accounts in bulk.

> **Important**
>
> The first line of your csv file must contain the header, as depicted in the following example – `Account ID,Email`. Each subsequent line must contain a single valid AWS account ID and its associated email address. The format of a row is valid if it contains only one AWS account ID and the associated email address separated by a comma.
>
> ```
> Account ID,Email
>                                 555555555555,user@example.com
> ```

6. After you have added all the accounts' details, choose **Next**. You can view the newly-added accounts in the Accounts table. The **Status** of these accounts will be **Invite not sent**. For information about sending an invite to one or more added accounts, see Step 2 - Invite an account.

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. Select one or more accounts that you want to invite to Amazon GuardDuty.
4. Choose **Actions** dropdown menu and then choose **Invite**.
5. In the **Invitation to GuardDuty** dialog box, enter an (optional) invitation message.

If the invited account does not have access to email, select the checkbox **Also send an email notification to the root user on the invitee's AWS account and generate an alert in the invitee's AWS Health Dashboard**.

6. Choose **Send invitation**. If the invitees have access to the specified email address they can view the invite by opening the GuardDuty console at https://console.aws.amazon.com/guardduty/.
7. When an invitee accepts the invite, the value in the **Status** column changes to **Invited**. For information about accepting an invite, see Step 3 - Accept an invitation.

Step 3 - Accept an invitation

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   > ⚠ **Important**
   >
   > You must enable GuardDuty before you can view or accept a membership invitation.

2. Do the following only if you haven't enabled GuardDuty yet; otherwise, you can skip this step and continue with the next step.

   If you haven't yet enabled GuardDuty, choose **Get Started** on the Amazon GuardDuty page.

   On the **Welcome to GuardDuty** page, choose **Enable GuardDuty**.
3. After you enable GuardDuty for your account, use the following steps to accept the membership invitation:
   a. In the navigation pane, choose **Settings**.
   b. Choose **Accounts**.
   c. On the **Accounts**, ensure to verify the owner of the account from which you accept the invitation. Turn on **Accept** to accept the membership invite.
4. After you accept the invite, your account becomes a GuardDuty member account. The account whose owner sent the invitation becomes the GuardDuty administrator account. The administrator account will know that you have accepted the invitation. The **Accounts** table in their GuardDuty account will get updated. The value in the **Status** column corresponding to your member account ID will change to **Enabled**. The administrator account owner can now view and manage GuardDuty and protection plan configurations

on behalf of your account. The administrator account can also view and manage GuardDuty findings generated for your member account.

API/CLI

You can designate a GuardDuty administrator account, and create or add GuardDuty member accounts by invitation through the API operations. Run the following GuardDuty API operations in order to designate administrator account 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.

Creating or adding member accounts

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 account 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.

   To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

   ```
   aws guardduty create-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
   account-details AccountId=111122223333,Email=guardduty-member@organization.com
   ```

2. Run InviteMembers by using the credentials of the AWS account that has GuardDuty enabled. This is the account that you want to be the administrator account 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 by using the `message` request parameter.

You can also use AWS Command Line Interface to designate member accounts by running the following command. Make sure to use your own valid detector ID and valid account IDs for the accounts you want to invite.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty invite-members --detector-id 12abc34d567e8fa901bc2d34e56789f0 --
account-ids 111122223333
```

Accepting invitations

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 AcceptAdministratorInvitation 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 an invitation ID.

To find the `detectorId` for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

```
aws guardduty accept-invitation --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --administrator-id 444455556666 --invitation-
id 84b097800250d17d1872b34c4daadcf5
```

Consolidating GuardDuty administrator accounts under a single organization delegated GuardDuty administrator account

GuardDuty recommends using association through AWS Organizations to manage member accounts under a delegated GuardDuty administrator account. You can use the example process outlined below to consolidate administrator account and member associated by invitation in an organization under a single GuardDuty delegated GuardDuty administrator account.

> **Note**
>
> Accounts that are already being managed by a delegated GuardDuty administrator account, or active member accounts that are associated with delegated GuardDuty administrator account can't be added to a different delegated GuardDuty administrator account. Each organization can have only one delegated GuardDuty administrator account per Region, and each member account can have only one delegated GuardDuty administrator account.

Choose one of the access methods to consolidate GuardDuty administrator accounts under a single delegated GuardDuty administrator account.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

To log in, use the credentials of the management account of the organization.

2. All the accounts for which you want to manage GuardDuty must be a part of your organization. For information about adding an account to your organization, see Inviting an AWS account to join your organization.
3. Make sure all the member accounts are associated with the account that you want to designate as the single delegated GuardDuty administrator account. Disassociate any member account that is still associated with the pre-existing administrator accounts.

   The following steps will help you disassociate member accounts from the pre-existing administrator account:

   a. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
   b. To log in, use the credentials of the pre-existing administrator account.
   c. In the navigation pane, choose **Accounts**.
   d. On the **Accounts** page, select one or more accounts that you want to disassociate from the administrator account.
   e. Choose **Actions** and then choose **Disassociate account**.
   f. Choose **Confirm** to finalize the step.
4. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   To log in, use the management account credentials.
5. In the navigation pane, choose **Settings**. On the **Settings** page, designate the delegated GuardDuty administrator account for the organization.
6. Log in to the designated delegated GuardDuty administrator account.
7. Add members from the organization. For more information, see Managing GuardDuty accounts with AWS Organizations.

API/CLI

1. All the accounts for which you want to manage GuardDuty must be a part of your organization. For information about adding an account to your organization, see Inviting an AWS account to join your organization.
2. Make sure all the member accounts are associated with the account that you want to designate as the single delegated GuardDuty administrator account.

   a. Run DisassociateMembers to disassociate any member account that is still associated with the pre-existing administrator accounts.

   b. Alternatively, you can use AWS Command Line Interface to run the following command and replace *777777777777* with the detector ID of the pre-existing administrator account from which you want to disassociate the member account. Replace *666666666666* with the AWS account ID of the member account that you want to disassociate.

   ```
   aws guardduty disassociate-members --detector-id 777777777777 --account-
   ids 666666666666
   ```

3. Run EnableOrganizationAdminAccount to delegate an AWS account as the delegated GuardDuty administrator account.

   Alternatively, you can use AWS Command Line Interface to run the following command to delegate a delegated GuardDuty administrator account:

   ```
   aws guardduty enable-organization-admin-account --admin-account-id 777777777777
   ```

4. Add members from the organization. For more information, see Create or add member member accounts using API.

⚠ **Important**

To maximize the effectiveness of GuardDuty, a regional service, we recommend that you designate your delegated GuardDuty administrator account and add all your member accounts in every Region.

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 at 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, see to the instructions in the shared link.

The `enableguardduty.py` script enables GuardDuty, sends invitations from the administrator account, and accepts invitations in all member accounts. The result is a administrator account 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 – AWS 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 AWS managed policy: AmazonGuardDutyFullAccess policy to enable GuardDuty.
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.

```
111122223333,guardduty-member@organization.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 444455556666 --assume_role
 roleName accountID.csv
```

Estimating GuardDuty cost

You can use the GuardDuty console or API operations to estimate the daily average usage costs for GuardDuty. During the 30-day free trial period, the cost estimation projects what your estimated costs will be after the trial period. If you are operating in a multi-account environment, your GuardDuty administrator account can monitor cost metrics for all of the 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 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, CloudTrail data events, or DNS logs.
- **Features –** Lists the estimated cost on the specified data source for the following GuardDuty features: CloudTrail data events for S3, EKS Audit Log Monitoring, EBS volume data, RDS login activity, EKS Runtime Monitoring, Fargate Runtime Monitoring, EC2 Runtime Monitoring, or Lambda Network Activity Monitoring.
- **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.

Understanding how GuardDuty calculates usage costs

The estimates displayed in the GuardDuty console may differ slightly than those in your AWS Billing and Cost Management console. The following list explains how GuardDuty estimates usage costs:

- The GuardDuty usage estimate is for the current Region only.
- The GuardDuty usage cost is based on the last 30 days of usage.

- The trial usage cost estimate includes the estimate for foundational data sources and features that are currently in the trial period. Each feature and data source within GuardDuty has its own trial period but it may overlap with the trial period of GuardDuty or another feature that was enabled at the same time.
- 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.
- The sum of the usage cost for each AWS account in your organization may not always be the same as the last 30-day estimated cost for the selected data source. The pricing tier may change as GuardDuty processes more events or data. For more information, see Pricing Tiers in the *AWS Billing User Guide*.

Runtime Monitoring – How VPC flow logs from EC2 instances impact usage cost

When you manage the security agent (either manually or through GuardDuty) in EKS Runtime Monitoring or Runtime Monitoring for EC2 instances, and GuardDuty is presently deployed on an Amazon EC2 instance and receives the Collected runtime event types from this instance, GuardDuty will not charge your AWS account for the analysis of VPC flow logs from this Amazon EC2 instance. This helps GuardDuty avoid double usage cost in the account.

How GuardDuty estimates usage cost for CloudTrail events

When you enable GuardDuty, it automatically starts consuming AWS CloudTrail event logs recorded for your account in the selected AWS Region. GuardDuty replicates Global service events logs and then processes these events independently in each Region where you have GuardDuty enabled. This helps GuardDuty maintain user and role profiles in each Region to identify anomalies.

Your CloudTrail configuration does not impact GuardDuty usage cost or the way GuardDuty processes your event logs. Your GuardDuty usage cost is affected by your usage of AWS APIs which log to CloudTrail. For more information, see AWS CloudTrail event logs.

Reviewing GuardDuty usage statistics

Choose your preferred access method to review the usage statistics for your GuardDuty account. If you're a GuardDuty administrator account, the following methods will help you review the usage statistics for all the members.

Console

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.

   Make sure to use the GuardDuty administrator account account.
2. In the navigation pane, choose **Usage**.
3. On the **Usage** page, a GuardDuty administrator account with member accounts can view the **Estimated organization cost** for the last 30 days. This is an estimated total usage cost for your organization.
4. GuardDuty administrator accounts with members can either view the usage cost breakdown by data source or by accounts. Individual or standalone accounts can view the breakdown by data source.

   If you have member accounts, you can view the statistics for an individual account by selecting that account in the Accounts table.

   Under the **By data sources** tab, when you select a data source that has a usage cost associated with it, the corresponding sum of the cost breakdown at the accounts level may not always be the same.

API/CLI

Run the GetUsageStatistics API operation using the credentials of GuardDuty administrator account account. Provide the following information to run the command:

- (Required) provide the Regional GuardDuty detector ID of the account for which you want to retrieve the statistics.
- (Required) provide one of the types of statistics to retrieve: `SUM_BY_ACCOUNT | SUM_BY_DATA_SOURCE | SUM_BY_RESOURCE | SUM_BY_FEATURE | TOP_ACCOUNTS_BY_FEATURE`.

Currently, TOP_ACCOUNTS_BY_FEATURE does not support retrieving usage statistics for RDS_LOGIN_EVENTS.

- (Required) provide one or more data sources or features to query your usage statistics.
- (Optional) provide a list of account IDs for which you want to retrieve usage statistics.

You can also use the AWS Command Line Interface. The following command is an example about retrieving the usage statistics for all the data sources and features, calculated by accounts. Make sure to replace the detector-id with your own valid detector ID. For standalone accounts, this command returns the usage cost over the past 30 days for your account only. If you are a GuardDuty administrator account with member accounts, you see costs listed by account for all members.

To find the detectorId for your account and current Region, see **Settings** page in the https://console.aws.amazon.com/guardduty/ console.

Replace SUM_BY_ACCOUNT by the type with which you want to calculate the usage statistics.

To monitor cost for data sources only

```
aws guardduty get-usage-statistics --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --usage-statistic-type SUM_BY_ACCOUNT --usage-criteria '{"DataSources":
["FLOW_LOGS", "CLOUD_TRAIL", "DNS_LOGS", "S3_LOGS", "KUBERNETES_AUDIT_LOGS",
 "EC2_MALWARE_SCAN"]}'
```

To monitor cost for features

