User Guide

# AWS Security Agent

# AWS Security Agent: User Guide

# Table of Contents

# What is AWS Security Agent?

AWS Security Agent is a frontier agent that proactively secures your applications throughout the development lifecycle. It conducts automated security reviews tailored to your organizational requirements and delivers context-aware penetration testing on demand. By continuously validating security from design to deployment, AWS Security Agent helps prevent vulnerabilities early across all your environments.

Security teams define organizational security requirements once in the AWS Console: approved authorization libraries, logging standards, and data access policies. AWS Security Agent automatically enforces these security requirements throughout development, evaluating architectural documents and code against your standards and providing specific guidance when it detects violations. This delivers consistent security enforcement for design and code reviews across all teams.

For deployment validation, AWS Security Agent transforms penetration testing from a periodic bottleneck into an on-demand capability. Security teams provide target URLs, authentication details, source code and application documentation. AWS Security Agent develops deep application understanding and executes sophisticated attack chains to discover and validate vulnerabilities, enabling teams to test whenever needed.

## Key capabilities

AWS Security Agent provides comprehensive security capabilities spanning the entire development lifecycle.

### Design security review

AWS Security Agent shifts security left by providing real-time security feedback on design documents and assessing compliance with organizational security requirements before any code is written. Security teams upload documents through the web application and receive remediation guidance to prioritize findings, accelerating time-consuming manual reviews into focused analysis. By proactively embedding your security standards into every design review, you reduce late-stage architectural rework and keep pace with multiple development teams.

## Code security review

AWS Security Agent proactively secures applications by analyzing pull requests against your organizational security requirements and common vulnerabilities like missing input validation and SQL injection risks. Developers receive remediation guidance directly in their GitHub workflow, while security teams configure which repositories to monitor and intervene on critical issues. This embeds security expertise across all repositories, reducing security-related delays in the development pipeline and scaling evaluation across all codebases.

## Penetration testing

AWS Security Agent delivers on-demand penetration testing by deploying specialized AI agents to discover, validate, report and remediate security vulnerabilities through tailored multi-step attack scenarios. The agent understands your application's context by analyzing source code and documentation to identify and exploit vulnerabilities that automated security scanning tools cannot find. It documents findings with impact analysis, reproducible attack paths, and creates pull requests with ready-to-implement code fixes, transforming periodic assessments into continuous validation that scales across all applications rather than being limited to only critical ones.

# Benefits

## On-demand accessibility

AWS Security Agent provides immediate access to security expertise. Development teams can run design reviews, code analyses, and penetration tests on-demand whenever needed, matching the pace of modern development cycles and enabling proactive security at every stage.

## Validate organizational requirements automatically

Define your organization's security requirements once in the AWS Console. AWS Security Agent automatically validates these requirements across all applications during every design and code security review, ensuring teams address your specific requirements rather than generic checklists.

## Scale security expertise

AWS Security Agent scales security reviews to match development velocity by automating enforcement of organizational security requirements. Configure requirements centrally, conduct

comprehensive reviews with findings analysis, and manage penetration testing scopes across your organization through the AWS Console.

## Actionable fixes for confirmed vulnerabilities

AWS Security Agent validates security findings found during penetration testing through proof-based exploitation, delivering reproducible exploit paths, comprehensive impact analysis, and creates pull requests with ready-to-implement fixes in developer-friendly language. This helps teams focus on legitimate high-impact security risks without wasting time on false positives.

## Multi and hybrid cloud support

AWS Security Agent operates across AWS, on-premise, hybrid, multicloud, and SaaS environments, ensuring consistent security guidance and testing regardless of your infrastructure or platform choices.

# How AWS Security Agent works

AWS Security Agent operates across three interfaces to provide proactive application security throughout the development lifecycle. Security configurations are managed in the AWS Management Console, security reviews are conducted in the Security Agent Web Application, and automated code and security finding remediation occur directly in code repository platforms like GitHub.

## Overview

AWS Security Agent consists of three main components:

- **AWS Management Console** - Configure Agent Spaces, define security requirements, configure penetration testing, connect code repositories, and manage user access
- **Security Agent Web Application** - Conduct design reviews, execute penetration tests, and review security findings within your assigned Agent Spaces
- **Code Repository Integration** - Receive automated code reviews on pull requests and penetration test remediation pull requests. Currently AWS Security Agent supports connecting to GitHub.

### Console configuration

Administrators configure AWS Security Agent through the AWS Management Console.

**Agent Spaces** - When you create your first Agent Space in the AWS Management Console, AWS Security Agent creates the Web Application for your account. Each Agent Space you create represents a distinct application or project you want to secure. In the Web Application, users select which Agent Space to work in when conducting security assessments.

**Security requirements** - Define organizational security requirements centrally in the Console, such as approved authorization libraries, logging standards, and data access policies. These requirements apply across all Agent Spaces, and AWS Security Agent automatically validates them during design and code reviews.

**Penetration testing configuration** - Configure penetration testing capabilities for each Agent Space by:

- Verifying target domains for testing through DNS or HTTP verification

- Configuring VPC access for testing private applications
- Setting up CloudWatch logging for penetration test runs
- Configuring AWS Secrets Manager or Lambda functions for test credentials
- Specifying S3 buckets for additional application context

**Integrations** - Connect GitHub repositories to each Agent Space to enable three key capabilities:

- Provide application context for more accurate penetration testing
- Enable automated code review on pull requests
- Enable penetration test finding remediation through automated pull requests

**User access** - Manage how users access the Security Agent Web Application. If you've enabled IAM Identity Center (SSO), assign users either in IAM Identity Center or the AWS Security Agent console to provide direct SSO access to the Web Application. If you're using IAM-only access, users with AWS Console access can launch the Web Application through the admin access link in the Console for any Agent Space.

## Web Application activities

Users access the Security Agent Web Application to conduct security assessments within their assigned Agent Spaces.

**Select Agent Space** - When logging into the Web Application, users select which Agent Space to work in. Users can only see and access Agent Spaces they've been assigned to.

**Design reviews** - Upload design documents and architecture specifications for analysis against organizational security requirements. Review findings with remediation guidance.

**Penetration tests** - Configure and execute penetration tests by providing target URLs, authentication details, and documentation. AWS Security Agent performs autonomous testing to discover exploitable vulnerabilities through multi-step attack scenarios.

**Review findings** - Examine detailed security findings from design reviews and penetration tests, including impact analysis, reproducible attack paths, and remediation guidance.

**Validate fixes** - Re-run security assessments after implementing remediations to verify vulnerabilities have been addressed.

# GitHub integration

AWS Security Agent integrates directly with GitHub to provide automated security feedback in developers' workflows.

**Automated code review** - After administrators install the AWS Security Agent GitHub App and enable code review for an Agent Space in the Console, AWS Security Agent automatically analyzes pull requests in connected repositories. Developers receive security findings and remediation guidance directly in pull request comments, validating code changes against organizational security requirements and common vulnerabilities.

**Penetration test remediation** - When administrators enable finding remediation in the Console, users can request automatic remediation for penetration test findings from the Web Application. AWS Security Agent opens a pull request to the associated GitHub repository with code fixes to address the vulnerability.

# Quickstart: Run a penetration test

This quickstart walks you through running your first penetration test (pentest) with AWS Security Agent. AWS Security Agent tests your deployed application and identifies security vulnerabilities with detailed findings.

> **Note**
>
> You need access to AWS Management console to setup a new penetration test

## Step 1: Set up AWS Security Agent in the AWS console

1. Navigate to AWS Security Agent in the AWS Management Console.
2. Select **Set up AWS Security Agent**
3. Create an agent space. An agent space can be used by multiple users and should be specific for every application you want to test. Enter a name and description for your first agent space. This name appears to users in the web application. The name of the agent space should be based on the application you want to penetration test.
4. Select **IAM-only access** under *User access configuration*
   - This quickstart does not cover enabling single sign-on (SSO) with IAM Identity Center. This allows users to directly access the AWS Security Agent web application, from the AWS Console.
   - If you want to enable users without AWS Management Console Access to perform tasks such as starting a penetration test or design review, you should enable the IAM Identity Center integration.
5. Click **Set up AWS Security Agent**

> **Note**
>
> When you choose Set up, AWS Security Agent will create your Agent Space, and establish a web application where your users can carry out design reviews and penetration tests.

# Step 2: Enable and configure penetration testing

> **Note**
>
> In the AWS console, you define the scope of what can be tested. Users then run specific penetration tests within that scope from the AWS Security Agent web application.

1. From the left sidebar, select **Agent Spaces** and then select the Agent Space you created in Step 1.
2. From the header, select **Enable penetration test** to enable this capability.
3. **Step 1 — Configure domain**: Enter the target domain you want to test and select a verification method (**DNS_TXT** or **HTTP_ROUTE**). The domain should be live and host the application you want to penetration test. Choose **Next** to proceed.
4. **Step 2 — Verify domains**: Verify ownership of each domain in the **Target domains** table:
   - For Route 53 domains in the same AWS account: select the domain and choose **One-click verification**. AWS Security Agent creates the DNS record and completes verification automatically.
   - For other DNS providers: copy the verification token, add the TXT record with your DNS registrar, then select the domain and choose **Verify**.
   - AWS Security Agent can only run penetration tests against verified domains.
5. **Step 3 — Configure additional capabilities (optional)**: Configure optional resources such as VPCs, CloudWatch logs, and credentials. The **Service access** section is pre-configured — AWS Security Agent automatically creates a service role with the required permissions unless you want to use an existing IAM role.

# Step 3: Connect to GitHub (optional)

> **Note**
>
> This step is optional, however we recommend connecting to your GitHub account to give AWS Security Agent access to the source code for your application. This helps the Security Agent understand your application context and improve penetration testing coverage.

1. Once you have completed the pentest setup, you will see a banner with an option to connect GitHub for penetration testing, click **Add** in the right side of the banner.
2. Click **Create new registration**
3. Select **GitHub** and then **Next**
4. Click **Install and authorize**. You'll be redirected to GitHub to complete the installation.
   a. Select the *GitHub User* or *GitHub Organization* that owns the repository you want to test.
   b. Select either **All repositories** or **Only select repositories**. AWS suggests installing AWS Security Agent on all repositories, and then creating a unique agent space for each repository you want to test.
   c. Select **Install & Authorize** and complete GitHub authentication.
5. Define the **Registration Name** and confirm the **account type** matches where you installed the GitHub application.
6. Click **Next**
7. Select the repositories you want to be associated for penetration testing. This allows the web application users to associate these repositories to a penetration test, when they create a new pentest.
8. Click **Next**
9. If you want to enable automatic code remediation, enable **Pentest remediation enabled** on the repositories you want to allow AWS Security Agent to create pull requests with ready-to-implement code fix for pentest findings.
10. Click **Connect**

## Step 4: Run a penetration test

> **Note**
>
> You can create and run a penetration test only in the AWS Security Agent web application.

1. Select the **Web app** tab and then **Admin access** to launch the AWS Security Agent Web Application with administrator privileges. This will only work if you had setup your agent using **IAM-only access** under *User access configuration*. Alternatively, you will need to add users and create a login.
2. In the left sidebar, click **Penetration Test**, then select **Create your first penetration test**.

3. Define the penetration test details:
   a. Select the domain you want to test or specify one or more paths. You can only test verified domains.
   b. If your application needs to access URLs that are outside of your target domain, add them to the **Accessible URLs** field.

      > **Note**
      >
      > Add accessible domains for third-party services (such as Okta, Auth0, Stripe) that are outside your target domain. This is required so AWS Security Agent can access these URLs for login and navigation during testing. AWS Security Agent does NOT penetration test these domains—they are used solely for access purposes.

   c. Select the IAM role and log group AWS Security Agent should use to store logs. If you do not select a log group, AWS Security agent will create a log group at the start of the pentest run.
   d. Select **Enable automatic code remediation** to allow AWS Security Agent to automatically create a pull request with ready-to-implement code fix for all the pentest findings.
   e. Click **Next**.
4. (Optional) If your application requires a login, then input the credentials directly into the web application. Define how AWS Security Agent should authenticate to your application. Provide authentication instructions into **Agent Space login prompt**, then click **Next**.
5. (Optional) Provide additional resources to help test your application. You can upload files such as design documents, threat model, API specifications or other documents that are helpful to understand the application context.
6. Click **Create and execute**. You'll be redirected to the penetration test detail screen.
   - To save the configuration for future use without running it immediately, click **Create penetration test** instead.

## Step 5: Review penetration test findings

1. The penetration test can take up to several hours to complete.
2. Once complete, review the details of the pentest on the Pentest overview, logs and findings screens.

# AWS Security Agent capabilities

AWS Security Agent provides three core security capabilities throughout your development lifecycle:

- **Design security review** — Reviews design and architecture documents to identify security risks. Upload documents through the web application, and the service analyzes them against your organizational security requirements. AWS Security Agent evaluates compliance with your defined security requirements such as approved authorization libraries, logging standards, and data access policies. This helps you catch insecure designs and policy violations early in the development process.
- **Code security review** — Analyzes code in your code repositories to identify security vulnerabilities and violations of organizational security requirements across languages, frameworks, and architectures. After connecting AWS Security Agent to your repositories, you can enable it to automatically review pull requests against your defined security requirements and provide specific remediation guidance directly in your code repository platform. This ensures consistent enforcement of your security policies across all development teams.
- **On-demand penetration testing** — Discovers, validates, reports and remediates security vulnerabilities in live web applications and APIs through tailored multi-step attack scenarios. Configure the service to create a pentest through the web application by specifying testing scope, authentication details, and resources. AWS Security Agent develops application context from provided source code and documentation and executes sophisticated attack chains to identify exploitable vulnerabilities that static analysis and conventional tools miss. It also provides ready-to-implement code fixes and creates pull requests directly into your code repository, enabling you to resolve vulnerabilities even faster.

# Configure AWS Security Agent

As an administrator, you set up AWS Security Agent in the AWS Management Console and configure Agent Spaces that users access through the AWS Security Agent web application. Each Agent Space represents a distinct environment with specific permissions and resources.

AWS recommends creating a unique Agent Space for each application you want to test. For example, if you have two internal projects—a billing application and a task tracking application—you should create two separate Agent Spaces.

## Example configuration

Consider an administrator setting up an agent space to assess the security of an internal billing application. The administrator would:

- Verify the domain (such as `beta.billing.example.com`)
- Connect to GitHub and enable Code Review
- Configure network access by assigning an appropriate VPC, Subnet, and Security Group for penetration testing

When users initiate a penetration test or design review, they can select from these pre-configured resources, working within the guardrails you've defined while maintaining flexibility for their specific assessment needs.

## Set up AWS Security Agent

Configure AWS Security Agent for your organization by creating your first Agent Space and establishing how users will access the web application.

This initial setup enables administrators to configure security capabilities in the AWS Console and provides users with access to design review and penetration testing through the Security Agent Web Application. After this one-time setup, you can create additional Agent Spaces with a simplified process.

# Overview

## Understanding Agent Spaces

An Agent Space is a dedicated workspace for securing a specific application or project. It contains all security reviews, optionally connected GitHub repositories, penetration test configurations, results, and findings for that application.

Agent Spaces help you organize security work by keeping each application's security assessments separate, allowing teams to focus on their specific application. We recommend creating one Agent Space per application or project to maintain clear boundaries.

Security requirements are defined at the organization level and apply across all Agent Spaces, while each Agent Space maintains its own design documents, code repositories, penetration testing configurations, penetration testing results and security findings.