```
aws guardduty get-usage-statistics --detector-id 12abc34d567e8fa901bc2d34e56789f0
 --usage-statistic-type SUM_BY_ACCOUNT --usage-criteria '{"Features":
["FLOW_LOGS", "CLOUD_TRAIL", "DNS_LOGS", "S3_DATA_EVENTS", "EKS_AUDIT_LOGS",
 "EBS_MALWARE_PROTECTION", "RDS_LOGIN_EVENTS", "LAMBDA_NETWORK_LOGS",
 "EKS_RUNTIME_MONITORING", "FARGATE_RUNTIME_MONITORING", "EC2_RUNTIME_MONITORING"]}'
```

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. You are also responsible for 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 users with AWS IAM Identity Center or 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 require TLS 1.2 and recommend TLS 1.3.
- 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 sensitive 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 text 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 text 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 managed keys.

Encryption in transit

GuardDuty analyzes log data from other services. It 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.

Opting out of using your data for service improvement

You can choose to opt out of having your data used to develop and improve GuardDuty and other AWS security services by using the AWS Organizations opt-out policy. You can choose to opt out even if GuardDuty doesn't currently collect any such data. For more information about how to opt out, see AI services opt-out policies in the *AWS Organizations User Guide*.

> **Note**
>
> For you to use the opt-out policy, your AWS accounts must be centrally managed by AWS Organizations. If you haven't already created an organization for your AWS accounts, see Creating and managing an organization in the *AWS Organizations User Guide*.

Opting out has the following effects:

- GuardDuty will delete the data that it collected and stored for service improvement purposes prior to your opt out (if any).
- After you opt out, GuardDuty will no longer collect or store this data for service improvement purposes.

The following topics explain how each feature within GuardDuty potentially handles your data for service improvement.

Contents

GuardDuty Runtime Monitoring

GuardDuty Runtime Monitoring provides runtime threat detection for Amazon Elastic Kubernetes Service (Amazon EKS) clusters, AWS Fargate (Fargate) Amazon Elastic Container Service(Amazon ECS) only, and Amazon Elastic Compute Cloud (Amazon EC2) instances in your AWS environment. After you enable Runtime Monitoring and deploy the GuardDuty security agent for your resource, GuardDuty starts to monitor and analyze the runtime events associated with your resource. These runtime event types include process events, container events, DNS events, and more. For more information, see Collected runtime event types that GuardDuty uses.

Although GuardDuty now collects command-line arguments that you may direct to your workloads, it doesn't currently use these arguments for service improvement purposes (it may do so in the future). We have started collecting command-line arguments in anticipation of new threat detection rules and findings that will be released soon. Your trust, privacy, and the security of your content are our highest priority, and ensure that our use complies with our commitments to you. For more information, see Data Privacy FAQ.

GuardDuty Malware Protection

GuardDuty Malware Protection scans and detects malware contained in EBS volumes attached to your potentially compromised Amazon EC2 instance and container workloads. When GuardDuty Malware Protection identifies an EBS volume file as being malicious or harmful, GuardDuty Malware Protection collects and stores this file to develop and improve its malware detections, and the GuardDuty service. This file may also be used to develop and improve other AWS security services. Your trust, privacy, and the security of your content are our highest priority, and ensure that our use complies with our commitments to you. For more information, see Data Privacy FAQ.

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 all 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 Simple Storage Service (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 additional details.

For more information 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

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 user or IAM user's sign-in 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 CloudTrail userIdentity element.

GuardDuty control plane events in CloudTrail

By default, CloudTrail logs all the GuardDuty API operations provided in the Amazon GuardDuty API Reference as events in CloudTrail files.

GuardDuty data events in CloudTrail

GuardDuty Runtime Monitoring uses a GuardDuty security agent deployed to your Amazon Elastic Kubernetes Service (Amazon EKS) clusters, Amazon Elastic Compute Cloud (Amazon EC2) instances, and AWS Fargate (Amazon Elastic Container Service (Amazon ECS) only) tasks to collect add-on (`aws-guardduty-agent`) that collects Collected runtime event types for your AWS workloads and then send them to GuardDuty for threat detection and analysis.

Logging and monitoring data events

You can optionally configure the AWS CloudTrail logs to view the data events for your GuardDuty security agent.

To create and configure CloudTrail, see Data events in the *AWS CloudTrail User Guide* and follow the instructions for **Logging data events with advanced event selectors in the AWS Management Console**. While logging the trail, ensure to make the following changes:

- For the **Data event type**, choose **GuardDuty detector**.
- For the **Log selector template**, choose **Log all events**.
- Expand the **JSON view** for the configuration. It should be similar to the following JSON:

```
[
  {
    "name": "",
    "fieldSelectors": [
      {
        "field": "eventCategory",
        "equals": [
          "Data"
        ]
      },
      {
        "field": "resources.type",
        "equals": [
          "AWS::GuardDuty::Detector"
        ]
      }
    ]
  }
]
```

After you enable the selector for the trail, navigate to the Amazon S3 console at https://console.aws.amazon.com/s3/. You can download the data events from your S3 bucket chosen at the time of configuring the CloudTrail logs.

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 data plane event.

```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "111122223333:aws:ec2-instance:i-123412341234example",
            "arn": "arn:aws:sts::111122223333:assumed-role/aws:ec2-
instance/i-123412341234example",
            "accountId": "111122223333",
            "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "111122223333:aws:ec2-instance",
                    "arn": "arn:aws:iam::111122223333:role/aws:ec2-instance",
                    "accountId": "111122223333",
                    "userName": "aws:ec2-instance"
                },
                "attributes": {
                    "creationDate": "2023-03-05T04:00:21Z",
                    "mfaAuthenticated": "false"
                },
                "ec2RoleDelivery": "2.0"
            }
        },
        "eventTime": "2023-03-05T06:03:49Z",
        "eventSource": "guardduty.amazonaws.com",
        "eventName": "SendSecurityTelemetry",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "54.240.230.177",
        "userAgent": "aws-sdk-rust/0.54.1 os/linux lang/rust/1.66.0",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "resources": [{
            "accountId": "111122223333",
            "type": "AWS::GuardDuty::Detector",
            "ARN": "arn:aws:guardduty:us-
west-2:111122223333:detector/12abc34d567e8fa901bc2d34e56789f0"
        }],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "111122223333",
        "eventCategory": "Data",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "guardduty-data.us-east-1.amazonaws.com"
        }
    }
```

The following example shows a CloudTrail log entry that demonstrates the `CreateIPThreatIntelSet` action (control plane event).

```
{
    "eventVersion": "1.08",
    "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": "12abc34d567e8fa901bc2d34e56789f0",
        "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 a user named Alice on June 14, 2018.

Identity and Access Management for Amazon 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
- Authenticating with identities
- Managing access using policies
- How Amazon GuardDuty works with IAM
- Identity-based policy examples for Amazon GuardDuty
- Using service-linked roles for Amazon GuardDuty
- AWS managed policies for Amazon GuardDuty
- Troubleshooting Amazon GuardDuty identity and access

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 Amazon GuardDuty identity and access.

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 service users 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 Amazon GuardDuty works with IAM.

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 Amazon GuardDuty.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider] in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets] in the *AWS IAM Identity Center User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **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 AWS 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 AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. 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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 an 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 Amazon 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 Amazon GuardDuty

IAM feature	GuardDuty support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes

IAM feature	GuardDuty support
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	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 Amazon GuardDuty.

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 Amazon 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 Amazon GuardDuty.

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 Amazon GuardDuty in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon GuardDuty.

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for Amazon GuardDuty.

Policy condition keys for GuardDuty

Supports service-specific 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 Amazon GuardDuty in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon GuardDuty.

To view examples of GuardDuty identity-based policies, see Identity-based policy examples for Amazon GuardDuty.

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.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

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 forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

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 AWS 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.

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 Amazon GuardDuty

By default, users and roles don't have permission to create or modify GuardDuty resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by GuardDuty, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon GuardDuty in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the GuardDuty console
- Permissions required to enable GuardDuty
- Allow users to view their own permissions
- Custom IAM policy to grant read-only access to GuardDuty
- Deny Access to GuardDuty findings
- Using a custom IAM policy to limit access to GuardDuty resources

Policy best practices

Identity-based policies 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 with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and

functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM 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 (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 they'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

To grant permissions that various IAM identities (users, groups, and roles) must have, attach the required AWS managed policy: AmazonGuardDutyFullAccess policy to enable GuardDuty.

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 role, user, 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",
                "guardduty:DescribeMalwareScans",
                "guardduty:UpdateMalwareScanSettings",
                "guardduty:GetMalwareScanSettings"
            ],
            "Resource": "*"
        }
    ]
}
```

Deny Access to GuardDuty findings

You can use the following policy to deny an IAM role, user, 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:AcceptAdministratorInvitation",
                "guardduty:ListInvitations",
                "guardduty:GetInvitationsCount",
                "guardduty:DeclineInvitations",
                "guardduty:DeleteInvitations"
            ],
            "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::123456789012: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-1:123456789012: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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:123456789012: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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:123456789012:detector/*/
ipset/000000"
        }
    ]
}
```

- This policy allows a user to run the `guardduty:UpdateIPSet` operation, using their detector ID 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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "guardduty:UpdateIPSet",
             ],
            "Resource": "arn:aws:guardduty:us-east-1:123456789012:detector/1234567/
ipset/*"
        }
    ]
}
```

Using service-linked roles for Amazon GuardDuty

Amazon GuardDuty uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role (SLR) 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.

With service-linked role, you can set up GuardDuty without adding the necessary permissions manually. 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.

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 (SLR) named `AWSServiceRoleForAmazonGuardDuty`. The SLR allows GuardDuty to perform the following tasks. It also allows GuardDuty to include the retrieved metadata belonging to the EC2 instance in the findings that GuardDuty may generate about the potential threat. The `AWSServiceRoleForAmazonGuardDuty` service-linked role trusts the `guardduty.amazonaws.com` service to assume the role.

The permission policies help GuardDuty perform the following tasks:

- Use Amazon EC2 actions to manage and retrieve information about your EC2 instances, images, and networking components such as VPCs, subnets, transit gateways, and security groups.
- Use AWS Systems Manager actions to manage SSM associations on Amazon EC2 instances when you enable GuardDuty Runtime Monitoring with automated agent for Amazon EC2. When GuardDuty automated agent configuration is disabled, GuardDuty considers only those EC2 instances that have an inclusion tag (`GuardDutyManaged`:`true`).
- Use AWS Organizations actions to describe associated accounts and organization ID.
- Use Amazon S3 actions to retrieve information about S3 buckets and objects.
- Use AWS Lambda actions to retrieve information about your Lambda functions and tags.
- Use Amazon EKS actions to manage and retrieve information about the EKS clusters and manage Amazon EKS add-ons on EKS clusters. The EKS actions also retrieve the information about the tags associated to GuardDuty.
- Use IAM to create the Service-linked role permissions for Malware Protection after Malware Protection has been enabled.
- Use Amazon ECS actions to manage and retrieve information about the Amazon ECS clusters, and manage the Amazon ECS account setting with `guarddutyActivate`. The actions pertaining to Amazon ECS also retrieve the information about the tags associated with GuardDuty.

The role is configured with the following AWS managed policy, named `AmazonGuardDutyServiceRolePolicy`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GuardDutyGetDescribeListPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeImages",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeTransitGatewayAttachments",
                "organizations:ListAccounts",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "s3:GetBucketPublicAccessBlock",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketTagging",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "s3:GetBucketPolicy",
                "s3:GetBucketPolicyStatus",
                "lambda:GetFunctionConfiguration",
                "lambda:ListTags",
                "eks:ListClusters",
                "eks:DescribeCluster",
                "ec2:DescribeVpcEndpointServices",
                "ec2:DescribeSecurityGroups",
                "ecs:ListClusters",
                "ecs:DescribeClusters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "GuardDutyCreateSLRPolicy",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "malware-protection.guardduty.amazonaws.com"
                }
            }
        },
        {
            "Sid": "GuardDutyCreateVpcEndpointPolicy",
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyManaged"
                },
                "StringLike": {
                    "ec2:VpceServiceName": [
                        "com.amazonaws.*.guardduty-data",
                        "com.amazonaws.*.guardduty-data-fips"
                    ]
                }
            }
        },
        {
            "Sid": "GuardDutyModifyDeleteVpcEndpointPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpoint",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "GuardDutyCreateModifyVpcEndpointNetworkPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:ModifyVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
             "Sid": "GuardDutyCreateTagsDuringVpcEndpointCreationPolicy",
             "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyManaged"
                }
            }
        },
        {
            "Sid": "GuardDutySecurityGroupManagementPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/GuardDutyManaged": false
                }
            }
        },
        {
            "Sid": "GuardDutyCreateSecurityGroupPolicy",
            "Effect": "Allow",
            "Action": "ec2:CreateSecurityGroup",
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/GuardDutyManaged": "*"
                }
            }
        },
        {
            "Sid": "GuardDutyCreateSecurityGroupForVpcPolicy",
            "Effect": "Allow",
            "Action": "ec2:CreateSecurityGroup",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        },
        {
            "Sid": "GuardDutyCreateTagsDuringSecurityGroupCreationPolicy",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateSecurityGroup"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyManaged"
                }
            }
        },
        {
            "Sid": "GuardDutyCreateEksAddonPolicy",
            "Effect": "Allow",
            "Action": "eks:CreateAddon",
            "Resource": "arn:aws:eks:*:*:cluster/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyManaged"
                }
            }
        },
        {
            "Sid": "GuardDutyEksAddonManagementPolicy",
            "Effect": "Allow",
            "Action": [
                "eks:DeleteAddon",
                "eks:UpdateAddon",
                "eks:DescribeAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/aws-guardduty-agent/*"
        },
        {
            "Sid": "GuardDutyEksClusterTagResourcePolicy",
            "Effect": "Allow",
            "Action": "eks:TagResource",
            "Resource": "arn:aws:eks:*:*:cluster/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyManaged"
                }
            }
        },
        {
            "Sid": "GuardDutyEcsPutAccountSettingsDefaultPolicy",
            "Effect": "Allow",
            "Action": "ecs:PutAccountSettingDefault",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:account-setting": [
                        "guardDutyActivate"
                    ]
                 }
            }
        },
        {
            "Sid": "SsmCreateDescribeUpdateDeleteStartAssociationPermission",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:DeleteAssociation",
                "ssm:UpdateAssociation",
                "ssm:CreateAssociation",
                "ssm:StartAssociationsOnce"
            ],
            "Resource": "arn:aws:ssm:*:*:association/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/GuardDutyManaged": "true"
                }
            }
        },
        {
            "Sid": "SsmAddTagsToResourcePermission",
            "Effect": "Allow",
            "Action": [
                "ssm:AddTagsToResource"
            ],
            "Resource": "arn:aws:arn:aws:ssm:*:*:association/*",
            "Condition":{
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyManaged"
                    ]
                },
                "StringEquals": {
                    "aws:ResourceTag/GuardDutyManaged": "true"
                }
            }
        },
        {
            "Sid": "SsmCreateUpdateAssociationInstanceDocumentPermission",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateAssociation",
                "ssm:UpdateAssociation"
            ],
            "Resource": "arn:aws:ssm:*:*:document/AmazonGuardDuty-
ConfigureRuntimeMonitoringSsmPlugin"
        },
        {
            "Sid": "SsmSendCommandPermission",
            "Effect": "Allow",
            "Action": "ssm:SendCommand",
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ssm:*:*:document/AmazonGuardDuty-
ConfigureRuntimeMonitoringSsmPlugin"
            ]
        },
        {
            "Sid": "SsmGetCommandStatus",
            "Effect": "Allow",
            "Action": "ssm:GetCommandInvocation",
            "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"
    }
  ]
}
```

For details about updates to the `AmazonGuardDutyServiceRolePolicy` policy, see GuardDuty updates to AWS managed policies. For automatic alerts about changes to this policy, subscribe to the RSS feed on the Document history page.

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 service-linked role manually using the IAM console, the AWS 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 principal (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 principal that you use GuardDuty with must have the required permissions. To grant the required permissions, attach the following policy to this user, group, or role:

> ⓘ **Note**
>
> Replace the sample *`account ID`* in the following example 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::123456789012: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

If you have enabled Malware Protection, deleting `AWSServiceRoleForAmazonGuardDuty` doesn't automatically delete `AWSServiceRoleForAmazonGuardDutyMalwareProtection`. If you want to delete `AWSServiceRoleForAmazonGuardDutyMalwareProtection`, see Deleting a service-linked role for Malware Protection.

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.

When you disable GuardDuty, the `AWSServiceRoleForAmazonGuardDuty` doesn't get deleted automatically. If you enable GuardDuty again, it'll start using the existing `AWSServiceRoleForAmazonGuardDuty`.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS 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*.

Supported AWS Regions

Amazon GuardDuty supports using the `AWSServiceRoleForAmazonGuardDuty` service-linked role in all the AWS Regions where GuardDuty is available. For a list of Regions where GuardDuty is currently available, see Amazon GuardDuty endpoints and quotas in the *Amazon Web Services General Reference*.

Service-linked role permissions for Malware Protection

Malware Protection uses the service-linked role (SLR) named `AWSServiceRoleForAmazonGuardDutyMalwareProtection`. This SLR allows Malware Protection to perform agentless scans to detect malware in your GuardDuty account. It allows GuardDuty to create an EBS volume snapshot in your account, and share that snapshot with the GuardDuty service account. After GuardDuty evaluates the snapshot, it includes the retrieved EC2 instance and container workload metadata in the Malware Protection findings. The

`AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role trusts the `malware-protection.guardduty.amazonaws.com` service to assume the role.

The permission policies for this role helps Malware Protection to perform the following tasks:

- Use Amazon Elastic Compute Cloud (Amazon EC2) actions to retrieve information about your Amazon EC2 instances, volumes, and snapshots. Malware Protection also provides permission to access the Amazon EKS and Amazon ECS cluster metadata.
- Create snapshots for EBS volumes that have `GuardDutyExcluded` tag not set to `true`. By default, the snapshots get created with a `GuardDutyScanId` tag. Don't remove this tag, otherwise Malware Protection will not have access to the snapshots.

> ⚠ **Important**
>
> When you set the `GuardDutyExcluded` to `true`, the GuardDuty service won't be able to access these snapshots in the future. This is because the other statements in this service-linked role prevent GuardDuty from performing any action on the snapshots that have the `GuardDutyExcluded` set to `true`.

- Allow sharing and deleting snapshots only if the `GuardDutyScanId` tag exists and `GuardDutyExcluded` tag is not set to `true`.

> ⓘ **Note**
>
> Doesn't allow Malware Protection to make the snapshots public.

- Access customer managed keys, except those that have a `GuardDutyExcluded` tag set to `true`, to call `CreateGrant` to create and access an encrypted EBS volume from the encrypted snapshot that gets shared with the GuardDuty service account. For a list of GuardDuty service accounts for each Region, see GuardDuty service accounts by AWS Region.
- Access customers' CloudWatch logs to create the Malware Protection log group as well as put the malware scan events logs under the `/aws/guardduty/malware-scan-events` log group.
- Allow the customer to decide if they want to keep the snapshots on which malware was detected, in their account. If the scan detects malware, the service-linked role allows GuardDuty to add two tags to snapshots - `GuardDutyFindingDetected` and `GuardDutyExcluded`.

> **Note**
>
> The `GuardDutyFindingDetected` tag specifies that the snapshots contains malware.

- Determine if a volume is encrypted with an EBS managed key. GuardDuty performs the `DescribeKey` action to determine the `key  Id` of the EBS-managed key in your account.
- Fetch the snapshot of the EBS volumes encrypted using AWS managed key, from your AWS account and copy it to the GuardDuty service account. For this purpose, we use the permissions `GetSnapshotBlock` and `ListSnapshotBlocks`. GuardDuty will then scan the snapshot in the service account. Presently, the Malware Protection support for scanning EBS volumes encrypted with AWS managed key might not be available in all the AWS Regions. For more information, see Region-specific feature availability.
- Allow Amazon EC2 to call AWS KMS on behalf of Malware Protection to perform several cryptographic actions on customer managed keys. Actions such as `kms:ReEncryptTo` and `kms:ReEncryptFrom` are required to share the snapshots that are encrypted with the customer managed keys. Only those keys are accessible for which the `GuardDutyExcluded` tag is not set to `true`.

The role is configured with the following AWS managed policy, named `AmazonGuardDutyMalwareProtectionServiceRolePolicy`.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "DescribeAndListPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeVolumes",
                "ec2:DescribeSnapshots",
                "ecs:ListClusters",
                "ecs:ListContainerInstances",
                "ecs:ListTasks",
                "ecs:DescribeTasks",
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateSnapshotVolumeConditionalStatement",
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshot",
            "Resource": "arn:aws:ec2:*:*:volume/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/GuardDutyExcluded": "true"
                }
            }
        },
        {
            "Sid": "CreateSnapshotConditionalStatement",
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshot",
            "Resource": "arn:aws:ec2:*:*:snapshot/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "GuardDutyScanId"
                }
            }
        },
        {
            "Sid": "CreateTagsPermission",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:*/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateSnapshot"
                }
            }
        },
        {
            "Sid": "AddTagsToSnapshotPermission",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:snapshot/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/GuardDutyScanId": "*"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "GuardDutyExcluded",
                        "GuardDutyFindingDetected"
                    ]
                }
            }
        },
        {
            "Sid": "DeleteAndShareSnapshotPermission",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteSnapshot",
                "ec2:ModifySnapshotAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:snapshot/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/GuardDutyScanId": "*"
                },
                "Null": {
                    "aws:ResourceTag/GuardDutyExcluded": "true"
                }
            }
        },
        {
            "Sid": "PreventPublicAccessToSnapshotPermission",
            "Effect": "Deny",
            "Action": [
                "ec2:ModifySnapshotAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:snapshot/*",
            "Condition": {
                "StringEquals": {
                    "ec2:Add/group": "all"
                }
            }
        },
        {
            "Sid": "CreateGrantPermission",
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "arn:aws:kms:*:*:key/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/GuardDutyExcluded": "true"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:ebs:id": "snap-*"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "CreateGrant",
                        "GenerateDataKeyWithoutPlaintext",
                        "ReEncryptFrom",
                        "ReEncryptTo",
                        "RetireGrant",
                        "DescribeKey"
                    ]
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        },
        {
            "Sid": "ShareSnapshotKMSPermission",
            "Effect": "Allow",
            "Action": [
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom"
            ],
            "Resource": "arn:aws:kms:*:*:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "ec2.*.amazonaws.com"
                },
                "Null": {
                    "aws:ResourceTag/GuardDutyExcluded": "true"
                }
            }
        },
        {
            "Sid": "DescribeKeyPermission",
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "arn:aws:kms:*:*:key/*"
        },
        {
            "Sid": "GuardDutyLogGroupPermission",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/guardduty/*"
        },
        {
            "Sid": "GuardDutyLogStreamPermission",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/guardduty/*:log-stream:*"
        },
        {
            "Sid": "EBSDirectAPIPermissions",
            "Effect": "Allow",
            "Action": [
                "ebs:GetSnapshotBlock",
                "ebs:ListSnapshotBlocks"
            ],
            "Resource": "arn:aws:ec2:*:*:snapshot/*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/GuardDutyScanId": "*"
                },
                "Null": {
                    "aws:ResourceTag/GuardDutyExcluded": "true"
                }
            }
        }
    ]
}
```

The following trust policy is attached to the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "malware-protection.guardduty.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating a service-linked role for Malware Protection

The `AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role is automatically created when you enable Malware Protection for the first time or enable Malware Protection in a supported Region where you previously didn't have it enabled. You can also create the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role manually using the IAM console, the IAM CLI, or the IAM API.

> **Note**
>
> By default, if you are new to Amazon GuardDuty, Malware Protection is automatically enabled.

> **Important**
>
> The service-linked role that is created for the delegated GuardDuty administrator account doesn't apply to the member GuardDuty accounts.

You must configure permissions to allow an IAM principal (such as a user, group, or role) to create, edit, or delete a service-linked role. For the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` 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 user, group, or role:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "guardduty:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "malware-protection.guardduty.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:RegisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/
*AWSServiceRoleForAmazonGuardDutyMalwareProtection"
        }
    ]
}
```

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 Malware Protection

Malware Protection doesn't allow you to edit the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` 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 Malware Protection

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**
>
> In order to delete the `AWSServiceRoleForAmazonGuardDutyMalwareProtection`, you must first disable Malware Protection in all of the Regions where it is enabled.
> If Malware Protection isn't disabled when you try to delete the service-linked role, the deletion will fail. For more information, see To enable or disable GuardDuty-initiated malware scan.