## What's included in an Agent Space

Each Agent Space contains:

- Optionally connected GitHub repositories associated with the application or project
- Previous design reviews for the application
- Penetration testing configurations and boundaries specific to the application
- Penetration testing test results and security findings

## What you'll configure during setup

During this first-time setup, you'll make organizational decisions that apply to all Agent Spaces:

- **Access method** - Choose how users will access the Security Agent Web Application (IAM Identity Center SSO or IAM-only access through the AWS Console)
- **Permissions** - Configure the IAM role that the web application uses to access AWS services
- **First Agent Space** - Create your initial Agent Space with a name and description

> **Note**
>
> After completing this setup, creating additional Agent Spaces is simpler - just provide a name and description. See the section called "Create an Agent Space" for details on creating subsequent Agent Spaces.

## Prerequisites

Before you begin, ensure you have:

- Permissions to create and manage AWS Security Agent resources
- Understanding of your organization's identity management requirements
- (Optional) AWS account with permissions to create IAM roles and configure IAM Identity Center (if using SSO access)
- (Optional) Existing IAM role if you want to use a custom permissions configuration

## Step 1: Create your first Agent Space

Define the basic properties of your first Agent Space that will be displayed to users in the web application.

1. On the **AWS Security Agent** console page, click **Set up Security Agent**.
2. In the **Agent Space name** field, enter a name for your Agent Space.

> **Note**
>
> The Agent Space name is displayed to users in the web application and helps identify which application or project this space represents.

3. (Optional) In the **Description** field, provide a description that assists in distinguishing the Agent Space purpose.

> **Tip**
>
> The description helps distinguish the Agent Space's purpose. We recommend describing the specific application or project this Agent Space will secure, such as "Customer portal web application" or "Payment processing microservices" or "Internal analytics platform."

## Step 2: Choose your access method

Select how users will access the Security Agent Web Application. You'll choose between enabling SSO access through IAM Identity Center or providing access through the AWS Console.

> **Note**
>
> If you choose IAM-only access and later want to use IAM Identity Center, you'll need to delete your AWS Security Agent setup and complete the setup process again.

1. In the **Access method** section, select one of the following options:
   - **IAM Identity Center (SSO)** - Enable SSO access for your team through IAM Identity Center. This option is recommended for teams that need centralized user management and want users to access the web application directly without going through the AWS Console.
   - **IAM-only access** - Provide access through an admin access link in the AWS Console. This option is simpler to set up and suitable for teams that prefer console-based access or don't require SSO capabilities.
2. If you selected **IAM Identity Center (SSO)**, continue to Step 2a below.
3. If you selected **IAM-only access**, skip to Step 3.

### Step 2a: Configure IAM Identity Center (SSO access only)

Complete these steps only if you selected IAM Identity Center (SSO) as your access method.

1. In the **Connect to IAM Identity Center** section, review the **AWS Region**.

> **Important**
>
> Your application must be configured in the same Region where you enable IAM Identity Center. The displayed Region is where your Agent Space will be created. IAM Identity Center must be enabled in the same Region where you create your Agent Space.

2. In the **IAM Identity Center** section, choose one of the following:
   - Click **Create account instance** to create a new IAM Identity Center account instance
   - If an organization instance already exists in the same Region, AWS Security Agent will automatically connect to it

   > **Note**
   >
   > After AWS Security Agent is connected to IAM Identity Center, you can manage access by assigning users in IAM Identity Center to the application.

3. If you are connecting Active Directory or an external identity provider, review the informational alert.

> **Important**
>
> If you're already managing users in Active Directory or another identity source and plan to connect that identity source to IAM Identity Center, cancel setup and go to the IAM Identity Center console first. Choose the identity source you want to connect before setting up AWS Security Agent. Changing identity sources later might remove existing user assignments.

## Step 3: Configure permissions (Optional)

Configure the IAM role that your Security Agent Web Application uses to access AWS services, APIs, and accounts.

1. Locate the **Permissions configuration - optional** section.
2. If the section is collapsed, click to expand it.
3. Select one of the following options:

- **Create default role** - AWS Security Agent automatically creates a new IAM role with the necessary permissions for the web application
- **Use another role** - Select an existing IAM role from the available options

4. Review the role description:

> **Note**
>
> A default IAM role will be created for your web application to access other AWS services, APIs, and accounts. The role will be granted permissions required for security assessment operations.

## Step 4: Complete setup

After configuring all required settings, complete the setup to create your first Agent Space and establish the Security Agent Web Application.

1. Review all configuration sections to ensure accuracy.
2. Click **Set up** at the bottom of the page.
3. AWS Security Agent will create your Agent Space, establish the web application, and configure the necessary AWS resources.

> **Note**
>
> After initial setup, you'll have the option to configure capabilities including penetration testing boundaries, security requirements, and GitHub integration.

## Next steps

After setting up AWS Security Agent:

- Define security requirements for your organization
- Configure penetration testing capabilities including domain verification and VPC access for your Agent Space
- Connect GitHub repositories for code review and penetration testing context

- **(If using IAM Identity Center)** Assign users to the Agent Space in IAM Identity Center
- **(If using IAM-only access)** Launch the web application through the admin access link in the AWS Console
- Create additional Agent Spaces for other applications (see <>)

# Create an Agent Space

Create Agent Spaces to secure applications or projects in your organization. Each Agent Space provides a workspace for security assessments, findings, and configurations specific to that application or project and can be accessed by multiple users.

This procedure assumes you have already completed the initial AWS Security Agent setup. If you haven't set up AWS Security Agent yet, see the section called "Set up AWS Security Agent".

## Overview

### Understanding Agent Spaces

An Agent Space is a dedicated workspace for securing a specific application or project. It contains all security reviews, optionally connected code repositories, penetration test configurations, results, and findings for that application.

Agent Spaces help you organize security work by keeping each application's security assessments separate, allowing teams to focus on their specific application. We recommend creating one Agent Space per application or project to maintain clear boundaries.

For a comprehensive explanation of Agent Spaces and how they fit into your organization's security structure, see *Understand the resource hierarchy and lifecycle*.

### What's included in an Agent Space

Each Agent Space contains:

- Optionally connected code repositories associated with the application or project
- Previous design reviews for the application or project
- Penetration testing configurations and boundaries specific to the application
- Penetration testing test results and security findings

# Create an Agent Space

Create a new Agent Space for an application or project you want to secure.

1. In the AWS Security Agent console, navigate to the **Agent Spaces** page.
2. Click **Create Agent Space**.
3. In the **Agent Space name** field, enter a name for your Agent Space.

> **Note**
>
> The Agent Space name is displayed to users in the web application and helps identify which application or project this space represents.

4. (Optional) In the **Description** field, provide a description that assists in distinguishing the Agent Space purpose.

> **Tip**
>
> The description helps distinguish the Agent Space's purpose. We recommend describing the specific application or project this Agent Space will secure, such as "Customer portal web application" or "Payment processing microservices" or "Internal analytics platform."

5. Click **Create**.

> **Note**
>
> AWS Security Agent will create your Agent Space. You can now configure capabilities and connect resources specific to this application.

## Next steps

After creating your Agent Space:

- Connect GitHub repositories for code review and penetration testing context
- Enable code review capability for connected repositories (see the section called "Enable code review capability for a GitHub repository")
- Configure penetration testing capabilities including domain verification

- **(If using IAM Identity Center)** Assign users to this Agent Space under the Web App section of the Agent Space page. (see the section called "Grant users access to the AWS Security Agent web app")
- **(If using IAM-only access)** Users with console access can launch the web application through the admin access link for this Agent Space

# Enable penetration test

Configure AWS Security Agent to run autonomous penetration tests on your applications. This setup enables AWS Security Agent to access your AWS resources, verify domain ownership, and perform comprehensive security testing that identifies exploitable vulnerabilities in your web applications and APIs.

Enabling penetration testing allows AWS Security Agent to continuously validate your application security without the delays of manual testing. By configuring the necessary AWS integrations, you ensure AWS Security Agent can test both public and private applications, log findings to CloudWatch, and access credentials for authenticated testing.

In this procedure, you'll configure target domains, optionally set up VPC, CloudWatch logging, credentials storage, and Lambda functions for testing, configure S3 integrations for providing additional context, and set up service access through IAM roles.

## Prerequisites

Before you begin, ensure you have:

- AWS account with administrative permissions to create IAM roles
- Domain ownership verification capability (DNS or HTTP record modification)
- Target domains or applications to test
- (Optional) VPC configuration details if testing private applications
- (Optional) S3 bucket if providing additional artifacts to AWS Security Agent

## Step 1: Configure domain

In the first step of the wizard, specify the target domains you want to test and how AWS Security Agent should verify ownership.

1. In the **Target domains** section, enter your domain in the **Domain** field.
2. Select a **Verification method**:
   - **DNS_TXT** – Prove domain ownership by adding a TXT record to your domain's DNS configuration.
   - **HTTP_ROUTE** – Prove domain ownership by hosting a verification file at a specific URL on your domain.
   - For more information, see the section called "Enable an application domain for penetration testing".
3. Choose **Add another domain** to add additional domains (up to 5 total).
4. Choose **Next** to proceed to domain verification.

## Step 2: Verify domains

In the second step of the wizard, verify ownership of each domain you configured. AWS Security Agent requires verified ownership before it can perform penetration testing.

1. Review the domains listed in the **Target domains** table.
2. For each domain, select it and trigger verification based on your chosen method:
   - **Route 53 domains (same AWS account)**: Choose **One-click verification**. AWS Security Agent automatically creates the DNS record and completes verification.
   - **DNS TXT (other DNS providers)**: Copy the verification token, add the TXT record with your DNS registrar, then select the domain and choose **Verify**.
   - **HTTP route**: Place the verification token at the required route path on your web server, then select the domain and choose **Verify**. For details, see the section called "Enable an application domain for penetration testing".
3. Choose **Next** to proceed to optional configuration.

> **Note**
>
> Sub-domains of a verified domain do not require individual verification. For private domains inside a VPC, you can proceed even if the domain verification status is UNREACHABLE. AWS Security Agent will attempt domain verification for private endpoints at the start of each pentest run.

# Step 3: (Optional) Configure additional capabilities

The third step of the wizard lets you configure optional AWS resources to expand your penetration testing scope. All sections in this step are optional and collapsed by default.

## (Optional) Configure VPC settings

If you plan to test private target domains hosted within a VPC, configure VPC settings for AWS Security Agent. This section is optional and collapsed by default.

1. Expand the **VPCs** section.
2. In the **VPC** dropdown, select the VPC that hosts your private target domains.
3. In the **Subnet** dropdown, select the VPC subnets that AWS Security Agent should use:

> **Tip**
>
> For high availability, select multiple subnets from multiple Availability Zones. Ensure your subnets include a NAT gateway for outbound connectivity.

4. In the **Security group** dropdown, select the VPC security groups that AWS Security Agent should use:

> **Important**
>
> Ensure your security groups allow outbound connections for AWS Security Agent to perform penetration testing.

## (Optional) Configure CloudWatch logging

Configure CloudWatch to store logs from your penetration test runs. This section is optional and collapsed by default.

1. Expand the **CloudWatch logs** section.
2. In the **Log Groups** dropdown, select existing CloudWatch log groups
3. If not selecting any log group, AWS Security Agent will create a log group named `/aws/securityagent/<agent name>/<pentest id>` with appropriate permissions.

**Note**

Ensure your IAM role has permissions to write to the selected CloudWatch log group.

## (Optional) Configure Secrets for test credentials

If your application requires authentication credentials for testing, AWS Security Agent can securely retrieve them from AWS Secrets Manager. This section is optional and collapsed by default.

1. Expand the **Secrets** section.
2. Select the AWS Secrets Manager secrets that contain credentials for your application.
3. When configuring a penetration test in the web application, you can reference these secrets for authenticated testing.

**Important**

Credentials are encrypted and stored in AWS Secrets Manager. Ensure your IAM role has permissions to access Secrets Manager for AWS Security Agent to use these credentials during testing.

## (Optional) Configure Lambda functions for test credentials

Configure Lambda functions that can provide credentials for your application during testing. This section is optional and collapsed by default.

1. Expand the **Lambda functions** section.
2. Select the Lambda functions that can provide authentication credentials for your application.
3. AWS Security Agent will invoke these functions during penetration tests to obtain credentials dynamically.

> **Note**
>
> Ensure your IAM role has permissions to invoke the specified Lambda functions. Lambda functions should return credentials in the expected format for AWS Security Agent to use during testing.

## (Optional) Configure S3 bucket

Provide S3 bucket details if you plan to upload documents or artifacts to provide as input to AWS Security Agent. This section is optional and collapsed by default.

1. Expand the **S3 buckets** section.
2. In the **Bucket** field, enter or search for your S3 bucket name.

> **Note**
>
> You can also connect GitHub repositories later or upload files directly in the web application. Information provided can ensure thorough coverage, reduce false positives, and deliver actionable results.

## (Optional) Configure service access

AWS Security Agent requires an IAM role to access your AWS resources (VPC, CloudWatch log groups, Secrets Manager, Lambda functions, etc.) for penetration testing. This section is optional and collapsed by default.

1. Expand the **Service access** section.
2. By default, AWS Security Agent uses a default IAM role with the required permissions for penetration testing.
3. To customize the IAM role, select one of the following options:
   a. **Create default role** – AWS Security Agent automatically creates a new IAM role with the necessary permissions
   b. **Use an existing service role** – Select an existing IAM role from the dropdown menu
4. If using an existing role:

a. Click the dropdown menu under **Choose an existing role**

b. Select your IAM role from the list

c. Click the refresh icon to update the list if needed

> **Note**
>
> The default IAM role includes permissions for accessing VPC resources, CloudWatch log groups, Secrets Manager, Lambda functions, and other services required for penetration testing. It is recommended to use the default IAM role unless you have specific security requirements.

## Step 4: Save and enable penetration testing

After configuring all required settings, enable penetration testing for your AWS Security Agent agent.

1. Review all configuration sections to ensure accuracy.
2. Click **Save** at the bottom of the page.
3. AWS Security Agent will validate your configuration and create the necessary AWS resources.

## Next steps

After enabling penetration testing:

- Configure penetration test scopes in the AWS Security Agent web application
- Set up notification preferences for test findings
- Review and respond to penetration test findings as they are discovered
- (Optional) Configure additional repositories for findings remediation

For more information about running and managing penetration tests, see the AWS Security Agent web application documentation.

# Enable code review capability for a GitHub repository

Configure AWS Security Agent to automatically review pull requests in your connected GitHub repositories. Code review analyzes code changes against your organizational security requirements and common security vulnerabilities for consistent enforcement.

AWS Security Agent automatically comments on pull requests with security findings and remediation guidance, helping developers address issues directly in their GitHub workflow.

In this procedure, you'll connect repositories to your Agent Space, select which repositories have code review enabled, configure code review settings, and activate the capability.

## Prerequisites

Before you begin, ensure you have:

- Installed and authorized the AWS Security Agent GitHub App for your GitHub organization (see the section called "Connect AWS Security Agent to GitHub repositories")
- Appropriate permissions to configure code review settings for your Agent Space
- (Optional) At least one custom security requirement enabled if you plan to use security requirement validation (see the section called "Manage security requirements")

## Step 1: Access the code review configuration

Navigate to the code review configuration for your Agent Space.

1. In the AWS Security Agent console, select your Agent Space.
2. Choose **Enable code review** from the capabilities menu.

## Step 2: Connect repositories to your Agent Space

Select which repositories from your authorized GitHub organization or user account to connect to this Agent Space.

1. From the list select the registered **GitHub organization or user** that you authorized.

> **Note**
>
> If you registered multiple GitHub organizations or users, you can select one and connect repositories.