When you choose **Disable** to stop the Malware Protection service, the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` is not automatically deleted. If you then choose **Enable** to start the Malware Protection service again, GuardDuty will start using the existing `AWSServiceRoleForAmazonGuardDutyMalwareProtection`.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the IAM API to delete the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported AWS Regions

Amazon GuardDuty supports using the `AWSServiceRoleForAmazonGuardDutyMalwareProtection` service-linked role in all the AWS Regions where Malware Protection is available.

For a list of Regions where GuardDuty is currently available, see Amazon GuardDuty endpoints and quotas in the *Amazon Web Services General Reference*.

Note

Malware Protection is currently unavailable in AWS GovCloud (US-East) and AWS GovCloud (US-West).

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.

The permission to perform an `iam:GetRole` action on `AWSServiceRoleForAmazonGuardDutyMalwareProtection` establishes if the service-linked role (SLR) for Malware Protection exists in an account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonGuardDutyFullAccessSid1",
            "Effect": "Allow",
            "Action": "guardduty:*",
            "Resource": "*"
        },
        {
            "Sid": "CreateServiceLinkedRoleSid1",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "guardduty.amazonaws.com",
                        "malware-protection.guardduty.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "ActionsForOrganizationsSid1",
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:RegisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "organizations:ListAccounts"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IamGetRoleSid1",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/
*AWSServiceRoleForAmazonGuardDutyMalwareProtection"
        }
    ]
}
```

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",
                "organizations:ListAccounts"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policy: AmazonGuardDutyServiceRolePolicy

You can't attach `AmazonGuardDutyServiceRolePolicy` 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 Service-linked role permissions for GuardDuty.

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
AmazonGuardDutyServiceRolePolicy – Update to an existing policy.	Use AWS Systems Manager actions to manage SSM associations on Amazon EC2 instances when you enable GuardDuty Runtime	March 26, 2024

Change	Description	Date
	Monitoring with automated agent for Amazon EC2. When GuardDuty automated agent configuration is disabled, GuardDuty considers only those EC2 instances that have an inclusion tag (`GuardDuty Managed :true`).	
AmazonGuardDutyServiceRolePolicy – Update to an existing policy.	GuardDuty has added a new permission - `organization:DescribeOrganization` to retrieve the organization ID of the shared Amazon VPC account and set the Amazon VPC endpoint policy with organization ID.	February 9, 2024
AmazonGuardDutyMalwareProtectionServiceRolePolicy – Update to an existing policy.	Malware Protection has added two permissions - `GetSnapshotBlock` and `ListSnapshotBlocks` to fetch the snapshot of an EBS volume (encrypted using AWS managed key) from your AWS account and copy it to the GuardDuty service account before starting the malware scan.	Jan 25, 2024

Change	Description	Date
AmazonGuardDutyServiceRolePolicy – Update to an existing policy	Added new permissions to allow GuardDuty to add `guarddutyActivate` Amazon ECS account setting, and perform list and describe operations on Amazon ECS clusters.	Nov 26, 2023
AmazonGuardDutyReadOnlyAccess – Update to an existing policy	GuardDuty added a new policy for `organizations` to `ListAccounts` .	November 16, 2023
AmazonGuardDutyFullAccess – Update to an existing policy	GuardDuty added a new policy for `organizations` to `ListAccounts` .	November 16, 2023
AmazonGuardDutyServiceRolePolicy – Update to an existing policy	GuardDuty added new permissions to support the upcoming GuardDuty EKS Runtime Monitoring feature.	March 8, 2023

Change	Description	Date

AmazonGuardDutyServiceRolePolicy – Update to an existing policy	GuardDuty has added new permissions to allow GuardDuty to create Service-linked role for Malware Protection. This will help GuardDuty streamline the process of enabling Malware Protection.
	GuardDuty can now perform the following IAM action:
	`{`
	`    "Effect": "Allow",`
	` "Action": "iam:CreateServiceLinkedRole",`
	` "Resource": "*",`
	` "Condition": {`
	`    "StringEquals": {`
	`        "iam:AWSServiceName": "malware-protection.guardduty.amazonaws.com"`
	`    }`
	` }`
	`}`
		Feb 21, 2023

AmazonGuardDutyFullAccess – Update to an existing policy	GuardDuty updated ARN for `iam:GetRole` to `*AWSServiceRoleForAmazonGuardDutyMalwareProtection` .	Jul 26, 2022

Change	Description	Date
AmazonGuardDutyFullAccess – Update to an existing policy	GuardDuty added a new `AWSServiceName` to allow the creation of service-linked role using `iam:CreateServiceLinkedRole` for GuardDuty Malware Protection service. GuardDuty can now perform the `iam:GetRole` action to gain information for `AWSServiceRole` .	Jul 26, 2022
AmazonGuardDutyServiceRolePolicy – 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

Change	Description	Date
GuardDuty started tracking changes	GuardDuty started tracking changes for its AWS managed policies.	Aug 3, 2021

Troubleshooting Amazon 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
- I'm not authorized to perform iam:PassRole.
- I want to allow people outside of my AWS account to access my GuardDuty resources.

I am not authorized to perform an action in GuardDuty

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

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 doesn't 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, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `guardduty:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm not authorized to perform iam:PassRole.

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated 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 that are 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's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

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 Amazon GuardDuty works with IAM.
- 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*.

Compliance validation for Amazon GuardDuty

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. 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 AWS services 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 baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service 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. Security Hub uses security controls to evaluate your AWS resources and to check your

compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

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, Amazon GuardDuty is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well‐Architected Framework*.

You use AWS published API calls to access GuardDuty through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by 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.

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.

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. For more information, see Viewing findings in the *AWS Security Hub User Guide*. You can also track the status of an investigation into a finding. For more information, 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 you enable GuardDuty and Security Hub in the same account within the same AWS Region, GuardDuty starts sending all the generated findings to Security Hub. These 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 new 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 new observations of these findings are sent to Security Hub based on the Step 5 – Export update frequency settings in your AWS account.

When you archive or unarchive a finding, GuardDuty doesn't send that finding to Security Hub. Any manually unarchived finding that later become active in GuardDuty is not 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.

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

GuardDuty finding type	ASFF finding type
Backdoor:Lambda/C&CActivity.B	TTPs/Command and Control/Backdoor:Lambda-C&CActivity.B
Backdoor:Runtime/C&CActivity.B	TTPs/Command and Control/Backdoor:Runtime-C&CActivity.B
Backdoor:Runtime/C&CActivity.B!DNS	TTPs/Command and Control/Backdoor:Runtime-C&CActivity.B!DNS
CredentialAccess:IAMUser/AnomalousBehavior	TTPs/Credential Access/IAMUser-AnomalousBehavior
CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed	TTPs/AnomalousBehavior/CredentialAccess:Kubernetes-SecretsAccessed
CredentialAccess:RDS/AnomalousBehavior.FailedLogin	TTPs/Credential Access/CredentialAccess:RDS-AnomalousBehavior.FailedLogin
CredentialAccess:RDS/AnomalousBehavior.SuccessfulBruteForce	TTPs/Credential Access/RDS-AnomalousBehavior.SuccessfulBruteForce
CredentialAccess:RDS/AnomalousBehavior.SuccessfulLogin	TTPs/Credential Access/RDS-AnomalousBehavior.SuccessfulLogin
CredentialAccess:RDS/MaliciousIPCaller.FailedLogin	TTPs/Credential Access/RDS-MaliciousIPCaller.FailedLogin
CredentialAccess:RDS/MaliciousIPCaller.SuccessfulLogin	TTPs/Credential Access/RDS-MaliciousIPCaller.SuccessfulLogin
CredentialAccess:RDS/TorIPCaller.FailedLogin	TTPs/Credential Access/RDS-TorIPCaller.FailedLogin
CredentialAccess:RDS/TorIPCaller.SuccessfulLogin	TTPs/Credential Access/RDS-TorIPCaller.SuccessfulLogin
CryptoCurrency:EC2/BitcoinTool.B	TTPs/Command and Control/CryptoCurrency:EC2-BitcoinTool.B

GuardDuty finding type	ASFF finding type
CryptoCurrency:EC2/BitcoinTool.B!DNS	TTPs/Command and Control/CryptoCurrency:EC2-BitcoinTool.B!DNS
CryptoCurrency:Lambda/BitcoinTool.B	TTPs/Command and Control/CryptoCurrency:Lambda-BitcoinTool.B Effects/Resource Consumption/CryptoCurrency:Lambda-BitcoinTool.B
CryptoCurrency:Runtime/BitcoinTool.B	TTPs/Command and Control/CryptoCurrency:Runtime-BitcoinTool.B
CryptoCurrency:Runtime/BitcoinTool.B!DNS	TTPs/Command and Control/CryptoCurrency:Runtime-BitcoinTool.B!DNS
DefenseEvasion:EC2/UnusualDNSResolver	TTPs/DefenseEvasion/EC2:Unusual-DNS-Resolver
DefenseEvasion:EC2/UnusualDoHActivity	TTPs/DefenseEvasion/EC2:Unusual-DoH-Activity
DefenseEvasion:EC2/UnusualDoTActivity	TTPs/DefenseEvasion/EC2:Unusual-DoT-Activity
DefenseEvasion:IAMUser/AnomalousBehavior	TTPs/Defense Evasion/IAMUser-AnomalousBehavior
DefenseEvasion:Runtime/FilelessExecution	TTPs/Defense Evasion/DefenseEvasion:Runtime-FilelessExecution
DefenseEvasion:Runtime/PtraceAntiDebugging	TTPs/DefenseEvasion/DefenseEvasion:Runtime-PtraceAntiDebugging
DefenseEvasion:Runtime/SuspiciousCommand	TTPs/DefenseEvasion/DefenseEvasion:Runtime-SuspiciousCommand
Discovery:IAMUser/AnomalousBehavior	TTPs/Discovery/IAMUser-AnomalousBehavior

GuardDuty finding type	ASFF finding type
Discovery:Kubernetes/AnomalousBehavior.PermissionChecked	TTPs/AnomalousBehavior/Discovery:Kubernetes-PermissionChecked
Discovery:RDS/MaliciousIPCaller	TTPs/Discovery/RDS-MaliciousIPCaller
Discovery:RDS/TorIPCaller	TTPs/Discovery/RDS-TorIPCaller
Discovery:S3/AnomalousBehavior	TTPs/Discovery:S3-AnomalousBehavior
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
Execution:Kubernetes/AnomalousBehavior.ExecInPod	TTPs/AnomalousBehavior/Execution:Kubernetes-ExecInPod
Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed	TTPs/AnomalousBehavior/Execution:Kubernetes-WorkloadDeployed
Persistence:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount	TTPs/AnomalousBehavior/Persistence:Kubernetes-WorkloadDeployed!ContainerWithSensitiveMount
PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer	TTPs/AnomalousBehavior/PrivilegeEscalation:Kubernetes-WorkloadDeployed!PrivilegedContainer
Execution:EC2/MaliciousFile	TTPs/Execution/Execution:EC2-MaliciousFile
Execution:ECS/MaliciousFile	TTPs/Execution/Execution:ECS-MaliciousFile
Execution:Kubernetes/MaliciousFile	TTPs/Execution/Execution:Kubernetes-MaliciousFile

GuardDuty finding type	ASFF finding type
Execution:Container/MaliciousFile	TTPs/Execution/Execution:Container-MaliciousFile
Execution:EC2/SuspiciousFile	TTPs/Execution/Execution:EC2-SuspiciousFile
Execution:ECS/SuspiciousFile	TTPs/Execution/Execution:ECS-SuspiciousFile
Execution:Kubernetes/SuspiciousFile	TTPs/Execution/Execution:Kubernetes-SuspiciousFile
Execution:Container/SuspiciousFile	TTPs/Execution/Execution:Container-SuspiciousFile
Execution:Runtime/MaliciousFileExecuted	TTPs/Execution/Execution:Runtime-MaliciousFileExecuted
Execution:Runtime/NewBinaryExecuted	TTPs/Execution/Execution:Runtime-NewBinaryExecuted
Execution:Runtime/NewLibraryLoaded	TTPs/Execution/Execution:Runtime-NewLibraryLoaded
Execution:Runtime/ReverseShell	TTPs/Execution/Execution:Runtime-ReverseShell
Execution:Runtime/SuspiciousCommand	TTPs/Execution/Execution:Runtime-SuspiciousCommand
Execution:Runtime/SuspiciousTool	TTPs/Execution/Execution:Runtime-SuspiciousTool
Exfiltration:S3/AnomalousBehavior	TTPs/Exfiltration:S3-AnomalousBehavior
Exfiltration:S3/ObjectRead.Unusual	TTPs/Exfiltration:S3-ObjectRead.Unusual
Exfiltration:S3/MaliciousIPCaller	TTPs/Exfiltration:S3-MaliciousIPCaller

GuardDuty finding type	ASFF finding type
Impact:EC2/AbusedDomainRequest.Reputation	TTPs/Impact:EC2-AbusedDomainRequest.Reputation
Impact:EC2/BitcoinDomainRequest.Reputation	TTPs/Impact:EC2-BitcoinDomainRequest.Reputation
Impact:EC2/MaliciousDomainRequest.Reputation	TTPs/Impact:EC2-MaliciousDomainRequest.Reputation
Impact:EC2/PortSweep	TTPs/Impact/Impact:EC2-PortSweep
Impact:EC2/SuspiciousDomainRequest.Reputation	TTPs/Impact:EC2-SuspiciousDomainRequest.Reputation
Impact:EC2/WinRMBruteForce	TTPs/Impact/Impact:EC2-WinRMBruteForce
Impact:IAMUser/AnomalousBehavior	TTPs/Impact/IAMUser-AnomalousBehavior
Impact:Runtime/AbusedDomainRequest.Reputation	TTPs/Impact/Impact:Runtime-AbusedDomainRequest.Reputation
Impact:Runtime/BitcoinDomainRequest.Reputation	TTPs/Impact/Impact:Runtime-BitcoinDomainRequest.Reputation
Impact:Runtime/CryptoMinerExecuted	TTPs/Impact/Impact:Runtime-CryptoMinerExecuted
Impact:Runtime/MaliciousDomainRequest.Reputation	TTPs/Impact/Impact:Runtime-MaliciousDomainRequest.Reputation
Impact:Runtime/SuspiciousDomainRequest.Reputation	TTPs/Impact/Impact:Runtime-SuspiciousDomainRequest.Reputatio
Impact:S3/AnomalousBehavior.Delete	TTPs/Impact:S3-AnomalousBehavior.Delete
Impact:S3/AnomalousBehavior.Permission	TTPs/Impact:S3-AnomalousBehavior.Permission

GuardDuty finding type	ASFF finding type
Impact:S3/AnomalousBehavior.Write	TTPs/Impact:S3-AnomalousBehavior.Write
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
InitialAccess:IAMUser/AnomalousBehavior	TTPs/Initial Access/IAMUser-AnomalousBehavior
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/AnomalousBehavior	TTPs/Persistence/IAMUser-AnomalousBehavior
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

GuardDuty finding type	ASFF finding type
Policy:S3/AccountBlockPublicAccessDisabled	TTPs/Policy:S3-AccountBlockPublicAccessDisabled
Policy:S3/BucketAnonymousAccessGranted	TTPs/Policy:S3-BucketAnonymousAccess Granted
Policy:S3/BucketBlockPublicAccessDisabled	Effects/Data Exposure/Policy:S3-BucketBlockPublicAccessDisabled
Policy:S3/BucketPublicAccessGranted	TTPs/Policy:S3-BucketPublicAccessGranted
PrivilegeEscalation:IAMUser/AnomalousBehavior	TTPs/Privilege Escalation/IAMUser-AnomalousBehavior
PrivilegeEscalation:IAMUser/AdministrativePermissions	TTPs/Privilege Escalation/PrivilegeEscalation:IAMUser-AdministrativePermissions
PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated	TTPs/AnomalousBehavior/PriviledgeEscalation:Kubernetes-RoleBindingCreated
PriviledgeEscalation:Kubernetes/AnomalousBehavior.RoleCreated	TTPs/AnomalousBehavior/PriviledgeEscalation:Kubernetes-RoleCreated
PrivilegeEscalation:Runtime/ContainerMountsHostDirectory	TTPs/Privilege Escalation/PrivilegeEscalation:Runtime-ContainerMountsHostDirectory
PrivilegeEscalation:Runtime/CGroupsReleaseAgentModified	TTPs/Privilege Escalation/PrivilegeEscalation:Runtime-CGroupsReleaseAgentModified
PrivilegeEscalation:Runtime/DockerSocketAccessed	TTPs/Privilege Escalation/PrivilegeEscalation:Runtime-DockerSocketAccessed
PrivilegeEscalation:Runtime/RuncContainerEscape	TTPs/Privilege Escalation/PrivilegeEscalation:Runtime-RuncContainerEscape
PrivilegeEscalation:Runtime/UserfaultfdUsage	TTPs/Privilege Escalation/PrivilegeEscalation:Runtime-UserfaultfdUsage

GuardDuty finding type	ASFF finding type
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
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

GuardDuty finding type	ASFF finding type
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
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
Trojan:Lambda/BlackholeTraffic	TTPs/Command and Control/Trojan:Lambda-BlackholeTraffic
Trojan:Lambda/DropPoint	Effects/Data Exfiltration/Trojan:Lambda-DropPoint
Trojan:Runtime/BlackholeTraffic	TTPs/Command and Control/Trojan:Runtime-BlackholeTraffic

GuardDuty finding type	ASFF finding type
Trojan:Runtime/BlackholeTraffic!DNS	TTPs/Command and Control/Trojan:Runtime-BlackholeTraffic!DNS
Trojan:Runtime/DGADomainRequest.C!DNS	TTPs/Command and Control/Trojan:Runtime-DGADomainRequest.C!DNS
Trojan:Runtime/DriveBySourceTraffic!DNS	TTPs/Initial Access/Trojan:Runtime-DriveBySourceTraffic!DNS
Trojan:Runtime/DropPoint	Effects/Data Exfiltration/Trojan:Runtime-DropPoint
Trojan:Runtime/DropPoint!DNS	Effects/Data Exfiltration/Trojan:Runtime-DropPoint!DNS
Trojan:Runtime/PhishingDomainRequest!DNS	TTPs/Command and Control/Trojan:Runtime-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
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

GuardDuty finding type	ASFF finding type
UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B	TTPs/UnauthorizedAccess:IAMUser-ConsoleLoginSuccess.B
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS	Effects/Data Exfiltration/UnauthorizedAccess:IAMUser-InstanceCredentialExfiltration.InsideAWS
UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS	Effects/Data Exfiltration/UnauthorizedAccess:IAMUser-InstanceCredentialExfiltration.OutsideAWS
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:Lambda/MaliciousIPCaller.Custom	TTPs/Command and Control/UnauthorizedAccess:Lambda-MaliciousIPCaller.Custom
UnauthorizedAccess:Lambda/TorClient	Effects/Resource Consumption/UnauthorizedAccess:Lambda-TorClient
UnauthorizedAccess:Lambda/TorRelay	Effects/Resource Consumption/UnauthorizedAccess:Lambda-TorRelay
UnauthorizedAccess:Runtime/MetadataDNSRebind	TTPs/UnauthorizedAccess:Runtime-MetadataDNSRebind
UnauthorizedAccess:Runtime/TorRelay	Effects/Resource Consumption/UnauthorizedAccess:Runtime-TorRelay
UnauthorizedAccess:Runtime/TorClient	Effects/Resource Consumption/UnauthorizedAccess:Runtime-TorClient

GuardDuty finding type	ASFF finding type
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.

Detective ingests finding details for all finding types, and provides access to the entity profiles 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 role, user, 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 frequency to 15 minutes in each region in which you use Detective with GuardDuty. For more information see Step 5 – Setting frequency to export updated active findings.

Pivoting to Amazon Detective from a GuardDuty finding

1. Log into the https://console.aws.amazon.com/guardduty/ console.
2. Choose a single finding from your findings table.
3. Choose **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, see Troubleshooting the pivot in the *Amazon Detective User Guide*.

Note

If you archive a GuardDuty finding in the Detective console, that finding gets 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 the GuardDuty service. You don't get charged for using GuardDuty when the service is suspended.

- All member accounts must be disassociated or deleted before you can suspend or disable 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.
- When you disable GuardDuty in an account, it will be disabled only for the currently selected AWS Region. If you want to completely disable GuardDuty, you must disable it in each Region where it is enabled.
- 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 confirm to disable GuardDuty. For information on how to export findings, see Exporting findings.

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. In the **Suspend GuardDuty** section, choose **Suspend GuardDuty** or **Disable GuardDuty**, then **Confirm** your action.

To re-enable GuardDuty after suspending

1. Open the GuardDuty console at https://console.aws.amazon.com/guardduty/.
2. In the navigation pane, choose **Settings**.
3. Choose **Re-enable GuardDuty**.

Subscribing to Amazon SNS GuardDuty announcements

This section provides information about subscribing to Amazon SNS (Simple Notification Service) for GuardDuty announcements 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 sends announcement about updates to the GuardDuty service across AWS to any subscribed account. To receive notifications about findings within your account, see Creating custom responses to GuardDuty findings with Amazon CloudWatch Events.

Note

Your IAM user must have `sns::subscribe` permissions to subscribe to an SNS.

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 Queue 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`

AWS Region	Amazon SNS topic ARN
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
ca-west-1	arn:aws:sns:ca-west-1:440427180217: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

AWS Region	Amazon SNS topic ARN
eu-west-3	arn:aws:sns:eu-west-3:436163563069:GuardDutyAnnouncements
eu-central-1	arn:aws:sns:eu-central-1:378365507264:GuardDutyAnnouncements
eu-central-2	arn:aws:sns:eu-central-2:383009515534:GuardDutyAnnouncements
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

AWS Region	Amazon SNS topic ARN
`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`
`me-south-1`	`arn:aws:sns:me-south-1:552740612889:GuardDutyAnnouncements`

AWS Region	Amazon SNS topic ARN
`me-central-1`	`arn:aws:sns:me-central-1:030935290150:GuardDutyAnnouncements`
`eu-south-1`	`arn:aws:sns:eu-south-1:188461706213:GuardDutyAnnouncements`
`eu-south-2`	`arn:aws:sns:eu-south-2:445632894446:GuardDutyAnnouncements`
`us-gov-east-1`	`arn:aws:sns:us-gov-east-1:143972945659:GuardDutyAnnouncements`
`ap-northeast-3`	`arn:aws:sns:ap-northeast-3:129086577509:GuardDutyAnnouncements`
`ap-southeast-3`	`arn:aws:sns:ap-southeast-3:225965583551:GuardDutyAnnouncements`
`ap-south-2`	`arn:aws:sns:ap-south-2:595653072700:GuardDutyAnnouncements`

AWS Region	Amazon SNS topic ARN
ap-southeast-4	arn:aws:sns:ap-southeast-4:529900636122:GuardDutyAnnouncements
il-central-1	arn:aws:sns:il-central-1:847886274986: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.

> **Note**
>
> For quotas specific to GuardDuty Malware Protection, see Malware Protection quotas.

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. You cannot request a quota increase.

Resource	Default	Comments
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 addresses or CIDR ranges to include in a Trusted IP List or a Threat List. You cannot request a quota increase.

Resource	Default	Comments
Member accounts (by invitation)	5000	The maximum number of member accounts associated with a administrator account account.
Member accounts	50,000	The maximum number of member accounts associated with a administrator account account through AWS Organizations. This includes member accounts that are added to the organization by invitation.
Threat intel sets	6	The maximum number of Threat intel sets that you can add per AWS account per Region. You cannot request a quota increase.

Resource	Default	Comments
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.

Troubleshooting Amazon GuardDuty

When you receive issues related to performing an action specific to GuardDuty, consult the topics in this section.

Topics

General issues in GuardDuty

I am getting an access error while exporting GuardDuty findings. How can I resolve this?

After you configure settings to export findings, if GuardDuty is unable to export findings, it displays an error message on the **Settings** page in the GuardDuty console. This can potentially happen when GuardDuty can no longer access the target resource, for example, if your Amazon S3 bucket was deleted or the permission to access the bucket was modified. This can also potentially happen when GuardDuty can no longer access the AWS KMS key that was used to encrypt the data in your Amazon S3 bucket. When GuardDuty is unable to export, it sends a notification to the email associated with the account to provide information about this issue.

To resolve the issue, make sure that the corresponding resources exist and GuardDuty has the permissions to access the needed resources. If you don't resolve the issue before the 90-day finding retention period completes in GuardDuty, your findings will not get exported. GuardDuty will disable finding export settings for this account in the specific Region. Even beyond this retention date, you can update the configuration settings to restart exporting the findings in the specific Region.

For more information, see Exporting findings.

Malware Protection issues

I am initiating an On-demand malware scan but it results in a missing required permissions error.

If you receive an error suggesting that you do not have the required permissions to start an On-demand malware scan on an Amazon EC2 instance, verify that you've attached the AWS managed policy: AmazonGuardDutyFullAccess policy to your IAM role.

If you're a member of an AWS organization and still receive the same error, connect with your management account. For more information, see AWS Organizations SCP – Denied access.

I receive an `iam:GetRole` error while working with Malware Protection.

If you receive this error – `Unable to get role: AWSServiceRoleForAmazonGuardDutyMalwareProtection`, it means that you're missing the permission to either enable GuardDuty-initiated malware scan or use On-demand malware scan. Verify that you've attached the AWS managed policy: AmazonGuardDutyFullAccess policy to your IAM role.

I am a GuardDuty administrator account who needs to enable GuardDuty-initiated malware scan but doesn't use AWS managed policy: AmazonGuardDutyFullAccess to manage GuardDuty.

- Configure the IAM role that you use with GuardDuty to have the required permissions to enable GuardDuty-initiated malware scan. For more information on the required permissions, see Creating a service-linked role for Malware Protection.
- Attach the AWS managed policy: AmazonGuardDutyFullAccess to your IAM role. This will help you enable GuardDuty-initiated malware scan for the member accounts.

Runtime Monitoring issues

My AWS Step Functions workflow is failing unexpectedly

If the GuardDuty container contributed to the workflow failure, see Troubleshooting coverage issues. If the issue persists, then to prevent the workflow failure because of the GuardDuty container, perform **one** of the following steps:

- Add the `GuardDutyManaged:false` tag to associated Amazon ECS cluster.
- Disable the automated agent configuration for AWS Fargate (ECS only) at the account level. Add the inclusion tag `GuardDutyManaged:true` to the associated Amazon ECS cluster that you want to continue monitoring with the GuardDuty automated agent.

Troubleshooting out of memory error in Runtime Monitoring (Amazon EC2 support only)

This section provides the troubleshooting steps when you experience out of memory error based on the CPU and memory limit to deploy the GuardDuty security agent manually.

If `systemd` terminates the GuardDuty agent because of the `out-of-memory` issue and you evaluate that providing more memory to the GuardDuty agent is reasonable, you can update the limit.

1. With the root permission, open `/lib/systemd/system/amazon-guardduty-agent.service`.
2. Find `MemoryLimit` and `MemoryMax`, and update both the values.

   ```
   MemoryLimit=256MB
   MemoryMax=256MB
   ```
3. After updating the values, restart the GuardDuty agent by using the following command:

   ```
   sudo systemctl daemon-reload
   sudo systemctl restart amazon-guardduty-agent
   ```
4. Run the following command to view the status:

   ```
   sudo systemctl status amazon-guardduty-agent
   ```

The expected output will show the new memory limit:

```
Main PID: 2540 (amazon-guardduty)
Tasks: 16
Memory: 21.9M (limit: 256.0M)
```

Managing multiple accounts issues

I want to manage multiple accounts but don't have required AWS Organizations management permission.

If you receive this error – `The request failed because you do not have required AWS Organization master permission.`, it means that you're missing the permission to enable GuardDuty-initiated malware scan for multiple accounts in your organization. For more information about providing permission to the management account, see Establishing trusted access to enable GuardDuty-initiated malware scan.

Other troubleshooting issues

If you don't find a scenario suitable to your issue, view the following troubleshooting options:

- For general IAM issues when you access the https://console.aws.amazon.com/guardduty/, see Troubleshooting Amazon GuardDuty identity and access.
- For authentication and authorization issues when you access AWS AWS Console Home, see Troubleshooting IAM.

Regions and endpoints

To view the AWS Regions where Amazon GuardDuty is available, see Amazon GuardDuty endpoints in the *Amazon Web Services General Reference*.

We 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 the supported AWS Regions, its ability to detect activity that involves global services is reduced.

Region-specific feature availability

A list of regional differences to specify the availability of GuardDuty features.

ListFindings and GetFindingsStatistics APIs

The GetFindingsStatistics and ListFindings APIs have a temporary `consoleOnly` flag. When you use any or both of these APIs, the `consoleOnly` flag means that the API can fetch results to a maximum limit of 1000.

GuardDuty features with Region disparity

GuardDuty Malware Protection

GuardDuty supports the Malware Protection feature in the AWS Dedicated Local Zones.

GuardDuty Runtime Monitoring

The Runtime Monitoring feature is presently **not** supported in the Canada West (Calgary) Region.

GuardDuty RDS Protection

The following list specifies the AWS Regions where RDS Protection is presently **not** supported:

- Canada West (Calgary)
- Asia Pacific (Hyderabad)
- Europe (Spain)
- Europe (Zurich)
- Middle East (UAE)

- Israel (Tel Aviv)
- Asia Pacific (Melbourne)

The following APIs in the Amazon GuardDuty API Reference may have regional differences because of the unavailability of some of the data sources or features in previously specified AWS Regions:

- CreateDetector
- UpdateDetector
- UpdateMemberDetectors
- UpdateOrganizationConfiguration
- GetDetector
- GetMemberDetectors
- DescribeOrganizationConfiguration

Amazon EC2 finding types – DefenseEvasion:EC2/UnusualDoHActivity and DefenseEvasion:EC2/UnusualDoTActivity

The following table shows the AWS Regions where GuardDuty is available but these two Amazon EC2 finding types are not yet supported.

AWS Region	Region code
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Jakarta)	ap-southeast-3

AWS GovCloud (US) Regions

For latest information, see Amazon GuardDuty in the *AWS GovCloud (US) User Guide*.

China Regions

For latest information, see Feature availability and implementation differences.

GuardDuty legacy actions and parameters

Amazon GuardDuty has deprecated the some of the API actions and parameters but still supports them. The best practice is to use the new API actions and parameters that replace the legacy options. The following table compares the legacy and new actions and parameters.

Legacy actions/ parameters	New actions/parameters	Comparison
DisassociateFromMasterAccount	DisassociateFromAdministratorAccount	With the same implementation in both the actions, GuardDuty uses the term `Administrator` in `DisassociateFromAdministratorAccount` .
`autoEnable` parameter in DescribeOrganizationConfiguration and UpdateOrganizationConfiguration	`autoEnableOrganizationMembers`	With `autoEnableOrganizationMembers` , the GuardDuty administrator account can audit and enforce GuardDuty for all member accounts to either of the values. Using the APIs, it may take up to 24 hours to update the configuration for all the member accounts. For more information about the possible values of the `autoEnableOrganizationMembers` field, see autoEnableOrganizationMembers
`dataSources` parameter in the APIs listed in GuardDuty API changes in March 2023.	`features`	Starting March 2023, you can configure Malware Protection in Amazon GuardDuty and the new GuardDuty protection plans using `features`. The protection plans launched prior to March 2023, including Malware Protectio

Legacy actions/ parameters	New actions/parameters	Comparison
		n still support configuration using `dataSources` . If you use APIs to configure a protection plan, each API request can either include `dataSources` or `features`, not both.

Document history for Amazon GuardDuty

The following table describes important changes to the documentation since the last release of the *Amazon GuardDuty User Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Support for IPAddressv6	GuardDuty has added IPAddressv6 support for both local and remote IP details. You can use the associated Filter attributes to filter GuardDuty findings or create suppression rules.	April 18, 2024
Updated console experience to configure exporting findings	GuardDuty has updated the console experience to export the findings generated in your AWS accounts, to an Amazon S3 bucket. For more information, see Exporting GuardDuty findings.	April 1, 2024
Updated functionality in Runtime Monitoring	Runtime Monitoring released a new security agent version 1.1.0 for the Amazon EC2 resource. This version supports GuardDuty automated agent configuration in Runtime Monitoring for Amazon EC2 instances. For information about release notes, see GuardDuty security agent for Amazon EC2 instance.	March 28, 2024

General availability of Runtime Monitoring for Amazon EC2 instances	GuardDuty announces general availability(GA) of Runtime Monitoring for Amazon EC2 instances. Now, you have an option to enable automated agent configuration that permits GuardDuty to install and manage the security agent for your Amazon EC2 instances on your behalf. With GuardDuty automated agent, you can also use inclusion or exclusion tags to inform GuardDuty to install and manage the security agent on selected Amazon EC2 instances only. For more information, see How Runtime Monitoring works with Amazon EC2 instances. **List of new finding types released along with this GA** • Execution:Runtime/SuspiciousTool • Execution:Runtime/SuspiciousCommand • DefenseEvasion:Runtime/SuspiciousCommand • DefenseEvasion:Runtime/PtraceAntiDebugging • Execution:Runtime/MaliciousFileExecuted	March 28, 2024

Amazon GuardDuty has updated the Service-linked role (SLR)	Use AWS Systems Manager actions to manage SSM associations on Amazon EC2 instances when you enable GuardDuty Runtime Monitoring with automated agent for Amazon EC2. When GuardDuty automated agent configuration is disabled, GuardDuty considers only those EC2 instances that have an inclusion tag (`GuardDuty Managed :true`).
	- The following list shows the new permissions:
	`"ssm:DescribeAssociation",`
	`"ssm:DeleteAssociation",`
	`"ssm:UpdateAssociation",`
	`"ssm:CreateAssociation",`
	`"ssm:StartAssociationsOnce",`
	`"ssm:AddTagsToResource",`
	`"ssm:CreateAssociation",`
	`"ssm:UpdateAssociation",`
	`"ssm:SendCommand",`
	`"ssm:GetCommandInvocation"`
		March 26, 2024

Updated functionality in Runtime Monitoring	With the latest GuardDuty security agent (add-on) v1.5.0 release for Amazon EKS, Runtime Monitoring now supports configuring specific parameters of your GuardDuty security agent, such as CPU and memory settings, `PriorityClass` settings, and DNS policy settings. For more information, see Configuring GuardDuty security agent (EKS add-on) parameters.	March 7, 2024
Updated functionality in Runtime Monitoring	Runtime Monitoring released a new agent version 1.5.0 for Amazon EKS resources. For information about release notes, see EKS add-on agent release history.	March 7, 2024
Support for Canada West (Calgary)	Amazon GuardDuty is now available in the Canada West (Calgary) Region. Some of the protection plans within GuardDuty might not be available in this Region. For the latest information, see Regions and endpoints.	March 6, 2024

Updated functionality in Runtime Monitoring	The GuardDuty security agent versions 1.0.0 and 1.1.0 for Amazon EKS clusters will no longer be supported starting May 14, 2024. For information about what steps you can take before the end of standard support, see GuardDuty security agent for Amazon EKS clusters.	February 16, 2024
Updated functionality in Runtime Monitoring	Runtime Monitoring supports the latest Kubernetes version 1.29 with the existing security agent version 1.4.1. The support has been available since the launch of this Kubernetes version. For information about supported Kubernetes versions, see Kubernetes versions supported by GuardDuty security agent.	February 16, 2024

Updated functionality in Runtime Monitoring - Regional availability	GuardDuty Runtime Monitoring now supports shared Amazon VPC within the same AWS Organizations. GuardDuty service-linked role (SLR) has a new permission – `organizations:DescribeOrganization` that helps retrieving the organization ID for the shared Amazon VPC account to set the endpoint policy. For information about prerequisites to using a shared Amazon VPC endpoint in Runtime Monitoring, see Support for shared Amazon VPC. This capability is available in all the Regions where GuardDuty supports Runtime Monitoring.	February 12, 2024

Updated functionality in Runtime Monitoring - Regional availability	GuardDuty Runtime Monitoring now supports shared Amazon VPC within the same AWS Organizations. GuardDuty service-linked role (SLR) has a new permission – `organizations:DescribeOrganization` that helps retrieving the organization ID for the shared Amazon VPC account to set the endpoint policy. For information about prerequisites to using a shared Amazon VPC endpoint in Runtime Monitoring, see Support for shared Amazon VPC. Presently, this capability is available in some of the AWS Regions. For more information, see Regions and endpoints.	February 9, 2024
Updated functionality with support for new AWS Regions – Malware Protection	Malware Protection now supports scanning the EBS volumes encrypted with AWS managed keys in the US West (Oregon) Region.	February 6, 2024

Updated functionality with support for new AWS Regions – Malware Protection	Malware Protection now supports scanning the EBS volumes encrypted with AWS managed keys in the following AWS Regions: • Asia Pacific (Singapore) (ap-southeast-1) • Europe (Frankfurt) (eu-central-1) • Asia Pacific (Osaka) (ap-northeast-3) • US East (Ohio) (us-east-2) • Europe (Milan) (eu-south-1) • Asia Pacific (Tokyo) (ap-northeast-1) • Asia Pacific (Seoul) (ap-northeast-2) • Canada (Central) (ca-central-1) • Europe (Ireland) (eu-west-1) • US East (N. Virginia) (us-east-1)	February 5, 2024

Updated functionality in Runtime Monitoring	GuardDuty Runtime Monitoring has released a new GuardDuty security agent version (v1.0.2) for Amazon EC2 instances. This agent version includes support for the latest Amazon ECS AMIs. For more information about agent release history, see GuardDuty security agent for Amazon EC2 instances.	February 2, 2024

Updated functionality with support for new AWS Regions – Malware Protection	Malware Protection now supports scanning the Amazon EBS volumes encrypted with AWS managed keys in the following AWS Regions: • Europe (London) (eu-west-2) • Europe (Stockholm) (eu-north-1) • Asia Pacific (Hong Kong) (ap-east-1) • Africa (Cape Town) (af-south-1) • Middle East (Bahrain) (me-south-1) • Asia Pacific (Hyderabad) (ap-south-2) • Europe (Spain) (eu-south-2) • Asia Pacific (Melbourne) (ap-southeast-4) • Asia Pacific (Sydney) (ap-southeast-2) • Israel (Tel Aviv) (il-central-1)	January 31, 2024