2. Select repositories to connect:
   - Browse the list of available repositories
   - Select the checkbox next to each repository you want to connect
3. Click **Add repositories** to connect the selected repositories to your Agent Space.
4. The connected repositories will appear in your Agent Space's repository list.

## Step 3: Select repositories for code review

Review your connected GitHub repositories and enable code review for the repositories you want AWS Security Agent to monitor.

1. In the **Connected GitHub repositories** section, you'll see a list of all repositories connected to your Agent Space.
2. Review the repository list and identify which repositories should have code review enabled.

> **Note**
>
> Code review is only available for private repositories.

3. For each repository you want to enable:
   - Locate the repository in the table
   - Toggle the **Enable** switch in the **Code review** column to the on position

> **Tip**
>
> You can use the search field to quickly find specific repositories by name.

## Step 4: Configure code review settings

Configure the types of security issues AWS Security Agent analyzes in pull requests. This setting applies to all repositories with code review enabled in this Agent Space and can be modified at any time.

1. In the **Code review settings** section, select one of the following options:
   - **Security requirement validation** – Validate whether code changes comply with the custom security requirements you've enabled. This is the default setting.
   - **Security vulnerability findings** – Identify common security vulnerabilities in code changes.
   - **Security requirements and vulnerability findings** – Analyze code changes for both compliance with your organization's custom security requirements and common security vulnerabilities.

> **Note**
>
> When security requirement validation is enabled, AWS Security Agent only checks code changes against your enabled custom security requirements, not AWS managed requirements. Custom security requirements are organization-specific policies you define and enable. If you enable security requirement validation but do not have at least one custom security requirement enabled, AWS Security Agent will not perform code reviews. For more information about security requirements, see the section called "Manage security requirements".

## Step 5: Enable code review

After selecting your repositories and configuring your settings, activate code review capability for your Agent Space.

1. Review your repository selections and code review settings to ensure accuracy.
2. Click **Enable** at the bottom of the page.
3. AWS Security Agent will activate code review for the selected repositories with your configured settings.

> **Note**
>
> You can modify code review settings and which repositories have code review enabled at any time by returning to this configuration page.

### Next steps

After enabling code review:

- AWS Security Agent will automatically analyze pull requests in enabled repositories based on your configured code review settings
- Security findings will be posted as comments on pull requests with specific remediation guidance
- Review and respond to security findings as they are discovered
- Adjust code review settings or which repositories have code review enabled as your needs change

## Manage security requirements

Configure security requirements that AWS Security Agent uses to analyze your applications during design reviews and code reviews. You can enable AWS-managed requirements, customize them to fit your organization's standards, or create custom requirements from scratch.

### Overview

Security requirements define the security standards and policies that AWS Security Agent enforces when analyzing your applications. When you conduct design reviews or code reviews, AWS Security Agent evaluates your application against these requirements and identifies potential compliance issues.

AWS Security Agent provides two types of security requirements:

- **Managed security requirements** – AWS-provided requirements based on industry standards and best practices. These requirements are ready to use and maintained by AWS.
- **Custom security requirements** – Requirements you define and maintain to address your organization's specific security policies and standards.

You can customize managed requirements by creating a copy that you can modify, or create entirely new requirements tailored to your needs.

> **Tip**
>
> Click on any managed security requirement to view its full definition, including applicability criteria, compliance evaluation details, and remediation guidance.

## Enable or disable managed security requirements

Enable AWS-managed security requirements to enforce industry-standard security policies in your design and code reviews. You can enable multiple requirements at once and disable them when they're no longer needed.

1. In the AWS console, navigate to AWS Security Agent.
2. In the navigation pane, choose **Security requirements**.
3. Choose the **Managed security requirements** tab.
4. Select the checkbox next to one or more security requirements you want to enable or disable.
5. Do one of the following:
   a. To enable the selected requirements, choose **Enable**.
   b. To disable the selected requirements, choose **Disable**.
6. Verify the change in the **Status** column, which displays **Enabled** or **Disabled**.

> **Note**
>
> Enabled security requirements are immediately applied to new design reviews and code reviews. Existing reviews are not affected.

## Customize a managed security requirement

Create a customized copy of an AWS-managed security requirement when you want to modify it to match your organization's specific standards. The customized requirement becomes a custom security requirement that you can edit and maintain.

1. In the AWS console, navigate to AWS Security Agent.
2. In the navigation pane, choose **Security requirements**.
3. Choose the **Managed security requirements** tab.
4. Select the checkbox next to the security requirement you want to customize.

> **Tip**
>
> You can only customize one requirement at a time. To customize multiple requirements, repeat this procedure for each one.

5. Choose **Customize**.
6. AWS Security Agent opens a **Create custom security requirement** form pre-populated with the managed requirement's content.
7. (Optional) Edit any fields to customize the requirement for your organization's needs.
8. Do one of the following:
   a. To create the requirement without enabling it, choose **Create security requirement**.
   b. To create and immediately enable the requirement for all future security reviews, choose **Create and enable security requirement**.

> **Note**
>
> Customizing a managed requirement creates an independent custom security requirement. Changes to the original managed requirement by AWS do not affect your custom version. Both the managed requirement and your custom version can be enabled simultaneously.

## Create a custom security requirement

Create a custom security requirement from scratch to enforce security policies unique to your organization that aren't covered by AWS-managed requirements.

1. In the AWS console, navigate to AWS Security Agent.
2. In the navigation pane, choose **Security requirements**.
3. Choose the **Custom security requirements** tab.
4. Choose **Create security requirement**.

5. (Optional) To use a managed requirement as a template, in the **Customize a managed security requirement** section, search for and select a managed requirement.

> **Tip**
>
> Selecting a managed requirement pre-populates the form fields with that requirement's details, which you can then modify to fit your needs.

6. In the **Security requirement details** section, configure the following fields:
   a. **Security requirement name** – Enter a descriptive name that clearly identifies the security control (maximum 80 characters).
   b. **Description** – Provide a concise summary of what this security requirement enforces and why it matters (maximum 500 characters).
   c. **Applicability** – Define when this requirement applies by specifying the types of workloads, systems, or conditions where it's relevant. Include specific scenarios where the requirement should be marked NOT_APPLICABLE (maximum 10,000 characters).
   d. **Compliance criteria** – Define what makes a design or code compliant versus non-compliant. Provide specific indicators, examples, and technical details that AWS Security Agent should look for when evaluating compliance (maximum 10,000 characters).
   e. **Remediation guidance** (Optional) – Provide step-by-step instructions for fixing violations, including specific technical details, configuration examples, and links to your organization's internal documentation or standards (maximum 10,000 characters).
7. Do one of the following:
   a. To create the requirement without enabling it, choose **Create security requirement**.
   b. To create and immediately enable the requirement for all future security reviews, choose **Create and enable security requirement**.

> **Note**
>
> If you choose **Create security requirement** without enabling, the requirement appears in the Custom security requirements tab with a disabled status. You must manually enable it through the Custom security requirements tab before AWS Security Agent applies it to design and code reviews.

## Edit a custom security requirement

Modify an existing custom security requirement to update its definition, criteria, or remediation guidance.

1. In the AWS console, navigate to AWS Security Agent.
2. In the navigation pane, choose **Security requirements**.
3. Choose the **Custom security requirements** tab.
4. Select the custom security requirement you want to edit.
5. From the *Actions* menu, select **Edit**.
6. Update any of the security requirement fields as needed.
7. Choose **Update security requirement**.

> **Note**
>
> Changes to custom security requirements are immediately applied to new design reviews and code reviews. Existing reviews are not affected.

## Enable or disable custom security requirements

Enable or disable your custom security requirements to control which policies AWS Security Agent enforces during security reviews.

1. In the AWS console, navigate to AWS Security Agent.
2. In the navigation pane, choose **Security requirements**.
3. Choose the **Custom security requirements** tab.
4. Select the checkbox next to one or more custom security requirements you want to enable or disable.
5. Do one of the following:
   a. To enable the selected requirements, from *Actions* choose **Enable**.
   b. To disable the selected requirements, from *Actions* choose **Disable**.
6. Verify the change in the **Status** column, which displays **Enabled** or shows disabled requirements.

## Best practices for defining security requirements

When creating or customizing security requirements, follow these guidelines to help AWS Security Agent accurately evaluate your applications and provide actionable findings.

**Security requirement name** – Use clear, specific names that identify the security control being enforced. Avoid generic terms that don't convey the requirement's purpose.

**Description** – Explain what the security control enforces and why it matters. Focus on the control's purpose and the risk it mitigates to help users understand the requirement's importance.

**Applicability** – Specify what types of workloads or systems this applies to and define the conditions that trigger evaluation. Clearly state when the requirement should be marked NOT_APPLICABLE with specific scenarios to avoid false positives. Use phrases like "This control applies to ALL workloads that…" and "Mark as NOT_APPLICABLE if…" to provide clear scope boundaries and handle edge cases.

**Compliance criteria** – Structure this in two parts: what demonstrates compliance and what indicates non-compliance. Be specific with technical indicators and include edge cases to help AWS Security Agent accurately evaluate your applications. Start with "A design is compliant if it demonstrates…" followed by specific technical details, then "A design is clearly non-compliant if it…" with patterns that indicate violations. This structure helps AWS Security Agent distinguish between compliant and non-compliant implementations.

**Remediation guidance** – Provide step-by-step instructions with specific technical details and configuration examples. Include links to your organization's internal documentation or standards to give developers the resources they need to fix violations.

## Next steps

After configuring your security requirements:

- Enable penetration testing to complement your design and code reviews
- Configure the Agent Web App to provide users access to security capabilities
- Assign users to your agent through IAM Identity Center

# Connect AWS Security Agent to GitHub repositories

Connect your AWS Security Agent to GitHub repositories to enable code review and penetration testing capabilities. AWS Security Agent supports both cloud-hosted GitHub and cloud-hosted GitHub Enterprise. GitHub integration serves multiple purposes:

- **Code review** - Automatically analyze pull requests against your organizational security requirements
- **Penetration testing context** - Provide application understanding by analyzing source code, data flows, and architecture
- **Automated remediation** - Submit pull requests with fixes for vulnerabilities discovered during penetration testing

Connecting GitHub to AWS Security Agent requires authorizing the AWS Security Agent GitHub App for your GitHub organization or user account, then registering the connection in the AWS Console.

## How GitHub integration works

**Code review** happens within GitHub. After you authorize the GitHub App and connect repositories in the AWS Management Console, you can enable code review for specific repositories. AWS Security Agent will then automatically analyze pull requests in those repositories. You review the findings directly in GitHub as pull request comments.

**Penetration testing** is initiated within the AWS Security Agent Web Application. Users specify target domains and select connected repositories to provide application context. If you enable automated remediation, users can request AWS Security Agent to fix findings by opening pull requests to connected repositories.

## Prerequisites

Before you begin, ensure you have:

- GitHub organization admin access or GitHub user account owner access
- Permissions to configure integrations for your Agent Space in the AWS Management Console
- Understanding of which repositories you want to connect for code review and penetration testing

**Important**

A GitHub App can only be installed once to a GitHub account or GitHub organization. If you need to connect the same GitHub organization to AWS Security Agent, you must use the same AWS account where the integration was first registered.

**Note**

If your GitHub enterprise organization has enabled IP allowlisting, you must accept the allowed IP addresses on the GitHub app. You can also choose to automatically add the IP addresses to your allow list. For more information, see Allowing access by GitHub Apps and Enabling allowed IP addresses in the GitHub documentation.
The following IP addresses are used to access your GitHub resources:

- Asia Pacific (Sydney) (ap-southeast-2)
  - `13.237.95.197`
  - `13.238.84.102`
  - `52.64.174.242`
- Asia Pacific (Tokyo) (ap-northeast-1)
  - `13.192.12.233`
  - `35.74.181.230`
  - `57.183.50.158`
- Europe (Frankfurt) (eu-central-1)
  - `18.158.110.140`
  - `52.57.96.160`
  - `52.59.55.56`
- Europe (Ireland) (eu-west-1)
  - `34.251.85.24`
  - `52.30.157.157`
  - `52.51.192.222`
- US East (N. Virginia) (us-east-1)
  - `34.228.181.128`

- 44.219.176.187
- 54.226.244.221

- US West (Oregon) (us-west-2)
  - 34.212.16.133
  - 52.89.67.212
  - 54.187.135.61

## Authorize and register the AWS Security Agent GitHub App

Authorize the AWS Security Agent GitHub App to access your GitHub organization or user account, then register the connection in the AWS Console.

> **Important**
>
> Complete all steps in this process without closing your browser or navigating away. If the registration process is interrupted, you may need to uninstall the GitHub App and start over.

1. In the AWS Security Agent Management Console, navigate to **Integrations**.
2. Click **Add integration**.
3. Select **GitHub**.
4. Click **Next**.
5. Click **Install and authorize**.

   You'll be redirected to GitHub to complete the authorization. Ensure you're logged into GitHub with an account that has admin access to the organization or user account you want to connect.
6. In GitHub, select the account or organization where you want to install the AWS Security Agent GitHub App.
7. Select which repositories AWS Security Agent can access:
   - **All repositories** - Grant access to all current and future repositories in the organization or user account
   - **Only select repositories** - Choose specific repositories from the dropdown. You can select multiple repositories one at a time.

> **Note**
>
> You can modify repository access at any time by visiting the GitHub App settings in your GitHub organization or user account settings.

8. Click **Install and authorize**.
9. You'll be redirected back to the AWS Management Console to complete the registration.
10. In the **Registration details** section, configure the following fields:
    a. **Registration name** - Enter a descriptive name for this GitHub connection. Use a name that identifies the GitHub organization or user account, such as "Acme-Corp-Org" or "Production-Repos".
    b. **Account type** - Select one of the following from the dropdown:
        - **Organization** - If you connected a GitHub organization account
        - **User** - If you connected a personal GitHub user account
    c. **Organization name** (appears only if you selected Organization) - Enter the exact name of your GitHub organization as it appears in GitHub.
11. Click **Connect**.
12. You'll see a confirmation message and return to the Integrations page, where your new GitHub connection appears with its registration name. To connect additional GitHub organizations or user accounts, repeat this process by clicking **Add integration** again.

## Troubleshoot GitHub integration

If you encounter issues during the GitHub integration process, use the following guidance to resolve common problems.

### Unable to complete registration

If you were unable to complete the registration process (for example, your browser was closed, you navigated away from the registration page, or you encountered a session interruption), the AWS Security Agent GitHub App may be installed in your GitHub organization but not registered in the AWS Console.

**Symptoms:**

- When you try to authorize the GitHub App again, GitHub shows "Configure" instead of "Install"
- You cannot complete the registration in the AWS Console
- The integration does not appear in your Integrations list

**Resolution:**

1. Uninstall the AWS Security Agent GitHub App from your GitHub organization or user account.
2. Return to the AWS Security Agent console and start the integration process again from the beginning.

### Multiple AWS accounts trying to integrate the same GitHub organization

A GitHub App can only be installed once to a GitHub account or GitHub organization. If you need to use repositories from a GitHub organization that is already integrated with a different AWS Security Agent account, you must use the AWS account where the integration was first registered.

**Resolution:**

- Identify which AWS Security Agent account has the GitHub integration registered
- Use that AWS account to create Agent Spaces and connect repositories
- If you need to move the integration to a different AWS account, uninstall the GitHub App from the original AWS account first, then integrate it with the new account

## Next steps

After connecting GitHub to AWS Security Agent:

- Navigate to the Agent Space where you want to use these repositories
- Click **Enable code review** or **Setup penetration testing** to connect specific repositories to your Agent Space and configure their usage
- Enable automated remediation to allow AWS Security Agent to submit pull requests with vulnerability fixes
- Review GitHub App permissions and repository access in your GitHub organization settings

# Remove a GitHub integration

Remove a GitHub integration when you no longer need AWS Security Agent to access repositories from a specific GitHub organization or user account. You must first uninstall the AWS Security Agent GitHub App from GitHub before removing the integration in the AWS Console.