Updated Managing accounts with AWS Organizations	Reorganized the content under Managing accounts with AWS Organizations., added steps to change the delegated GuardDuty administrator account, and updated Understanding the relationship between GuardDuty administrator account and member accounts.	January 30, 2024
Updated functionality with support for new AWS Regions	Malware Protection now supports scanning the EBS volumes encrypted with AWS managed keys in the following AWS Regions: • Asia Pacific (Jakarta) (`ap-southeast-3`) • US West (N. California) (`us-west-1`) • Middle East (UAE) (`me-central-1`) • Europe (Zurich) (`eu-central-2`) • Asia Pacific (Mumbai) (`ap-south-1`) • South America (São Paulo) (`sa-east-1`)	January 29, 2024

Updated functionality in Malware Protection	Malware Protection now supports scanning the EBS volumes encrypted using AWS managed keys. Malware Protection service-linked role (SLR) has two new permissions – `GetSnapshotBlock` and `ListSnapshotBlocks` . These permissions will help GuardDuty fetch the snapshot of an EBS volume (encrypted using AWS managed key) from your AWS account and copy it to the GuardDuty service account before starting the malware scan. Presently, this functionality is available in Europe (Paris) (`eu-west-3`) only. For more information, see Supported volumes for malware scan.	January 25, 2024
Updated functionality in Runtime Monitoring	GuardDuty Runtime Monitoring has released a new GuardDuty security agent version (v1.0.1) with general performance tuning and enhancements. For more information about agent release history, see GuardDuty security agent for Amazon EC2 instances.	January 23, 2024

Updated functionality in Runtime Monitoring	Runtime Monitoring released a new agent version 1.4.1 for Amazon EKS resources. For more information, see EKS add-on agent release history.	January 16, 2024
Runtime Monitoring released new agent v1.4.0 for Amazon EKS resources	Runtime Monitoring released a new agent version 1.4.0 for Amazon EKS resources. For more information, see EKS add-on agent release history.	December 21, 2023

Added S3 and AWS CloudTrail machine learning (ML)-based findings types to the Europe (Zurich) , Europe (Spain), Asia Pacific (Hyderabad), Asia Pacific (Melbourne), and Israel (Tel Aviv)	The following S3 and CloudTrail findings that identify the anomalous behavior using the GuardDuty's anomaly detection machine learning (ML) model are now available in the Europe (Zurich) , Europe (Spain), Asia Pacific (Hyderabad), Asia Pacific (Melbourne), and Israel (Tel Aviv) Regions: - Discovery:S3/AnomalousBehavior - Impact:S3/AnomalousBehavior.Write - Impact:S3/AnomalousBehavior.Delete - Impact:S3/AnomalousBehavior.Permission - Exfiltration:S3/AnomalousBehavior - Exfiltration:IAMUser/AnomalousBehavior - Impact:IAMUser/AnomalousBehavior - CredentialAccess:IAMUser/AnomalousBehavior - DefenseEvasion:IAMUser/AnomalousBehavior - InitialAccess:IAMUser/AnomalousBehavior - Persistence:IAMUser/AnomalousBehavior	December 21, 2023

	• PrivilegeEscalation:IAMUser/AnomalousBehavior • Discovery:IAMUser/AnomalousBehavior	
GuardDuty supports 50,000 member accounts through AWS Organizations	A delegated GuardDuty administrator can now manage a maximum of 50,000 member accounts through AWS Organizations. This also includes a maximum of 5000 member accounts that associated with the GuardDuty administrator account by invitation.	December 20, 2023
GuardDuty Runtime Monitoring support expanded to 19 AWS Regions	Runtime Monitoring is now available in Asia Pacific (Jakarta), Europe (Paris), Asia Pacific (Osaka), Asia Pacific (Seoul), Middle East (Bahrain), Europe (Spain), Asia Pacific (Hyderabad), Asia Pacific (Melbourne), Israel (Tel Aviv), US West (N. California), Europe (London), Asia Pacific (Hong Kong), Europe (Milan), Middle East (UAE), South America (São Paulo), Asia Pacific (Mumbai), Canada (Central), Africa (Cape Town), Europe (Zurich).	December 6, 2023

GuardDuty expands Runtime Monitoring capability	In addition to detecting threats to your Amazon EKS clusters, GuardDuty announces general availability of Runtime Monitoring to detect threats to your Amazon ECS workloads and a preview release to detect threats to your Amazon EC2 instances. For more information about which AWS Regions presently support Runtime Monitoring, see Regions and endpoints.	November 26, 2023

Amazon GuardDuty has updated the Service-linked role (SLR)	GuardDuty has added new permissions to use Amazon ECS actions to manage and retrieve information about the Amazon ECS clusters, and manage the Amazon ECS account setting with `guarddutyActivate`. The actions pertaining to Amazon ECS also retrieve the information about the tags associated with GuardDuty.
	- The following permissions have been added as a part of GuardDuty expanding the Runtime Monitoring capability:
	`"ecs:ListClusters",`
	`"ecs:DescribeClusters",`
	`"ecs:PutAccountSettingDefault"`
		November 26, 2023

Updated the AWS managed policies	GuardDuty added a new permission, `organizations:ListAccounts` to the AmazonGuardDutyFullAccessPolicy and AmazonGuardDutyReadOnlyAccess.	November 16, 2023

GuardDuty released new finding types that use EKS Audit Log Monitoring.	EKS Audit Log Monitoring now supports the following finding types in Asia Pacific (Melbourne)(`ap-southeast-4`). - CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed - PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated - Execution:Kubernetes/AnomalousBehavior.ExecInPod - PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer - PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount - Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed - PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated - Discovery:Kubernetes/AnomalousBehavior.PermissionChecked	November 11, 2023

GuardDuty released new finding types that use EKS Audit Log Monitoring.	EKS Audit Log Monitoring now supports the following finding types in Asia Pacific (Hyderabad) (`ap-south-2`), Europe (Zurich) (`eu-central-2`), and Europe (Spain) (`eu-south-2`) Regions. • CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed • PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated • Execution:Kubernetes/AnomalousBehavior.ExecInPod • PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer • PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount • Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed • PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated • Discovery:Kubernetes/AnomalousBehavior.PermissionChecked	November 10, 2023

GuardDuty released new finding types that use EKS Audit Log Monitoring.	EKS Audit Log Monitoring now supports the following finding types. These finding types are not yet available in Asia Pacific (Hyderabad) (`ap-south-2`), Europe (Zurich) (`eu-central-2`), Europe (Spain) (`eu-south-2`), and Asia Pacific (Melbourne) (`ap-southeast-4`) Regions. - CredentialAccess:Kubernetes/AnomalousBehavior.SecretsAccessed - PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleBindingCreated - Execution:Kubernetes/AnomalousBehavior.ExecInPod - PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!PrivilegedContainer - PrivilegeEscalation:Kubernetes/AnomalousBehavior.WorkloadDeployed!ContainerWithSensitiveMount - Execution:Kubernetes/AnomalousBehavior.WorkloadDeployed - PrivilegeEscalation:Kubernetes/AnomalousBehavior.RoleCreated	November 8, 2023

	• Discovery:Kubernetes/AnomalousBehavior.PermissionChecked	
EKS Runtime Monitoring released new agent v1.3.1	EKS Runtime Monitoring released a new agent version 1.3.1 that includes important security patches and updates.	October 23, 2023
New filter attribute for finding	GuardDuty has added a new criteria to filter the generated findings. DNS request domain suffix provides the second- and top-level domain involved in the activity that prompted GuardDuty to generate the finding.	October 17, 2023
EKS Runtime Monitoring released new agent v1.3.0 that supports Kubernetes version 1.28	EKS Runtime Monitoring released a new agent version 1.3.0 that supports Kubernetes version 1.28. Added support for Ubuntu. For more information, see EKS add-on agent release history.	October 5, 2023

Added S3 and AWS CloudTrail machine learning (ML)-based findings types to the Asia Pacific (Jakarta) and Middle East (UAE) Regions	The following S3 and CloudTrail findings that identify the anomalous behavior using the GuardDuty's anomaly detection machine learning (ML) model are now available in the Asia Pacific (Jakarta) and Middle East (UAE) Regions: • Discovery:S3/AnomalousBehavior • Impact:S3/AnomalousBehavior.Write • Impact:S3/AnomalousBehavior.Delete • Impact:S3/AnomalousBehavior.Permission • Exfiltration:S3/AnomalousBehavior • Exfiltration:IAMUser/AnomalousBehavior • Impact:IAMUser/AnomalousBehavior • CredentialAccess:IAMUser/AnomalousBehavior • DefenseEvasion:IAMUser/AnomalousBehavior • InitialAccess:IAMUser/AnomalousBehavior • Persistence:IAMUser/AnomalousBehavior • PrivilegeEscalation:IAMUser/AnomalousBehavior	September 20, 2023

	• Discovery:IAMUser/AnomalousBehavior	
GuardDuty EKS Runtime Monitoring introduces managing GuardDuty security agent at the cluster level	EKS Runtime Monitoring adds support to manage the GuardDuty security agent for individual EKS clusters to monitor the runtime events from only these selective clusters. EKS Runtime Monitoring extends this capability with the support of tags.	September 13, 2023
GuardDuty Malware Protection extends support to more AWS Regions	Malware Protection is now available in Asia Pacific (Hyderabad), Asia Pacific (Melbourne), Europe (Zurich), and Europe (Spain).	September 11, 2023

GuardDuty is now available in Israel (Tel Aviv) Region	Added Israel (Tel Aviv) Region to the list of AWS Regions where GuardDuty is now available. The following protection plans are also available in the Israel (Tel Aviv) Region: • GuardDuty EKS Protection includes both EKS Audit Log Monitoring and EKS Runtime Monitoring. • GuardDuty Lambda Protection. • GuardDuty Malware Protection. • GuardDuty S3 Protection. For more information about protection plan availability in the Israel (Tel Aviv) Region, see Regions and endpoints.	August 24, 2023
GuardDuty added auto-enable configuration for your organization at protection plan level	Update organization configuration for the protection plans in your Region. Possible configuration options are either enable for all accounts, auto-enable for new accounts, or do not auto-enable for any account in your organization.	August 16, 2023

S3 finding types which identify anomalous behavior using GuardDuty's anomaly detection machine learning (ML) model are now available in Asia Pacific (Osaka)	The following findings types are now available in the Asia Pacific (Osaka) Region: • Discovery:S3/AnomalousBehavior • Impact:S3/AnomalousBehavior.Write • Impact:S3/AnomalousBehavior.Delete • Impact:S3/AnomalousBehavior.Permission • Exfiltration:S3/AnomalousBehavior	August 10, 2023
EKS Runtime Monitoring is now available in Asia Pacific (Melbourne)	EKS Runtime Monitoring within GuardDuty EKS Protection provides runtime threat detection for your Amazon EKS clusters in AWS environment. It is now supported in the Asia Pacific (Melbourne) Region.	August 8, 2023
Updated the list of GuardDuty findings that invoke GuardDuty-initiated malware scan	Certain EKS Runtime Monitoring finding types can now invoke GuardDuty-initiated malware scan in your AWS account.	July 19, 2023