## Prerequisites for removal

Before removing a GitHub integration, ensure you have:

- Checked which Agent Spaces have repositories connected from this integration
- Understood the impact: Removing this integration will break repository connections for code review, penetration testing context, and penetration test remediation across all Agent Spaces using repositories from this integration
- GitHub organization admin access or user account owner access to uninstall the GitHub App

## Step 1: Uninstall the AWS Security Agent GitHub App from GitHub

First, uninstall the AWS Security Agent GitHub App from your GitHub organization or user account.

**For GitHub Organizations:**

1. Go to github.com and open your organization page.
2. In the left sidebar, click **Settings**.
3. Under **Code, planning, and automation**, click **Installed GitHub Apps**.
4. Locate the **AWS Security Agent** app in the list.
5. Click **Configure** next to the AWS Security Agent app.
6. Scroll to the bottom of the configuration page and click **Uninstall**.
7. Confirm the uninstallation when prompted.

**For GitHub User Accounts:**

1. Go to github.com.
2. Click your profile picture in the top-right corner.
3. Click **Settings**.

4. In the left sidebar, select **Applications**.
5. Open the **Installed GitHub Apps** tab.
6. Locate the **AWS Security Agent** app in the list.
7. Click **Configure** next to the AWS Security Agent app.
8. Scroll to the bottom of the configuration page and click **Uninstall**.
9. Confirm the uninstallation when prompted.

## Step 2: Remove the integration from AWS Security Agent

After uninstalling the GitHub App, remove the integration registration from the AWS Console.

1. In the AWS Security Agent Management Console, navigate to **Integrations**.
2. Locate the GitHub integration you want to remove in the integrations list.
3. Select the integration by clicking on it.
4. Click **Remove**.
5. Review the confirmation dialog, which warns you about the impact:

> ⚠ **Warning**
>
> Removing this integration will affect all Agent Spaces that have repositories connected from this GitHub organization or user account. This will break:
>
> - Code review functionality for connected repositories
> - Penetration testing context from connected repositories
> - Penetration test remediation capabilities for connected repositories
>
> Ensure you have uninstalled the AWS Security Agent GitHub App from GitHub before proceeding.

6. If you have not yet uninstalled the GitHub App from GitHub, you'll receive a warning. Return to Step 1 to complete the uninstallation first.
7. If you have uninstalled the GitHub App and understand the impact, click **Confirm removal**.
8. The integration will be removed from your integrations list. Any Agent Spaces with repositories from this integration will no longer have access to those repositories for code review, penetration testing context, or automated remediation.

# Create an IAM Role for AWS Security Agent

AWS Security Agent uses IAM Roles in three ways:

1. **Application Role:** Used when creating the AWS Security Agent application. For IAM Identity Center and admin access link use cases, the service assumes this role to grant WebApp users permissions to interact with AWS Security Agent APIs.
2. **Penetration Test Service Role:** Specified when creating agent spaces as a list of available roles. Later, WebApp users select one of these roles when creating a penetration test. AWS Security Agent service assumes this role to access your AWS resources during testing.
3. **Actor Role:** Used to authenticate and authorize requests to your target web application (for example, AWS API Gateway APIs). These roles are provided during agent space creation. The AWS Security Agent agent assumes actor roles to interact with your target application.

## Application Role

The Application Role is used when creating your AWS Security Agent application in the service. For IAM Identity Center and admin access link authentication scenarios, the AWS Security Agent service assumes this role to grant WebApp users the necessary permissions to interact with AWS Security Agent APIs.

### Required Permissions

This role needs permissions to:

- Invoke AWS Security Agent API operations
- Read and write application configuration data
- Access user session information
- Manage authentication tokens for WebApp users

### Trust Policy

The trust policy must allow the AWS Security Agent service to assume this role:

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

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

### Permissions Policy

The role should include permissions for:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "securityagent:GetApplication",
        "securityagent:UpdateApplication",
        "securityagent:ListAgentInstances",
        "securityagent:CreatePentestSession"
      ],
      "Resource": "arn:aws:securityagent:*:*:application/*"
    }
  ]
}
```

**Note**

Customize the permissions based on your specific application requirements and principle of least privilege.

## Penetration Test Service Role

The Penetration Test Service Role is specified when creating agent spaces as a list of available roles. When WebApp users create a penetration test, they select one of these roles. The AWS Security Agent service then assumes this role to access and test your AWS resources.

## Required Permissions

This role needs permissions to access and analyze your AWS resources during penetration testing:

- Read and describe VPC configurations and network topology
- Inspect EC2 instances, security groups, and network ACLs
- Analyze IAM policies and resource permissions
- Read CloudWatch logs and metrics
- Access AWS service configurations relevant to security testing

## Trust Policy

The trust policy must allow the AWS Security Agent service to assume this role for penetration testing operations:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "your-external-id"
        }
      }
    }
  ]
}
```

**Note**

Use an External ID for additional security when allowing cross-account or service access.

## Permissions Policy

The role should include read-only access to your AWS resources. Consider using these managed policies:

- `SecurityAudit` - AWS managed policy for security auditing
- `ViewOnlyAccess` - Read-only access to most AWS services

Or create a custom policy with specific permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "vpc:Describe*",
        "iam:Get*",
        "iam:List*",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "*"
    }
  ]
}
```

**Important**

Grant only the minimum permissions required for your penetration testing scope. Review and adjust permissions based on which AWS services you want to include in security testing.

# Actor Role

The Actor Role is used to authenticate and authorize requests to your target web application during penetration testing. These roles are provided during agent space creation, and the AWS Security Agent agent assumes them to interact with your target application endpoints (such as AWS API Gateway APIs, Lambda function URLs, or other AWS-hosted applications).

## Required Permissions

This role needs permissions to:

- Invoke API Gateway endpoints
- Execute Lambda functions
- Access application-specific AWS resources
- Authenticate with your target application's authentication mechanisms
- Perform HTTP operations against your application endpoints

## Trust Policy

The trust policy must allow the AWS Security Agent agent service to assume this role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "your-external-id"
        }
      }
    }
  ]
}
```

## Permissions Policy

The permissions depend on your target application architecture. Here are examples for common scenarios:

### For API Gateway Applications

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": "arn:aws:execute-api:us-east-1:*:your-api-id/*"
    }
  ]
}
```

### For Lambda Function URLs

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunctionUrl",
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:*:function:your-function-name"
    }
  ]
}
```

### For Application Load Balancer with Cognito Authentication

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

```
      "Action": [
        "cognito-idp:InitiateAuth",
        "cognito-idp:RespondToAuthChallenge"
      ],
      "Resource": "arn:aws:cognito-idp:us-east-1:*:userpool/your-user-pool-id"
    }
  ]
}
```

**Important**

Configure Actor Role permissions to match your target application's authentication and authorization requirements. The role should have the same level of access as the users or services that the Security Agent will simulate during penetration testing.

## Connect agent to private VPC resources

If the application you want to run a penetration test on is not available on the public internet, you need to provide AWS Security Agent with a VPC configuration. AWS Security Agent will use this VPC configuration, including a VPC, subnet, and security groups, to access the application.

**Note**

When testing endpoints in a private VPC, only endpoints resolving to IPs in known private IP ranges are allowed (see VPC CIDR blocks for more information). The following IPv4 and IPv6 ranges are allowed:

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
fd00::/8
```

**Note**

When connecting to a subnet, AWS Security Agent will create an ENI (Elastic Network Interface) in the subnet configured for the penetration test. This ENI does not have an associated public IP address, meaning that it cannot communicate with VPC Internet

Gateways in public subnets. If your penetration test requires open internet access, please use a private subnet with an associated VPC NAT Gateway instead

You grant AWS Security Agent general access to a VPC from the AWS Management Console. In the Security Agent web app, users select the specific configuration for a penetration test.

## To add a VPC in the Agent Space

1. Navigate to the Agent Space overview page
2. Select **Actions** and then **Edit penetration test configuration**
3. Under the **VPC** heading, specify the **VPC**, **Subnets**, and **Security groups**

You can add up to 5 VPCs.

## To select a specific VPC configuration for a penetration test in the Security Agent web app

1. Navigate to the Penetration Tests overview page
2. Select the penetration test that you need to add VPC configuration for, and then choose **Modify pentest details**
3. Select **Next** at the bottom of the page to reach the **VPC Resources** section
4. Select the **VPC**, **Subnet**, and **Security groups**
5. Select **Next** to reach the last section and **Save** the penetration test

## Running a penetration test against VPC resources in another AWS account

You can run penetration tests against VPC resources shared with your account using AWS Resource Access Manager. Both accounts must be part of the same AWS Organization.

1. (Optional) Enable automatic resource sharing for your AWS organization

```
aws ram enable-sharing-with-aws-organization
```

2. Using credentials from the AWS account that owns the VPC resources, share subnet and security group resources with the penetration test owner account

```
aws ram create-resource-share \
    --name SharePentestResources \
    --resource-arns <subnet ARN> <security group ARN> \
    --principals <penetration test owner account ID>
```

3. Navigate to the Agent Space overview page
4. Select **Penetration test** and locate **Service role name**
5. Verify that the IAM role grants access to the shared VPC resources
6. Select **Actions** and then **Edit penetration test configuration**
7. Under the **VPC** heading, specify the shared **VPC**, **Subnets**, and **Security groups** and save the updated configuration.
8. Navigate to the Penetration Tests overview page on the AWS Security Agent web app
9. Select the penetration test that you need to add VPC configuration for, and then choose **Modify pentest details**
10. Update the penetration test to use the shared VPC resources

# Enable users to start remediation of penetration test findings

In the AWS Management Console, you can enable the code remediation feature that allows users to remediate findings in the penetration test web app.

When you enable this functionality in the AWS Management Console, users of the AWS Security Agent web app can start code remediation for a specific finding. The remediation will be available as GitHub pull requests.

> **Note**
>
> Code remediation is currently available in only us-east-1.

## Prerequisites

Before you begin, ensure you have:

1. Enabled penetration test (see the section called “Enable penetration test”)
2. Installed and authorized the AWS Security Agent GitHub App for your GitHub organization (see the section called “Connect AWS Security Agent to GitHub repositories”)

## Select repositories and enable code remediation capability

1. Navigate to the Agent Space overview page.
2. Choose **Penetration test** tab.
3. Select a GitHub registration that owns your GitHub repositories.
   a. If you haven’t associated any GitHub registration to the Agent Space, you can see a **Connect GitHub for penetration testing** information box. Click the **Add** button on its right side to select the GitHub registration.
   b. If you already associated some GitHub registration to the Agent Space, you can add more by clicking the **Add** button in the **Connected integrations** section.
4. Click **Next** to choose GitHub repositories.
5. Click **Next** to configure repositories capabilities. In the **Pentest remediation enabled** column, mark the repositories as **Enabled** to allow the Agent Space to remediate the code according to the penetration findings.
6. Click **Connect** to finish the configuration.

# Provide agent resources from an S3 bucket

You can grant AWS Security Agent access to resources in an S3 bucket, such as API documents, threat model documents, and source code that support penetration testing.

You grant AWS Security Agent read access to an S3 bucket in the AWS Management Console. In the Security Agent web app, users select individual resources from S3 as relevant.

1. Navigate to the Agent Space overview page
2. Select **Actions** and then **Edit penetration test configuration**
3. Under the **S3 buckets** section, define the **S3 URI** containing the **S3 Bucket**. You can add multiple S3 buckets.

# Enable an application domain for penetration testing

Before you can run a penetration test on an application, you need to add a target domain and verify ownership. AWS Security Agent will only perform penetration tests against verified domains.

> **Note**
>
> You do not need to validate ancillary domains that your application may use. You only need to validate the domain you will actively run penetration tests against.

## Step 1: Add a target domain

To add a target domain, navigate to the **Penetration test** tab on the Agent Space overview page. Depending on whether you have already configured penetration testing, use one of the following methods:

- **First-time setup**: Choose **Set up penetration test** to open the penetration test wizard. In the first step, enter the domain and choose a verification method.
- **Adding a domain to an existing configuration**: In the **Target Domains** table, choose **Add domain**. Enter the domain and choose a verification method in the modal.

You can add a base domain or a sub-domain, such as `example.com` or `billing.example.com`. AWS suggests using a sub-domain where you have permission to create TXT records.

Choose one of the following verification methods:

- **DNS TXT record**: Prove domain ownership by creating a DNS TXT record with your DNS provider.
- **HTTP route**: Prove domain ownership by creating a route on your web server that contains a unique token provided by AWS Security Agent.

## Step 2: Verify domain ownership

After adding the domain, verify ownership using the method you selected. You can trigger verification at any time from the **Target Domains** table by selecting the domain and choosing **Verify**.

## Verify using a DNS TXT record

AWS Security Agent generates a TXT DNS record value. You must add this record with your DNS provider to prove ownership.

**If the domain is registered in Route 53 (same AWS account):**

AWS Security Agent can create the DNS record automatically. Select the domain from the Target Domains table and choose **One-click verification**. AWS Security Agent creates the DNS validation record and completes verification automatically.

**If the domain is registered with another DNS provider:**

1. Copy the TXT record value provided by AWS Security Agent.
2. Add the TXT record with your DNS registrar.
3. Return to the Target Domains table, select the domain, and choose **Verify**.

## Verify using HTTP route validation

This method proves domain ownership by placing a unique token on your web server. Only domain owners or authorized web administrators can create routes on a web server, which proves ownership.

1. Create a file at the following path on your web server:

```
.well-known/aws/securityagent-domain-verification.json
```

2. Place the token provided by AWS Security Agent in the file using this format:

```
{
  "tokens": ["<insert-token>"]
}
```

3. Return to the Target Domains table, select the domain, and choose **Verify**. AWS Security Agent sends an HTTPS GET request to the verification URL and validates the token.
4. If the domain is accessible on the public internet, make sure that your domain has a valid SSL certificate before running verification.

> **Note**
>
> If your domain is registered in multiple agent spaces and you are using HTTP route validation, you can place the tokens for both agent spaces in the same `tokens` array.

# Managed target domains used for penetration testing

In the AWS Management Console, you can add and manage target domains consumed by agent spaces for penetration testing. These target domains will need to be verified before they can be used in a penetration test. For more information about verifying target domains, see the section called “Enable penetration test”

## Prerequisites

Before you begin, ensure you have:

1. Enabled penetration test (see the section called “Enable penetration test”)

## Manage target domain resources

- Navigate to the Target Domains overview page.
- You should see all target domain resources associated with your account
  - If you don’t see any target domains associated with your account, follow the steps in the section called “Enable penetration test” to create and verify a target domain
- Target domains can be reused between agent spaces and share verification status
  - To add an existing target domain to an agent space, navigate to the **Penetration test** tab of the agent space. Select **Add domain** and click the desired domain under **Select from available previously registered domains** in the domain name field
  - Target domains must be associated with an agent space before they can be used in a penetration test
- Removing a domain from an agent space does not delete the domain. The associated target domain can be permanently deleted from the Target Domains overview page

## Verify target domains

In order for a target domain to be used in a penetration test, it must first be verified using one of the below methods:

- **Route 53 domains (same AWS account)**: Choose **One-click verification**. AWS Security Agent automatically creates the DNS record and completes verification.
- **DNS TXT (other DNS providers)**: Copy the verification token, add the TXT record with your DNS registrar, then select the domain and choose **Verify**.
- **HTTP route**: Place the verification token at the required route path on your web server, then select the domain and choose **Verify**. For details, see the section called “Enable an application domain for penetration testing”.
- **Private VPC domain**: Verify the target domain IP falls within a private CIDR range (see the section called “Connect agent to private VPC resources” for a list of private CIDR ranges). Click **Verify** and confirm that the target domain status becomes **Unreachable**. This domain can now be used for penetration testing with a configured VPC

# Grant users access to the AWS Security Agent web app

AWS Security Agent provides two methods for users to access the web application, depending on how you configured your Agent Space during setup.

## Access methods overview

**IAM Identity Center (SSO)** - If you enabled IAM Identity Center when creating your Agent Space, users can access the web application directly through SSO. You assign users to the Agent Space either through the IAM Identity Center console or through the AWS Security Agent console, and users log in using their Identity Center credentials.

**Admin Access** - Users with AWS Console access can launch the web application through an admin access link on the Agent Space overview page in the AWS Management Console.

## Grant access with IAM Identity Center (SSO)

If you configured your Agent Space with IAM Identity Center, you can assign users to the Agent Space using either the AWS Security Agent console or the IAM Identity Center console.

## Assign users through the AWS Security Agent console

1. In the AWS Security Agent Management Console, navigate to your Agent Space.
2. Select the **Web app** tab.
3. In the **Users** table, click **Add users**.
4. Select existing users from IAM Identity Center or create new users.
5. Confirm the user assignments.

> **Tip**
>
> Users assigned to the Agent Space can access the web application by logging in through IAM Identity Center with their SSO credentials.

## Assign users through the IAM Identity Center console

You can also manage user access directly from the IAM Identity Center console. Learn how to assign user access to applications in the IAM Identity Center console in the AWS IAM Identity Center User Guide.

## Access the web application with SSO

After users are assigned to the Agent Space:

1. Users navigate to the web application URL for the Agent Space.

   > **Tip**
   >
   > Find the web app URL on the Agent Space detail page in the AWS Security Agent console by selecting **Copy web app URL**. Users should bookmark this URL for easy access.

2. Users log in using their SSO credentials.
3. After authentication, users can select the Agent Space and begin conducting security assessments.

## Grant access with IAM-only access

If you configured your Agent Space with IAM-only access, users with AWS Console access can launch the web application through an admin access link.

### Use the admin access link

1. Log into the AWS Security Agent console.
2. Navigate to the Agent Space you want to access.
3. On the Web app tab of the Agent Space landing page click the **Admin access** button to launch the web application.
4. The web application opens in a new tab with the user automatically authenticated.

> **Note**
>
> The admin access link is only available to users who are already authenticated to the AWS Console with appropriate AWS Security Agent permissions. This method does not require IAM Identity Center configuration.

## Next steps

After granting users access to the web application:

- Users can create and manage penetration test configurations and runs
- Users can create and manage design reviews
- Users can view security findings and remediation guidance
- Configure notification preferences for security findings

# Customer managed keys for AWS Security Agent

By default, AWS Security Agent encrypts all data at rest using AWS-managed encryption keys. You can optionally use a customer managed key from AWS Key Management Service (AWS KMS) to encrypt your data, giving you full control over the encryption keys that protect your resources.

AWS Security Agent supports resource-level customer managed keys. When you create a top-level resource such as an Agent Space or integration, you can specify a KMS key to encrypt all data belonging to that resource and its subresources. For example, specifying a customer managed key when creating an Agent Space encrypts all data associated with that Agent Space, including Agent Space configurations, penetration test configurations, jobs, and execution details, security findings, discovered endpoints, and screenshots.

## How customer managed keys work

AWS Security Agent uses the AWS KMS Hierarchical keyring to encrypt your data with customer managed keys. When you specify a KMS key during resource creation, the service creates a branch key protected by your KMS key and stores it in an Amazon DynamoDB-based key store. For each encryption operation, the Hierarchical keyring derives a unique wrapping key from the active branch key, which encrypts a unique data encryption key for each request.

The Hierarchical keyring provides the following benefits:

- **Per-resource encryption scope** – Each top-level resource has its own branch key, so data for different resources is encrypted under separate keys.
- **Automatic key rotation** – AWS Security Agent periodically rotates branch keys. After rotation, new data is encrypted with the new branch key version, while older versions are retained to decrypt previously encrypted data.

### Encryption context

AWS Security Agent uses encryption context in all cryptographic operations with your KMS key. The encryption context is a set of non-secret key-value pairs that provides additional authenticated data for the encryption operation.

The encryption context key follows the format `aws:securityagent:_<resource-type>_`, where *`<resource-type>`* is the type of resource being encrypted (for example, `agent-space` or `integration`). The value is the Amazon Resource Name (ARN) of the resource.

You can use the encryption context to scope down your KMS key policy to specific resource types. For more information, see the section called “Configure a KMS key policy”.

### Default encryption

When you create a resource without specifying a customer managed key, AWS Security Agent encrypts the resource using AWS-managed encryption keys provided by the underlying storage services (Amazon DynamoDB and Amazon S3). No additional configuration is required for default encryption.

### Default KMS key for applications

You can set a default KMS key at the application level. When a default KMS key is configured, AWS Security Agent uses it as the fallback for new Agent Spaces and integrations when you don't explicitly specify a KMS key during resource creation.

To set a default KMS key, specify the `defaultKmsKeyId` parameter when creating or updating your application using the AWS CLI or SDK. The console prompts you to specify a default KMS key during application setup.

When you explicitly specify a KMS key during resource creation, it takes precedence over the application-level default.

## Prerequisites

Before you configure a customer managed key, complete the following prerequisites:

- Create a symmetric encryption KMS key in AWS KMS. The key must meet the following requirements:
  - Key type: Symmetric
  - Key usage: Encrypt and decrypt
  - Key spec: `SYMMETRIC_DEFAULT`
  - Key state: Enabled

  For instructions, see Creating keys in the *AWS Key Management Service Developer Guide*.
- Configure the KMS key policy to grant AWS Security Agent the required permissions. For details, see the section called "Configure a KMS key policy".

## Configure a KMS key policy

Your KMS key policy must grant AWS Security Agent permission to use the key for cryptographic operations. The required permissions depend on which resource types you plan to encrypt.

AWS Security Agent accesses your KMS key using different identities depending on the operation:

- **AWS Management Console operations** – When administrators create or manage resources in the AWS Console (for example, creating an Agent Space or updating an application), the service uses your IAM credentials to call AWS KMS.
- **Web application operations** – When users access data through the AWS Security Agent web application (for example, viewing penetration test results or starting a penetration test), the service uses the application role created during AWS Security Agent setup to call AWS KMS.
- **Asynchronous workflows** – For background operations that run outside of any user session (for example, penetration testing execution, code review processing, and branch key rotation), the service uses its own service principal (`securityagent.amazonaws.com`) to call AWS KMS on your behalf.

Your key policy must include statements for all three identities.

## Key policy for Agent Spaces

The following key policy grants AWS Security Agent the permissions required to encrypt and decrypt Agent Space data, including penetration test results and screenshots.

Replace the following placeholder values in the policy:

- *111122223333* – Your AWS account ID
- *MyRole* – The IAM role you use to manage AWS Security Agent in the console
- *MyApplicationRole* – The application role created during AWS Security Agent setup
- *us-east-1* – The AWS Region where you use AWS Security Agent

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowKeyMetadataValidationForApplicationAndAgentSpaces",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyRole"
      },
      "Action": [
        "kms:DescribeKey"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowUseOfHierarchicalKeyringForAgentSpaces",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyRole"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:ReEncrypt*",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:securityagent:agent-space":
"arn:aws:securityagent:us-east-1:111122223333:agent-space/*"
        },
        "StringEquals": {
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowSynchronousDataAccessForAgentSpaces",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyApplicationRole"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
```

```
          "kms:EncryptionContext:aws:securityagent:agent-space":
 "arn:aws:securityagent:us-east-1:111122223333:agent-space/*"
        },
        "StringEquals": {
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowAsynchronousDataAccessForAgentSpaces",
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:securityagent:agent-space":
 "arn:aws:securityagent:us-east-1:111122223333:agent-space/*"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:securityagent:us-east-1:111122223333:agent-space/*"
        }
      }
    }
  ]
}
```

**Important**

To rotate branch keys, your KMS key policy must grant `kms:GenerateDataKeyWithoutPlaintext` and `kms:ReEncrypt*` permissions to the AWS Security Agent service principal (`securityagent.amazonaws.com`), or branch key rotation will fail. For more information about the required permissions, see Rotate a branch key.

## Key policy for integrations

The following key policy grants AWS Security Agent the permissions required to encrypt and decrypt integration data, including code review findings.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowKeyAccessValidationForIntegrations",
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::111122223333:role/MyRole"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws-crypto-ec:aws:securityagent:integration":
 "arn:aws:securityagent:us-east-1:111122223333:integration/*"
        },
        "StringEquals": {
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowKeyMetadataValidationForIntegrations",
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*"
    },
    {
      "Sid": "AllowAsynchronousDataAccessForIntegrations",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:securityagent:us-east-1:111122223333:integration/*"
        },
        "StringLike": {
          "kms:EncryptionContext:aws-crypto-ec:aws:securityagent:integration":
 "arn:aws:securityagent:us-east-1:111122223333:integration/*"
        }
      }
    }
  ]
}
```

**Note**

You can combine the Agent Space and integration key policy statements into a single key policy if you want to use the same KMS key for both resource types.

## IAM policy for the application role

In addition to the KMS key policy, attach the following IAM policy to the application role specified during AWS Security Agent setup. This policy grants the role permissions to use your customer managed keys for encrypting and decrypting Agent Space data and creating secrets in AWS Secrets Manager.

Replace the following placeholder values in the policy:

- *111122223333* – Your AWS account ID
- *us-east-1* – The AWS Region where you use AWS Security Agent
- The KMS key ARNs in `Resource` – The ARNs of your KMS keys

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSynchronousDataAccessForAgentSpaces",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890cd"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333",
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:securityagent:agent-space":
 "arn:aws:securityagent:us-east-1:111122223333:agent-space/*"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyAccessForCreatingSecrets",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890cd"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "secretsmanager.*.amazonaws.com",
```

```
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:*"
        }
      }
    }
  ]
}
```

> **Note**
>
> The application role is an IAM role that you specify or that the console creates during AWS Security Agent setup. The web application assumes this role to retrieve penetration test execution logs and create Secrets Manager secrets on your behalf.
> The `AllowKmsKeyAccessForCreatingSecrets` statement is required if you configure authentication resources for penetration tests and choose to enter credentials directly instead of specifying an existing secret. The web application creates a secret on your behalf, and the application role needs the permissions in this statement to encrypt the secret with the customer managed key specified for your Agent Space. Update the `Resource` ARNs in this statement to match the KMS key used for your Agent Space.
> If your CloudWatch Logs log group for storing penetration test execution logs is encrypted with a customer managed key, you must update the KMS key policy to allow CloudWatch Logs to decrypt log events. Otherwise, the web application cannot display log events. For instructions, see Encrypt log data in CloudWatch Logs using AWS KMS.

## IAM policy for the penetration test service role

During penetration testing, AWS Security Agent assumes the penetration test service role to access your AWS resources. If any of these resources are encrypted with a customer managed key, you must grant the service role permissions to use the corresponding KMS keys. This applies to the following resources:

- **S3 buckets** – If you provide learning resources (such as API documents, threat models, or source code) from an S3 bucket encrypted with a customer managed key, the service role needs permissions to decrypt objects in that bucket. For more information about configuring SSE-KMS for S3, see Protecting data with SSE-KMS.
- **Secrets Manager secrets** – If your penetration test credentials are stored in Secrets Manager secrets encrypted with a customer managed key, the service role needs permissions to decrypt

those secrets. For more information about secret encryption, see Secret encryption and decryption in Secrets Manager.

- **CloudWatch Logs log groups** – If the CloudWatch Logs log group used for storing penetration test execution logs is encrypted with a customer managed key, the service role needs permissions to encrypt and decrypt log data. For more information about encrypting log data, see Encrypt log data in CloudWatch Logs using AWS KMS.

> **Important**
>
> If you use a different KMS key for encrypting these resources than the one you specified for your Agent Space, you must grant the service role permissions for each KMS key that protects a resource the service role accesses during penetration testing.

Attach the following IAM policy to the penetration test service role. Include only the statements that apply to your configuration.

Replace the following placeholder values in the policy:

- *111122223333* – Your AWS account ID
- *us-east-1* – The AWS Region where you use AWS Security Agent
- The KMS key ARNs in `Resource` – The ARNs of the customer managed keys used to encrypt each resource

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowKmsAccessForEncryptedS3Buckets",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE-S3-KEY-ID"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "s3.*.amazonaws.com",
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::YOUR-BUCKET-NAME*"
        }
      }
    },
    {
      "Sid": "AllowKmsAccessForEncryptedSecrets",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE-SECRETS-KEY-ID"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "secretsmanager.*.amazonaws.com",
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:MyAppCredentials-*"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyValidationForCloudWatchLogs",
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE-LOGS-KEY-ID"
      ]
    },
    {
      "Sid": "AllowKmsAccessForEncryptedCloudWatchLogs",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
```

```
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE-LOGS-KEY-ID"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333",
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:MY-LOG-GROUP"
        },
        "StringLike": {
          "kms:ViaService": "logs.*.amazonaws.com"
        }
      }
    }
  ]
}
```

**Note**

The penetration test service role is the IAM role you specify when configuring penetration testing for an Agent Space. AWS Security Agent assumes this role to access your AWS resources during testing.

- The `AllowKmsAccessForEncryptedS3Buckets` statement is required only if you provide learning resources from S3 buckets encrypted with a customer managed key. Update the `Resource` ARN to match the KMS key used to encrypt your S3 bucket.
- The `AllowKmsAccessForEncryptedSecrets` statement is required only if your penetration test credentials are stored in Secrets Manager secrets encrypted with a customer managed key. Update the `Resource` ARN to match the KMS key used to encrypt your secrets.
- The `AllowKmsAccessForEncryptedCloudWatchLogs` statement is required only if your CloudWatch Logs log group is encrypted with a customer managed key. Update the `Resource` ARN to match the KMS key used to encrypt your log group.
- If multiple resources share the same KMS key, you can combine the statements and list the shared key ARN once in the `Resource` field.

# Create a resource with a customer managed key

You can specify a customer managed key when setting up AWS Security Agent or when creating individual resources. The KMS key encrypts all data belonging to that resource and its subresources.

## Set a default KMS key during setup (console)

You can configure a default KMS key during the initial AWS Security Agent setup. This key is used as the default for new Agent Spaces and integrations unless you specify a different key.

1. On the **Set up AWS Security Agent** page, expand the **Encryption - optional** section.
2. Select **Customize encryption settings (advanced)**.
3. For **Choose an AWS KMS key**, select a KMS key from your account, or enter a KMS key ARN.
4. Complete the remaining setup steps and choose **Set up AWS Security Agent**.

AWS Security Agent validates the KMS key to confirm that it exists, is enabled, and uses symmetric encryption. If validation fails, setup does not proceed and you receive an error message.

## Create an Agent Space with a customer managed key (console)

1. In the AWS Security Agent console, navigate to the **Agent Spaces** page.
2. Choose **Create Agent Space**.
3. Enter a name and optional description for your Agent Space.
4. Expand the **Advanced** section.
5. Under **Data encryption**, choose one of the following:
   - **Use application default key** – Uses the default KMS key configured during AWS Security Agent setup. This option is selected by default if a default key is configured.
   - **Use a different key** – Specify a different KMS key for this Agent Space. Select **Customize encryption settings (advanced)**, then for **Choose an AWS KMS key**, select a KMS key from your account or enter an ARN.
6. Choose **Create**.

## Create an integration with a customer managed key (console)

1. In the AWS Security Agent console, navigate to the **Integrations** page.
2. Choose **Add integration** and select your provider (for example, GitHub).

3. Complete **Step 1: Install and authorize** the provider application.
4. In **Step 2: Register details**, enter a registration name and configure the provider settings.
5. In the **Data encryption** section, select **Customize encryption settings (advanced)**.
6. For **Choose an AWS KMS key**, select a KMS key from your account, or enter a KMS key ARN.
7. Choose **Connect**.

### Create a resource with a customer managed key (AWS CLI or SDK)

To specify a customer managed key using the AWS CLI or SDK:

- Include the `defaultKmsKeyId` parameter when calling `CreateApplication` to set a default KMS key for your application. This key is used as the fallback when creating Agent Spaces or integrations without an explicit KMS key.
- Include the `kmsKeyId` parameter when calling `CreateAgentSpace` or `CreateIntegration` to encrypt a specific resource. This takes precedence over the application-level default.

For parameter details, see the AWS Security Agent API Reference.

If you don't specify a KMS key, AWS Security Agent uses the application-level default KMS key if one is configured. Otherwise, the resource is encrypted with AWS-managed keys.

## Key rotation

AWS Security Agent automatically rotates branch keys on a periodic basis. Branch key rotation creates a new active version of the branch key while retaining all previous versions. After rotation:

- New data is encrypted with the new branch key version.
- Previously encrypted data remains decryptable using the older branch key version.
- No data re-encryption is required.

Branch key rotation is separate from KMS key rotation. You can enable automatic KMS key rotation for your customer managed key independently. For more information, see Rotating KMS keys in the *AWS Key Management Service Developer Guide*.

After a branch key rotation, there is a brief period during which cached copies of the previous branch key may still be used for new encryption operations. This window is determined by the internal cache time-to-live (TTL) and does not affect the security of your data.

## Considerations

Keep the following in mind when using customer managed keys with AWS Security Agent:

- **Customer managed keys apply to new resources only.** Existing resources created before you configure a customer managed key continue to use AWS-managed encryption. There is no migration of existing data.
- **Customer managed keys are specified at creation time.** You specify the KMS key when creating a resource. You cannot add or change the KMS key for an existing resource.
- **Multiple KMS keys are supported.** You can use different KMS keys for different resources. For example, you can use one key for production Agent Spaces and a different key for development Agent Spaces.
- **Disabling or deleting a KMS key makes data inaccessible.** If you disable or schedule deletion of a KMS key, AWS Security Agent cannot decrypt data encrypted under that key. Affected resources become inaccessible until the key is re-enabled. Deleting a KMS key permanently prevents access to all data encrypted under that key.
- **Key policy permissions are required.** If you remove the required permissions from your KMS key policy, AWS Security Agent cannot access encrypted data. Ensure that the key policy grants permissions for your IAM role, the web application role, and the service principal as described in the section called “Configure a KMS key policy”.
- **AWS KMS quotas apply.** Cryptographic operations against your KMS key count toward your AWS KMS request quotas. Under normal usage, the hierarchical keyring architecture minimizes KMS calls through branch key caching. For more information, see Request quotas in the *AWS Key Management Service Developer Guide*.

# Troubleshooting

Find solutions to commonly seen errors when using AWS Security Agent.

## Access Denied: Incorrect GitHub account type selected or incorrect organization name specified

- You installed the AWS Security Agent application into your desired GitHub organization but incorrectly set the `GitHub Account Type` to `User` instead of `Organization`

- You installed the AWS Security Agent application into your desired GitHub organization and correctly set the `GitHub Account Type` to `Organization` but left the `Organization Name` field blank or entered an incorrect organization name that does not match the organization you installed the application into

**Solution:**

1. Go to GitHub and uninstall the app from the organization. For more information, see the section called “Step 1: Uninstall the AWS Security Agent GitHub App from GitHub”.
2. Go back to the integrations page and restart the integration process by clicking on `Add Integrations`, install and authorize the app into your desired GitHub organization once again.
3. Select `Organization` from the dropdown of `GitHub account type`
4. Make sure the `Organization Name` you input is the EXACT same as the one you installed the application into.
5. Click the Connect button to create your GitHub organization integration.

## Access Denied: Insufficient permissions to install GitHub App into organization

When you attempt to install the AWS Security Agent application into your desired GitHub organization, you will see two different messages on the button in the installation page.

- An organization `Member` will see `Authorize & Request`
- An organization `Owner` will see `Install & Authorize`

You can verify whether you are a `Member` or an `Owner` of the GitHub organization by following the below steps.

1. Go to github.com
2. Click on your profile in the top right of the website
3. Navigate to `Organizations` on the dropdown menu and click it
4. Find the organization you wish to install AWS Security Agent into from the list of organizations, it will specify whether you are a `Member` or an `Owner` next to the organization name.

Possible solutions:

- Have an owner approve your installation request BEFORE you try to create the integration
- Have an owner update your role in the GitHub organization from a `Member` to an `Owner` and restart the integration process again

## Agent cannot connect to endpoint during a penetration test

If the penetration test agent is unable to make calls to the configured target URL or fails to successfully navigate the target endpoint:

- If your endpoint makes calls to domains outside the configured target URL, verify the additional domains are added as **Accessible URLs** in your pentest configuration
- Penetration testing is currently only available for HTTP/HTTPS endpoints serving traffic on ports 80 or 443
- If you have a WAF configured, check that you WAF is not blocking penetration test traffic. You can allowlist penetration test traffic by `User-Agent` header, which will be set to `securityagent` by default

## Getting additional help

If you continue to experience issues after trying these troubleshooting steps:

- Check the AWS Security Agent service status page
- Contact AWS Support for assistance with complex configuration issues

# Log in to the AWS Security Agent Web App

You should log in to the AWS Security Agent Web App to complete tasks such as:

- Perform a design review
- Perform a penetration test

## Access methods

AWS Security Agent provides different methods to access the web application, depending on how your organization configured access during initial setup and whether you have AWS Console access.

**IAM Identity Center (SSO)** - If your organization enabled IAM Identity Center, you can log in using your SSO credentials. You must be assigned to at least one Agent Space in IAM Identity Center before you can access the web application.

**Admin access (IAM)** - If you have AWS Console access with appropriate permissions, you can launch the web application through the admin access link on any Agent Space page. This method provides authentication through your existing Console session.

> **Note**
>
> The primary access method your organization uses was determined during initial AWS Security Agent setup. For information on configuring user access and assignments, see the section called “Grant users access to the AWS Security Agent web app”.

# Log in with IAM Identity Center (SSO)

If your organization configured IAM Identity Center access, you can log in to the web application using your SSO credentials through multiple entry points.

## Prerequisites

- You have been assigned to at least one Agent Space in IAM Identity Center
- You have your IAM Identity Center SSO credentials

# Access the web application

Choose one of the following methods based on your needs:

**To view all Agent Spaces you're assigned to:**

1. In the AWS Security Agent console, navigate to **Settings**.
2. Locate the **Web application domain** and copy the URL.

> **Tip**
>
> Bookmark this URL for easy access. This is the universal entry point to view all Agent Spaces you're assigned to.

3. Navigate to the web application URL.
4. Enter your IAM Identity Center credentials when prompted.
5. After authentication, you'll see a list of all Agent Spaces you're assigned to.
6. Select the Agent Space you want to work in.

**To access a specific Agent Space directly (requires AWS Console access):**

1. Log into the AWS Management Console.
2. Navigate to the AWS Security Agent console.
3. Navigate to the Agent Space you want to access.
4. Click one of the following:
   - **Launch web application** button on the Agent Space overview page
   - **Launch web app** button in the Code review section
   - **Launch web app** button in the Penetration testing section
5. Enter your IAM Identity Center credentials when prompted.
6. After authentication, you'll be taken directly to that Agent Space in the web application.

> **Note**
>
> If you don't have AWS Console access, use the web application domain from the Settings page to access all your assigned Agent Spaces.

# Log in with admin access (IAM)

If you have AWS Console access with appropriate AWS Security Agent permissions, you can launch the web application through the admin access link with automatic authentication.

## Prerequisites

- You are logged into the AWS Management Console
- You have appropriate permissions to access AWS Security Agent resources

## Access the web application

Choose one of the following methods:

**To access a specific Agent Space:**

1. Log into the AWS Management Console.
2. Navigate to the AWS Security Agent console.
3. Navigate to the Agent Space you want to access.
4. Click the **Admin access** button on the Agent Space overview page.
5. The web application opens in a new tab with automatic authentication for that Agent Space.

> **Note**
>
> The admin access button is always available to users with AWS Console access, regardless of whether your organization uses IAM Identity Center as the primary access method.

# Understand the resource hierarchy and lifecycle

AWS Security Agent organizes security testing resources in a hierarchical structure that determines what's shared across your organization and what's scoped per application. Understanding this structure helps you configure AWS Security Agent effectively and know where to find and manage different resources.

## What's shared across your organization

Some resources in AWS Security Agent are configured once at the organizational level and apply across all your applications and Agent Spaces. These tenant-level resources provide consistency and reduce duplicate configuration work.

| Resource | What it is | Why it's shared |
|---|---|---|
| **Security requirements** | Organizational security standards that define what AWS Security Agent validates during design and code reviews | Your security policies apply to all applications. Define them once and AWS Security Agent enforces them everywhere. |
| **GitHub integrations** | Registered GitHub organizations or user accounts authorized to connect with AWS Security Agent | Register your GitHub organization once, then connect specific repositories to any Agent Space as needed. |
| **IAM Identity Center configurations** | SSO settings that control how users access AWS Security Agent | Centralized identity management applies across all Agent Spaces in your organization. |

**Important**

Changes to security requirements affect all future design reviews and code reviews across all Agent Spaces. Existing reviews are not affected.

# What's scoped per Agent Space

Each Agent Space represents a distinct application or project you want to secure. Resources at the Agent Space level are scoped to that specific application, allowing different teams to work independently with their own configurations and assessments.

| Resource | What it is | Why it's scoped per application |
|---|---|---|
| **Penetration test configurations** | Test configurations for specific features, API endpoints, or functiona lity within your application | Each application has unique targets, authentication methods, and scope boundaries specific to that applicati on. |
| **Design reviews** | Individual architectural security assessments of design documents | Each application has its own architecture and design documents that are assessed independently. |
| **Connected repositories** | GitHub repositories linked to this Agent Space | Different applications use different repositories. Connecting them at the Agent Space level keeps application boundaries clear. |
| **Code review settings** | Configuration of which connected repositories have automated code review enabled | Teams control which repositories receive automated security feedback based on their application's needs. |
| **Penetration test remediation settings** | Configuration of which connected repositories can receive automated fix pull requests for penetration testing findings | Teams control where AWS Security Agent can submit code changes based on their application's workflow. |
| **User assignmen ts** | Users who have access to this specific Agent Space | Teams only see security assessmen ts for applications they're responsib le for, keeping work organized and focused. |

> **Tip**
>
> We recommend creating one Agent Space per application or project to maintain clear boundaries between teams and organize security assessments effectively.

# How GitHub repositories fit into the hierarchy

GitHub repositories are integrated through a multi-step process that connects organizational resources to specific applications:

1. **Register at the tenant level** - Authorize the AWS Security Agent GitHub App for your GitHub organization or user account once
2. **Connect at the Agent Space level** - Select specific repositories to connect to each Agent Space
3. **Configure usage per repository** - Enable specific capabilities for each connected repository:
   - **Code review** - Automated security analysis of pull requests
   - **Penetration testing context** - Application understanding from source code during penetration tests
   - **Penetration test remediation** - Automated pull requests with vulnerability fixes for penetration testing findings

A single repository can be connected to multiple Agent Spaces with different capabilities enabled in each one.

# Key differences between security capabilities

Each security capability in AWS Security Agent follows a different workflow model based on how security teams use it.

## Penetration testing: Reusable configurations with independent executions

Penetration tests use a configuration-and-run model that supports iterative security testing:

- **Create once, execute many times** - Define a configuration for a specific target (API endpoint, feature area) with scope boundaries, authentication, and test parameters

- **Independent executions** - Execute the same configuration multiple times as you improve security. Each execution is independent and generates new findings

This model supports continuous security validation as you develop and deploy improvements.

## Design reviews: One-off assessments with cloning

Design reviews are independent assessments that don't follow a reusable configuration model:

- **Single assessment** - Each design review analyzes uploaded documents once against your organization's security requirements
- **Cannot re-run** - Design reviews are not reusable. You cannot re-run the same review
- **Clone for updates** - Clone an existing design review to create a new review with the original documents pre-loaded, allowing you to update documents and run a new analysis

This model supports point-in-time architectural security assessments.

## Code reviews: Automatic and independent

Code reviews integrate into your GitHub workflow without manual configuration per review:

- **Automatic trigger** - AWS Security Agent automatically analyzes pull requests in repositories where code review is enabled
- **Independent reviews** - Each pull request receives its own independent security analysis
- **Findings in GitHub** - Security findings appear as comments on pull requests, not in the Security Agent Web Application

This model embeds security feedback directly into your development workflow.

# Understanding resource relationships

The hierarchy determines where you configure and access different resources:

**In the AWS Management Console:**

- Configure tenant-level resources (security requirements, GitHub integrations, IAM Identity Center)

- Create and manage Agent Spaces
- Configure Agent Space settings (connected repositories, code review enablement, penetration test remediation)

**In the Security Agent Web Application:**

- Create and manage penetration test configurations and test executions
- Create and manage design reviews
- View findings from penetration tests and design reviews

**In GitHub:**

- View code review findings as pull request comments
- Receive automated remediation pull requests for penetration testing findings (when enabled in the Agent Space)

> **Note**
>
> Code review findings and penetration test remediation pull requests appear in GitHub. Penetration test and design review findings appear in the Security Agent Web Application.

# Create a design review

Assess your design documents against organization security requirements by uploading files for AWS Security Agent to review. Design reviews help identify security issues early in the development lifecycle, enabling you to address architectural concerns when they are most cost-effective to resolve.

AWS Security Agent analyzes your design documents against your organization's security requirements, providing detailed security findings to improve security posture before implementation begins.

In this procedure, you'll create a design review by uploading design files for analysis.

## Prerequisites

Before you begin, ensure you have:

- Access to the AWS Security Agent web application
- Design documents ready for upload (DOC, DOCX, JPEG, MD, PDF, PNG and TXT)
- Each file must be 2MB or smaller, with a combined total of 6MB across all files
- Understanding of which security requirements are enabled for your organization

## Step 1: Start creating a design review

Navigate to the design review creation page in the Agent Web App.

1. Log in to the AWS Security Agent web application.
2. Navigate to the **Design reviews** section.
3. Click **Create Design Review**.

> **Tip**
>
> You can view your organization's enabled security requirements by navigating to the **Security requirements** page in the AWS Security Agent console. Click on any enabled requirement to view its details. These requirements are used to analyze your design files.

## Step 2: Name your design review

Provide a descriptive name that helps identify the purpose and scope of this design review.

1. In the **Design review name** section, locate the **Name** field.
2. Enter a descriptive name for your design review.

> **Note**
>
> The name should clearly identify the project, feature, or component being reviewed. Maximum 80 characters.

## Step 3: Upload design files

Upload the design documents you want AWS Security Agent to analyze for security compliance.

1. In the **Files to review** section, review the file requirements:

> **Important**
>
> A maximum of 5 files may be uploaded per design review. Each file must be 2MB or smaller, with a combined total of 6MB across all files. Supported formats: DOC, DOCX, JPEG, MD, PDF, PNG and TXT.

2. Upload your files using one of these methods:
   a. **Drag and drop** – Drag files directly into the file dropzone area
   b. **Browse** – Click **Choose files** to browse and select files from your computer
3. Verify that all required files are uploaded.

> **Tip**
>
> For best results, include architecture diagrams, design specifications, and technical documentation that describe your system's security-relevant components and data flows.

## Step 4: Initiate the design review

After configuring all required information, initiate the security analysis of your design documents.

1. Review all uploaded files and settings to ensure accuracy.
2. Click **Start design review** at the bottom of the page.
3. AWS Security Agent will analyze your design documents against enabled security requirements.

> **Note**
>
> The design review process typically completes within minutes, depending on the number and size of files uploaded. You'll receive security findings based on your organization's security requirements.

## Next steps

After starting your design review:

- Monitor the review progress in the Agent Web App
- Review security findings
- Share findings with your development team
- Address identified security findings in your design
- Update design documents and resubmit if needed

## Review findings from a design review

Review findings help you understand which security requirements are met, which need attention, and what actions to take to improve your design's security posture before implementation begins.

In this procedure, you'll learn how to access, filter, and interpret design review findings to address security findings effectively.

### Prerequisites

Before you begin, ensure you have:

- A completed design review
- Access to the AWS Security Agent web application
- Familiarity with your organization's enabled security requirements

## Step 1: Access the design review

Navigate to your design review to view the findings and summary information.

1. Log in to the AWS Security Agent web application.
2. Navigate to the **Design reviews** section.
3. Select the design review you want to examine from the list.

> **Tip**
>
> The design review details page displays a summary of review status, completion date, and the number of files reviewed.

## Step 2: Review the findings summary

Examine the high-level summary to understand the overall security posture of your design.

1. Locate the **Summary** section near the top of the page.
2. Review the count for each compliance status category: **Compliant**, **Non-compliant**, **Insufficient data**, and **Not applicable**.

> **Note**
>
> The summary provides counts for each status type, helping you quickly assess the number of findings requiring attention. For detailed explanations of each status, see Step 4.

## Step 3: Filter and navigate findings

Use the filtering and search capabilities to focus on specific findings or compliance statuses.

1. In the **Review findings** section, locate the filter controls.
2. To filter by status:
   a. Click the status dropdown menu.
   b. Select a specific compliance status to view only findings with that status.
3. To search for specific security requirements:
   a. Enter keywords in the search field.
   b. Results update automatically as you type.
4. Use the pagination controls to navigate through multiple pages of findings.

> **Tip**
>
> Filter by **Non-compliant** status first to prioritize findings that require immediate attention in your design.

## Step 4: Understand compliance statuses

Each finding displays a compliance status that indicates how your design addresses a specific security requirement:

- **Compliant** – Your design meets the security requirement based on the analysis
- **Non-compliant** – Your design violates or inadequately addresses the security requirement
- **Insufficient data** – The uploaded files lack enough information to determine compliance
- **Not applicable** – The security requirement doesn't apply to your system design

> **Important**
>
> Focus on **Non-compliant** and **Insufficient data** statuses, as these require action. Address non-compliant findings by updating your design, and resolve insufficient data findings by uploading additional design documentation.

## Step 5: View finding details

Select individual findings to view detailed justification and remediation guidance.

1. In the findings table, click on a security requirement name.
2. Review the finding details, which include:
   - The specific security requirement being evaluated
   - A comment explaining why the finding received its compliance status, including specific details about what's missing or non-compliant
   - Recommended remediation guidance to address the finding
   - Links to your organization's internal documentation or standards for the security requirement

> **Note**
>
> The comment explains AWS Security Agent's reasoning with specific details. For insufficient data findings, the comment identifies what information is missing, such as "The design documents don't mention authentication mechanisms" or "No information found about data encryption at rest."

## Step 6: Address findings

Take action on findings that require attention to improve your design's security posture.

For **Non-compliant** findings:

1. Review the recommended remediation guidance.
2. Review any linked internal documentation for additional context.
3. Update your design documents to address the security requirement.
4. Document the changes you make for future reference.

For **Insufficient data** findings:

1. Read the comment carefully to understand what specific information is missing.
2. Create or update design documents with the missing details.
3. Prepare the updated files for resubmission.

## Next steps

After reviewing your design findings:

- Download the findings report as a CSV file for sharing with your team
- Update design documents to address non-compliant findings
- Create additional documentation for insufficient data findings
- Share findings with your development team for discussion
- Clone this design review to create a new review with the original documents pre-loaded, allowing you to update the name and run the analysis again to verify improvements
- Proceed with implementation for designs that meet compliance requirements

For more information about managing security requirements, see the section called “Manage security requirements”.