GuardDuty supports 10,000 member accounts through AWS Organizations	A GuardDuty administrator account can now manage a maximum of 10,000 member accounts through AWS Organizations. This also includes a maximum of 5000 member accounts that associated with the GuardDuty administrator account by invitation.	June 29, 2023
EKS Runtime Monitoring announces three new finding types.	EKS Runtime Monitoring supports three new finding types that are based on the process injection technique . The new finding types are DefenseEvasion:Runtime/ProcessInjection.Proc, DefenseEvasion:Runtime/ProcessInjection.Ptrace, and DefenseEvasion:Runtime/ProcessInjection.VirtualMemoryWrite.	June 22, 2023
EKS Runtime Monitoring released new agent v1.2.0 that supports Kubernetes version 1.27	EKS Runtime Monitoring released a new agent version 1.2.0 that also supports ARM64-based instances. Added support for Bottlerocket. For more information, see EKS add-on agent release history.	June 16, 2023

GuardDuty console provides a summarized view of your findings.	The summary dashboard in the GuardDuty console provides an aggregated view of the GuardDuty findings. Presently, the dashboard displays data through various widgets for the last 10,000 findings generated for your account (or member accounts if you're a GuardDuty administrator account) for the current Region.	June 12, 2023
EKS Audit Log Monitoring is now available in Asia Pacific (Hyderabad), Asia Pacific (Melbourne), Europe (Zurich), and Europe (Spain)	Enable EKS Audit Log Monitoring (in EKS Protection) for your accounts to monitor Kubernetes audit logs from your Amazon EKS clusters and analyze them for potentially malicious and suspicious activity.	June 1, 2023
EKS Audit Log Monitoring is now available in Middle East (UAE)	EKS Audit Log Monitoring is now available in Middle East (UAE). Enable EKS Audit Log Monitoring for your accounts to monitor Kubernetes audit logs from your Amazon EKS clusters and analyze them for potentially malicious and suspicious activity.	May 3, 2023

GuardDuty Malware Protection announces On-demand malware scan	Malware Protection helps you detect the potential presence of malware in the Amazon EBS volumes attached to your Amazon EC2 instances and container workloads. It now offers two types of scans – GuardDuty initiated and on-demand. GuardDuty-initiated malware scan initiates an agentless scan in the Amazon EBS volumes automatically only when GuardDuty generates one of the Findings that invoke GuardDuty-initiated malware scan. You can initiate an On-demand malware scan for Amazon EC2 instances in your account by providing the Amazon Resource Name (ARN) associated to that Amazon EC2 instance. For more information about how both the scan types differ, see Malware Protection. • GuardDuty-initiated malware scan • On-demand malware scan	April 27, 2023

GuardDuty announces Lambda Protection	Lambda Protection helps you identify potential security threats in your AWS Lambda functions. - Lambda Protection finding types - Remediating a potentially compromised Lambda function	April 20, 2023
GuardDuty is now available in the Asia Pacific (Melbourne) Region	Added Asia Pacific (Melbourne) to the list of AWS Regions where GuardDuty is available. For information about which features are available in this Region, see Regions and endpoints.	April 19, 2023
GuardDuty added 3 new EC2 findings types	GuardDuty introduces new finding types to detect the use of external DNS resolvers and encrypted DNS technologies. For information about AWS Regions where these finding types are supported, see Regions and endpoints. - DefenseEvasion:EC2/UnusualDNSResolver - DefenseEvasion:EC2/UnusualDoHActivity - DefenseEvasion:EC2/UnusualDoTActivity	April 5, 2023

GuardDuty announces EKS Runtime Monitoring in EKS Protection	EKS Runtime Monitoring within EKS Protection provides runtime threat detection for your Amazon EKS clusters in AWS environment. It uses an Amazon EKS add-on agent (`aws-guardduty-agent`) that collects Runtime events from your EKS workloads. After GuardDuty receives these runtime events, it monitors and analyzes them to identify potential suspicious security threats. For more information, see Finding details and EKS Runtime Monitoring finding types.	March 30, 2023

GuardDuty adds a new functionality – `autoEnableOrganizationMembers`	Amazon GuardDuty adds a new organization configuration option that helps GuardDuty administrator accounts audit and enforce (if required) that GuardDuty is enabled for `ALL` the members of their organization. The best practice now is to use `autoEnableOrganizationMembers` instead of `autoEnable`. `autoEnable` is deprecated but still supported. The following APIs are impacted by this new functionality: - DescribeOrganizationConfiguration - UpdateOrganizationConfiguration - DisassociateMembers - DeleteMembers - DisassociateFromAdministratorAccount - StopMonitoringMembers	March 23, 2023

The RDS Protection feature in Amazon GuardDuty is now generally available	GuardDuty RDS Protection monitors and profiles RDS login activity to identify suspicious login behavior on your Amazon Aurora database instances. For information about which AWS Regions support RDS Protection, see Regions and endpoints.	March 16, 2023
GuardDuty announces feature activation	Historically, the GuardDuty API allowed configuration of both features and data sources, but now, all new GuardDuty protection types will be configured as features and not as data sources. GuardDuty still supports the data sources via API but will not add a new API. Features activation affects the behavior of the APIs used to enable GuardDuty or a protection type within GuardDuty. If you manage your GuardDuty accounts through API, SDK, or CFN template, see GuardDuty API changes in March 2023.	March 16, 2023
GuardDuty Malware Protection is now available in Middle East (UAE) Region	The Malware Protection feature in GuardDuty is supported in the Middle East (UAE) Region. For more information, see Regions and endpoints.	March 13, 2023

Amazon GuardDuty has updated the Service-linked role (SLR)	GuardDuty added the following new permissions to support the upcoming GuardDuty EKS Runtime Monitoring feature.
	- Use Amazon EKS actions to manage and retrieve information about the EKS clusters, and manage EKS add-ons on EKS clusters. The EKS actions also retrieve the information about the tags associated with GuardDuty.
	`"eks:ListClusters",`
	`"eks:DescribeCluster",`
	`"ec2:DescribeVpcEndpointServices",`
	`"ec2:DescribeSecurityGroups"`
		March 8, 2023

Amazon GuardDuty has updated the Service-linked role (SLR)	The GuardDuty SLR has been updated to allow creation of Malware Protection SLR after Malware Protection has been enabled.	February 21, 2023
GuardDuty requires TLS v1.2 or later	To communicate with AWS resources, GuardDuty requires and supports TLS v1.2 or later. For more information, see Data protection and Infrastructure security.	February 14, 2023

GuardDuty is now available in Asia Pacific (Hyderabad) Region	Added Asia Pacific (Hyderabad) Region to the list of AWS Regions where GuardDuty is available. For more information, see Regions and endpoints.	February 14, 2023
Amazon GuardDuty User Guide is aligned with IAM best practices	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	February 10, 2023
GuardDuty is now available in Europe (Spain) Region	Added Europe (Spain) to the list of AWS Regions where GuardDuty is available. For more information, see Regions and endpoints.	February 8, 2023
GuardDuty is now available in Europe (Zurich) Region	Added Europe (Zurich) to the list of AWS Regions where GuardDuty is available . For more information, see Regions and endpoints.	December 12, 2022
Preview release of a new feature – GuardDuty RDS Protection	GuardDuty RDS Protection monitors and profiles RDS login activity to identify suspicious login behavior on your Amazon Aurora database instances. Presently, it is available for a preview release in five AWS Regions. For more information, see Regions and endpoints.	November 30, 2022

GuardDuty is now available in Middle East (UAE) Region	Added Middle East (UAE) to the list of AWS Regions where GuardDuty is available . For more information, see Regions and endpoints.	October 6, 2022

Added content for a new feature – GuardDuty Malware Protection	GuardDuty Malware Protection is an optional enhancement to Amazon GuardDuty. While GuardDuty identifies the resources at risk, Malware Protection detects the malware that may be the source of the compromise. With Malware Protection enabled, whenever GuardDuty detects suspicious behavior on an Amazon EC2 instance or a container workload indicative of malware, GuardDuty Malware Protection initiates an agentless scan on the EBS volumes attached to impacted EC2 instance or container workloads to detect the presence of malware. For information about how Malware Protection works and configuring this feature, see GuardDuty Malware Protection. • For information about Malware Protection findings, see Finding details. • For information about remediating the compromised EC2 instance and a standalone container, see Remediating security issues discovered by GuardDuty.	July 26, 2022

	- For information about auditing CloudWatch logs for malware scans and reasons for skipping a resource during malware scan, see Understanding CloudWatch Logs and skip reasons. - For information about false positive threat detections, see Reporting false positives in GuardDuty Malware Protection.	
Retired one finding type	Exfiltration:S3/ObjectRead.Unusual has been retired.	July 5, 2022

Added new S3 finding types which identify anomalous behavior using GuardDuty's anomaly detection machine learning (ML) model.	Added the following new S3 finding types. These finding types identify if an API request invoked an IAM entity in an anomalous way. The ML model evaluates all API requests in your account and identifies anomalous events that are associated with techniques used by adversaries. To learn more about each of these new findings, see S3 finding types. • Discovery:S3/AnomalousBehavior • Impact:S3/AnomalousBehavior.Write • Impact:S3/AnomalousBehavior.Delete • Impact:S3/AnomalousBehavior.Permission • Exfiltration:S3/AnomalousBehavior	July 5, 2022

Added GuardDuty EKS 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 EKS 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	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 about data sources that 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 Malicious IPCaller 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 account 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 user sign-in 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 continuou s security monitoring and threat detection in this Region. Learn more	March 29, 2018
Creating GuardDuty administr ator account and member accounts through AWS CloudFormation is now supported.	For more information, see `AWS::GuardDuty::ma ster` and `AWS::Guar dDuty::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 account). Learn more	January 25, 2018
Changes in upload and further management of trusted IP lists and threat lists for GuardDuty administrator account and member accounts.	With this release, Users from administrator account GuardDuty accounts can upload and manage trusted IP lists and threat lists. Users from member GuardDuty accounts can't upload and manage lists. Trusted IP lists and threat lists that are uploaded by the administrator account account are imposed on GuardDuty functionality in its member accounts. Learn more	January 25, 2018

Earlier updates

Change	Description	Date
Initial publication	Initial publication of the Amazon GuardDuty User Guide.	November 28, 2017