For more information about creating design reviews, see *Create a design review*.

# Create a penetration test

Set up automated penetration testing for your web applications by configuring test scope, target domains, and AWS resource access. Penetration tests help identify security vulnerabilities in running applications by simulating real-world attack scenarios against your verified domains.

AWS Security Agent performs comprehensive security testing against your web applications based on configured scope and permissions, providing detailed findings about exploitable vulnerabilities before attackers can discover them.

In this procedure, you'll create a penetration test by configuring test details, defining the test scope, and setting up required permissions.

## Prerequisites

Before you begin, ensure you have:

- Access to the AWS Security Agent web application
- At least one verified domain for testing
- IAM role with appropriate permissions for AWS Security Agent
- Understanding of your application's architecture and critical paths

## Start creating a penetration test

Navigate to the penetration test creation page in the Agent Web App.

1. Log in to the AWS Security Agent web application.
2. Navigate to the **Penetration tests** section.
3. Click **Create a penetration test**.

> **Tip**
>
> Only verified domains can be included in penetration tests. Ask your admin to verify the domain in AWS management console. See the section called "Enable an application domain for penetration testing".

# Name your penetration test

Provide a descriptive name that helps identify the purpose and scope of this penetration test.

1. In the **Penetration test name** field, enter a descriptive name for your penetration test.

   **Example**

   The name should clearly identify the application, environment, or component being tested. Maximum 100 characters.

# Configure penetration test scope

Define which domains and URL paths will be tested, and configure optional exclusions to control test boundaries.

## Add target domains

Specify the verified domains that will be actively tested for security vulnerabilities.

1. In the **Penetration test scope** section, locate **Target URLs**.
2. Expand the **Verified domains** section to view available domains.
3. Click in the **Target URL** field and enter a target domain URL.

   > ⚠ **Important**
   >
   > Only verified domains can be tested. The URL must match a domain you've previously verified in AWS Security Agent.

4. To add multiple target domains:
   a. Click **Add domain**.
   b. Enter each additional domain URL.
5. To remove a target domain, click **Remove** next to the domain URL.

> **Tip**
>
> For best results, include all domains that are part of your application's user flow, including subdomains for APIs, authentication services, and content delivery.

## Exclude risk types (optional)

Choose specific risk categories to exclude from testing if they're not applicable to your application.

1. Locate the **Exclude risk types** field.
2. Click the dropdown to view available risk types.
3. Select one or more risk types to exclude from the penetration test.

> **Note**
>
> Excluding risk types limits the scope of testing. Only exclude risk types that are not relevant to your application or that you want to test separately.

## Add out-of-scope URL paths (optional)

Specify URL paths that should not be tested during the penetration test.

1. Locate the **Out-of-scope URLs** section.
2. Click in the input field and enter a URL path to exclude (for example, `/admin/delete` or `/api/reset`).
3. To add multiple out-of-scope paths:
   a. Click **Add URL**.
   b. Enter each additional path.
4. To remove a path, click **Remove** next to the path.

> **Warning**
>
> Out-of-scope paths will not be tested for vulnerabilities. Ensure you only exclude paths that should not be accessed during testing, such as destructive operations or sensitive administrative functions.

## Add accessible domains (optional)

Specify domains that are required for the test but are not targets for vulnerability testing.

1. Locate the **Accessible URLs** section.
2. Click in the input field and enter a domain that should be accessible during testing.

> **Note**
>
> Add accessible domains for third-party services (such as Okta, Auth0, Stripe) that are outside your target domain. This is required so AWS Security Agent can access these URLs for login and navigation during testing. AWS Security Agent does NOT penetration test these domains—they are used solely for access purposes.

3. To add multiple accessible domains:
   a. Click **Add URL**.
   b. Enter each additional domain.
4. To remove a domain, click **Remove** next to the domain.

# Configure IAM Role

Select the pre-configured service role for this penetration test. AWS Security Agent uses an Agent Space-based permission model where administrators configure IAM roles when setting up your Agent Space. You'll select from roles that are already configured and ready to use.

1. In the **Permissions** section, locate the **Service roles** dropdown.
2. Select the IAM role that grants AWS Security Agent access to required AWS resources.

> **Important**
>
> The selected IAM role must have permissions to access VPC resources, CloudWatch Logs, and any other AWS services needed for the penetration test. Verify that the role has the correct trust relationship with AWS Security Agent.

3. Locate the **CloudWatch log group** dropdown.
4. Select the log group where penetration test logs will be stored. (optional)

> **Note**
>
> The selected CloudWatch log group will store detailed logs of the penetration test execution, including requests made, responses received, and vulnerabilities discovered. If you don't select a log group, a new CloudWatch log group will be automatically created with the `/aws/securityagent` prefix to store the penetration test logs.

## Automatic code remediation

Select the **Enable automatic remediation** checkbox.

> **Important**
>
> To remediate security findings in your source code repositories, AWS Security Agent may submit pull requests to your repositories. The pull requests may be visible to all users who have read access to the repositories.

## Configure VPC resources (optional)

If your target domains are private and hosted within a VPC, configure the VPC settings where AWS Security Agent should run penetration tests. This step is only necessary for applications that are not publicly accessible.

> **Note**
>
> Skip this step if your target domains are publicly accessible. VPC configuration is only required for testing private applications hosted within an Amazon Virtual Private Cloud.

Choose the VPC, subnets, and security groups for the penetration test environment.

1. In the **VPC** section, locate the **VPC ID** dropdown.
2. Select the VPC where your target domains are hosted.

> **Important**
>
> The selected VPC must contain the target domains you specified in Step 3. Ensure the VPC has appropriate routing and network configuration to allow AWS Security Agent to access your applications.

3. Locate the **Subnets** dropdown.
4. Select one or more subnets where the penetration test should run.

> **Note**
>
> Choose subnets that have network access to your target applications. The penetration test will execute from resources deployed in these subnets.

5. Locate the **Security group** dropdown.
6. Select the security group that controls network access for the penetration test.

> **Important**
>
> The selected security group must allow outbound traffic to your target domains and any accessible domains. Ensure the security group rules permit the necessary network access for comprehensive testing.

# Configure authentication credentials (optional)

If your target domains require authentication, provide credentials to allow AWS Security Agent to access protected areas of your application during penetration testing. This step is only necessary for applications that require user authentication.

> **Note**
>
> Skip this step if your target domains do not require authentication or if all areas you want tested are publicly accessible. Configure credentials only when you need AWS Security Agent to test authenticated sections of your application.

## Add credentials

Provide authentication credentials that AWS Security Agent will use to access your application.

1. In the **Credential #1** section, select a credential input method:
   - **Input credentials** - Enter your credentials directly into AWS Security Agent.
   - **Advanced setting** - For sensitive credential information, use advanced options such as AWS Secrets Manager or AWS Lambda functions. See the section called “Provide authentication credentials for penetration testing” for details.

     > **Tip**
     >
     > For production environments or sensitive credentials, we recommend using the advanced setting option to securely reference credentials stored in AWS Secrets Manager or Systems Manager Parameter Store.

## Enter credential details

Provide the username and password for the authenticated account.

1. In the **User name** field, enter the username for authentication.
2. In the **Password** field, enter the password for authentication.

> **Important**
>
> Ensure the credentials you provide have appropriate access levels for the areas you want tested. The credentials should represent a typical user's access level rather than administrative privileges.

## Select access domain

Specify which target domain will use these credentials for authentication.

1. In the **Access domain** dropdown, select the domain where these credentials will be used.

> **Note**
>
> If you have multiple target domains that require different credentials, you can add additional credential sets by clicking **Add another credential** after completing this credential configuration.

## Configure agent login prompt (optional)

Provide instructions to guide AWS Security Agent through your application's authentication process.

1. Expand the **Agent login prompt** section if your authentication flow requires specific instructions.
2. Enter detailed instructions describing how to access your application using the provided credentials.

> **Note**
>
> The agent login prompt is useful for complex authentication flows, multi-step login processes, or applications with non-standard login procedures. Include step-by-step instructions such as "Navigate to /login, enter username in the 'Email' field, enter password, and click 'Sign In'."

## Add multiple credentials (optional)

If your application requires multiple sets of credentials or different domains need separate authentication, add additional credential sets.

1. After completing the first credential configuration, click **Add another credential**.
2. Repeat the credential configuration steps for each additional credential set.
3. To remove a credential set, click **Remove** next to the credential header.

> **Tip**
>
> Configure multiple credentials when testing different user roles, accessing multiple authenticated domains, or verifying role-based access controls in your application.

# Attach additional resources (optional)

Provide supplementary resources to help AWS Security Agent conduct more thorough and accurate penetration testing. Additional resources can include architecture diagrams, API documentation, configuration files, GitHub repositories, or S3-hosted materials that give context about your application.

> **Note**
>
> Additional resources are optional but recommended. Providing comprehensive information about your application helps ensure thorough test coverage, reduces false positives, and delivers more actionable results.

## Add resources to the penetration test

Select existing resources or upload new files that will help guide the penetration test.

1. In the **Connected resources** section, you can:
   - Click **Select from available** to choose from resources already connected to AWS Security Agent (such as GitHub repositories or S3 buckets).
   - Click **Upload** to add new files directly from your local system.

> **Tip**
>
> Useful resources include API documentation, architecture diagrams, OpenAPI/Swagger specifications, configuration files, authentication flow diagrams, and any other materials that describe your application's structure and behavior.

## Select from available resources

Choose from resources that are already integrated with AWS Security Agent.

1. Click **Select from existing resources**.
2. Browse the list of available resources from connected sources such as:
   - GitHub repositories, under the **GitHub repositories tab**
   - S3 buckets
   - Previously uploaded files
   - Documentation repositories
3. Select the resources you want to include in the penetration test.
4. Click **Add to penetration test** to attach the selected resources.

#### Example

We recommend selecting and adding relevant GitHub repositories to your pentest, so AWS Security Agent can develop an understanding of your application context, and generate ready-to-implement code fixes through pull requests (when enabled)

> **Note**
>
> Resources selected from available sources remain synchronized with their original location. If you update a GitHub repository or S3 file, the penetration test will use the updated version.

## Upload new resources

Upload files directly from your local system or provide plain text content to AWS Security Agent.

1. Click **Upload**.
2. Choose one of the following input methods:
   - **Upload local files** - Select one or more files from your local system.
   - **Paste plain text** - Type or paste text content directly into the input field. Click **Upload**.
3. Then click **Add** to complete uploading.
4. The uploaded resources will appear in the **Connected resources** table.

> **Tip**
>
> Use the plain text option when you want to quickly provide API endpoint lists, URL patterns, test instructions, or other text-based information without creating a separate file.

> **Important**
>
> Ensure uploaded files and pasted content do not contain sensitive information such as production credentials, private keys, or personally identifiable information (PII). Use sanitized versions of configuration files and documentation.

## Connect existing resources

Existing resources can be from what you've previously uploaded to AWS Security Agent, from your S3 bucket, and your integrated GitHub repositories. Click **Select from existing resources** to select them.

## Manage connected resources

Review, organize, and remove resources attached to the penetration test.

The **Connected resources** table displays all resources included in the penetration test with the following information:

- **Name** - The filename or resource identifier
- **Type** - The resource category (Uploaded files, S3 resources, GitHub repositories, etc.)

To manage resources:

1. Select one or more resources using the checkboxes.
2. Click **Remove from penetration test** to detach selected resources.

> **Note**
>
> You can sort the table by Name or Type by clicking the column headers. This helps organize resources when working with many files.

## Create the penetration test

Finalize and launch your penetration test configuration.

After configuring all settings, you're ready to create the penetration test.

1. Review all configuration sections to ensure accuracy.
2. Choose one of the following options:
   - Click **Create penetration** to save the configuration without running it immediately.
   - Click **Create and execute** to save the configuration and immediately start the penetration test.
   - Click **Cancel** to discard the penetration test configuration.

> **Important**
>
> Before running a penetration test, verify that:
>
> - All target domains are correctly verified and accessible
> - IAM roles have appropriate permissions
> - Out-of-scope paths are properly configured to prevent testing destructive operations
> - You have authorization to perform security testing on all target domains

> **Note**
>
> After the penetration test starts, you can monitor its progress from the **Penetration test runs** section. The test may take several hours depending on the scope and complexity of your application.

# Review findings from a penetration test

Monitor pentest execution in real time on the Penetration Test Logs page after AWS Security Agent starts a pentest. AWS Security Agent logs every action during the pentest. After completion, review the pentest summary, which includes application overview, coverage with identified endpoints, and risk assessment of security findings.

Evaluate security findings to address application vulnerabilities. Each finding contains impact assessment, severity rating, supporting evidence and remediation pull request details (when automatic code remediation is enabled).

## Prerequisites

Before you begin, ensure you have:

- A completed or in-progress penetration test run
- Access to the AWS Security Agent web application

## Step 1: Access the penetration test run

Navigate to your penetration test run to view overview, logs and findings pages.

1. Log in to the AWS Security Agent web application.
2. Navigate to the **Penetration tests** section.
3. Select the penetration test run you want to examine from the list.

> **Tip**
>
> The penetration test details page displays a summary of test status, completion date, and the number of findings identified.

## Step 2: Monitor test progress

Track the progress of your penetration test run using the step indicator at the top of the page.

1. Locate the horizontal step indicator below the page header.
2. Review the status of each testing phase:
   - **Preflight** – Initial setup and connectivity checks
   - **Static analysis** – Code and configuration analysis
   - **Pentests** – Runtime testing and vulnerability scanning
   - **Finalizing** – Final validation and report generation

> **Note**
>
> Each step displays a status indicator (Complete, In progress, or Pending). Findings are discovered and validated throughout the testing process, with new vulnerabilities appearing as each phase completes.

## Step 3: Navigate to the penetration test run overview tab

1. Run Summary section provides test status, duration and other high level details. It also provides a dashboard of security findings categorized by severity level and risk-types
2. Application overview by AWS Security Agent provides a summary of the penetration test run
3. Discovered endpoints by AWS Security Agent provides a list of all endpoints discovered and tested by the AWS Security agent during the pentest run

## Step 4: Navigate to the penetration test logs tab

Access detailed logs of all actions AWS Security Agent executed during the pentest.

1. The actions are categorized by action type and risk-types.
2. Click on a specific action to view detailed logs:
   - **Testing Summary** – High-level summary of the agent actions and results
   - **Penetration test logs** – Detailed logs of all testing activities

> **Note**
>
> Validator actions provide logs that validate findings in each category

> **Note**
>
> The Findings tab displays a split view with the findings list on the left and selected finding details on the right.

## Step 5: Navigate to the findings

Each finding in the list displays key information to help you quickly assess its importance.

> **Note**
>
> Findings with Low agent confidence and False Positives are hidden by default. You can view them by disabling the toggle **Hide False Positives**.

Review the information displayed on each finding card:

- **Finding name** – The title and identifier for the vulnerability
- **Confidence badge** – Indicates the agent's confidence level in the finding (High, Medium, or Low)
- **Severity badge** – Shows the risk level with color coding:
  - **Critical** (red) – Requires immediate action; exploitation could lead to system compromise
  - **High** (red) – Requires prompt attention; exploitation could result in significant security impact
  - **Medium** (orange) – Should be addressed in a reasonable timeframe; contributes to overall security risk
  - **Low** (yellow) – Can be addressed as part of regular maintenance; minimal immediate risk
  - **Informational** (blue) – For informational purposes; minimal to no immediate risk
- **Last update timestamp** – Shows when the finding was last modified or validated
- **Description preview** – Brief summary of the vulnerability

> **Important**
>
> Prioritize findings with **Critical** or **High** severity badges and **High** confidence levels, as these represent validated vulnerabilities requiring immediate remediation.

## Step 6: Review finding details

Select individual findings to view comprehensive information about each vulnerability.

1. Click on a finding name in the left panel to display its details in the right panel.
2. Review the validation status at the top of the details panel:

> **Note**
>
> If a finding displays the Unknown "This finding is not validated by AWS Security Agent yet," it means the vulnerability detection is still being confirmed. These findings may require manual verification.

3. Review the key attributes displayed at the top:
   - **Agent confidence** – The confidence level AWS Security Agent has in this finding
   - **Severity** – The risk level with a color-coded badge
   - **Finding logs** – Click "Trace actions & logs" to view detailed execution logs and evidence
   - **Risk type** – The category or type of security risk (e.g., Authentication Bypass, SQL Injection)
4. Expand the **Description** section to read:
   - A detailed explanation of the vulnerability
   - How the vulnerability works
   - Why it represents a security risk
   - The potential impact on your application
5. Expand the **Risk Reasoning** section to understand the severity calculation:
   - CVSS (Common Vulnerability Scoring System) metrics breakdown
   - Attack Vector (AV) – How the vulnerability can be exploited
   - Attack Complexity (AC) – How difficult the exploit is
   - Privileges Required (PR) – What access level is needed

- User Interaction (UI) – Whether user action is required
- Scope (S) – Whether the vulnerability affects other components
- Confidentiality, Integrity, and Availability impacts

6. Expand the **Steps to reproduce** section to view:
   - Detailed technical steps to recreate the vulnerability
   - Request and response examples
   - Specific parameters or conditions that trigger the issue

> **Tip**
>
> Use the "Trace actions & logs" link to access the complete evidence package, including HTTP requests, responses, and exploitation attempts that demonstrate the vulnerability.

## Step 7: Interpret CVSS metrics

Understanding CVSS metrics helps you assess the true severity and prioritize remediation efforts.

When reviewing the **Reasoning** section, pay attention to these key metrics:

- **Attack Vector (Network/Adjacent/Local/Physical)** – Indicates how remotely the attack can be executed
- **Attack Complexity (Low/High)** – Shows whether specialized conditions are required to exploit
- **Privileges Required (None/Low/High)** – Identifies what access level an attacker needs
- **User Interaction (None/Required)** – Determines if the exploit needs user involvement
- **Confidentiality/Integrity/Availability Impact (None/Low/High)** – Measures the impact on your system's security

> **Important**
>
> Findings with Network attack vector, Low complexity, and High confidentiality/integrity impact represent the most dangerous vulnerabilities requiring immediate attention.

# Step 8: Prioritize and address findings

Take action on findings to remediate vulnerabilities and improve your application's security posture.

For **Critical** and **High** severity findings with **High** confidence:

1. Review the Description and Steps to reproduce sections thoroughly.
2. Access the detailed logs via the "Trace actions & logs" link to gather complete evidence.
3. Access ready-to-implement code fixes through one of these methods:
   - For automatic remediation: Use the pull request link in the remediation section
   - For manual requests: Click 'Remediation Code' on the findings page to request a pull request **Prerequisites:**
   - Admin must enable code remediation for GitHub repositories in the AWS Security Agent console
   - Repositories must be included in your pentest configuration
4. Plan for a follow-up penetration test to verify the fix is effective.

For **Medium** and **Low** severity findings:

1. Prioritize based on your risk tolerance and business context.
2. Include remediation tasks in your regular development sprint planning.
3. Consider whether multiple low-severity findings together create higher risk.
4. Document any accepted risks with appropriate justification.

> **Important**
>
> Do not ignore low-severity findings. Multiple low-severity vulnerabilities can often be chained together to create more serious exploits, especially when combined with social engineering or physical access.

## Step 9: Track remediation progress

Use the findings interface to track which vulnerabilities have been addressed and which require further action.

1. As you work on remediation, refer back to the Steps to reproduce section to verify your fixes.
2. Document your remediation approach for each finding for future reference and compliance audits.

> **Tip**
>
> Maintain a remediation log that maps each finding to its resolution, including the code changes, configuration updates, or architectural decisions made to address the vulnerability.

## Next steps

After reviewing your penetration test findings:

- Prioritize critical and high-severity findings with high confidence for immediate remediation
- Create tracking tickets in your issue management system with links to finding details and evidence
- Implement fixes and security controls to address identified vulnerabilities
- Monitor the penetration test run progress indicator for newly discovered vulnerabilities
- Schedule a follow-up penetration test to verify that vulnerabilities have been properly remediated
- Update your application security testing process and threat model based on findings
- Review CVSS metrics to understand your application's overall security posture

For more information about performing penetration tests, see *Create a penetration test*.

For more information about understanding the Security Agent lifecycle, see *Understand the resource hierarchy and lifecycle*.

# Provide authentication credentials for penetration testing

Provide credentials to enable AWS Security Agent to test authenticated areas of your web applications. Without credentials, the agent can only test publicly accessible pages and APIs.

## Configure authentication credentials

1. In the penetration test creation workflow, locate the **Authentication credentials - Optional** section.
2. In the **Credential #1** section, choose your credential input method:
   - **Input credentials** - Enter credentials directly. Best for development and testing environments.
   - **Advanced setting** - Use AWS-native credential management. Recommended for production environments and sensitive credentials.

### Advanced options

If you select **Advanced setting**, you can choose from three credential strategies:

- **IAM role assumption** - For applications using AWS Cognito or IAM authentication
- **AWS Secrets Manager** - For secure credential storage with encryption and rotation
- **Lambda function** - For dynamic credential generation or complex authentication flows

## Input credentials directly

1. Select **Input credentials**.
2. Enter the **User name** and **Password**.
3. In the **Access URL** dropdown, select the URL where these credentials will be used. This must be selected from the list of target endpoints.
4. (Optional) In the **2FA - optional** field, provide a TOTP secret for applications that require two-factor authentication. You can either:
   - Enter the TOTP secret directly (for example, `JBSWY3DPEHPK3PXP`), or enter the full `otpauth://totp/` URI (for example, `otpauth://totp/Example:user@example.com?secret=JBSWY3DPEHPK3PXP&issuer=Example`).
   - Click the upload icon to upload a QR code image from your authenticator app setup page. The QR code is scanned locally and the TOTP URI is extracted automatically.

When a TOTP secret is provided, the agent automatically generates fresh one-time codes and enters them when a 2FA prompt is detected during login.

5. (Optional) Expand **Agent Space login prompt** to provide specific login instructions if your application has a complex authentication flow.

> **Important**
>
> Use test accounts with representative access rather than personal or administrative accounts.

## Use advanced setting

1. Select **Advanced setting**.
2. In the **User access strategy** dropdown, choose one of the following:

### Select available IAM role for agent to assume

Use this option for applications using AWS Cognito, API Gateway with IAM authentication, or other AWS-native authentication systems. The IAM role must have a trust relationship allowing AWS Security Agent to assume it and permissions to access your application's authentication system.

### Select static credential from connected AWS Secrets Manager

Use this option to retrieve credentials securely from AWS Secrets Manager with encryption, rotation, and access auditing.

The IAM role must have `secretsmanager:GetSecretValue` and `secretsmanager:DescribeSecret` permissions.

Use the **Agent Space login prompt** to provide detailed instructions on how to interpret and use the credentials stored in the secret. You may use any format to store your secret, as the agent will dynamically interpret the format using these instructions.

For example, if the agent is to submit a username/password login form at https://example.com/login, you may format your secret as JSON with `username` and `password` fields. If the application

requires TOTP-based 2FA, include a `totpSecret` field with either the TOTP secret directly or a full `otpauth://totp/` URI:

```
{
  "username": "test-user",
  "password": "secure-password-here",
  "totpSecret": "JBSWY3DPEHPK3PXP"
}
```

Then, configure the authentication instructions: . Set **Access URL** to `https://example.com` (or any other URL selected from the list of target endpoints). . Enter the following into **Agent Space login prompt**: "Navigate to https://example.com/login and enter the provided username and password into the form."

As another example, if you instead have an API key to be provided in an HTTP header, you may store it as plaintext:

```
"api-key-here"
```

Then, configure the authentication instructions: . Enter the following into **Agent Space login prompt**: "Set the X-API-Key header to the provided API key for all requests."

> **Important**
>
> Only TOTP-based 2FA is supported. SMS, email, push notifications, hardware keys, and OAuth authentication are not supported.

## Select available Lambda function to retrieve credentials dynamically

Use this option for complex authentication systems, dynamic credential generation, or integration with external identity providers.

The IAM role must have `lambda:InvokeFunction` permissions and the function must complete within 30 seconds.

Like with Secrets Manager, the agent will dynamically interpret your Lambda function's output using any login instructions provided in the **Agent Space login prompt**. Refer to the section called "Select static credential from connected AWS Secrets Manager" for examples of how to format the output of your Lambda function and supported authentication types.

## Configure multiple credentials

To test different user roles or authentication systems:

1. Click **Add another credential**.
2. Configure the additional credential using either input method.
3. To remove a credential, click **Remove** in the credential section.

# Remediate a penetration test finding

When viewing the findings for a penetration test, you can request AWS Security Agent attempt to remediate a finding. For private GitHub repositories, AWS Security Agent opens a pull request with the proposed fix. For public repositories, the remediation is available as a downloadable diff file that you can apply locally.

You must enable finding remediation in the AWS Management Console. (See the section called "Enable users to start remediation of penetration test findings".) Users can start remediation for a specific finding from the AWS Security Agent Web App.

> **Note**
>
> Code remediation is currently available in only us-east-1.

## Prerequisites

Before you begin, ensure you have:

- A completed or in-progress penetration test run
- Access to the AWS Security Agent web application
- Familiarity with your application's architecture and security requirements

## Step 1: Enable or disable automatic remediation

You can configure code remediation options when you create or modify a penetration test. If you enable automatic remediation, AWS Security Agent will automatically attempt to remediate the associated GitHub repositories if the Agent confirms a finding during the pentest. You can also

manually start code remediation. . In the view to edit **Penetration test details**, in the **Automatic code remediation** section, enable or disable code remediation.

## Step 2: Select repositories for code remediation

1. Click **Next** all the way to the last step **Additional learning resources**.
2. Choose **Select from resources**.
3. Choose **GitHub repositories**.
4. Select the repositories that you want for code remediation.
5. Save the penetration test.
6. You can see the successfully associated repositories under the **Penetration test learning resources** tab.

## Step 3: Start a penetration test and view findings

Run the penetration test to detect findings. For more information, see the section called “Review findings from a penetration test”.

## Step 4: Start and view code remediation

1. Navigate to the finding.
2. If you’ve enabled automatic code remediation, a code remediation will be started once AWS Security Agent confirms a finding.
3. If you want to manually start a code remediation, click the **Remediate code** button.
4. In the **Code Remediation** section of the finding, you can view the code remediation status and links to the pull requests. If the GitHub repository is public, the code remediation is available as a downloadable file instead of a pull request. You can run `git apply /path/to/code_remediation_changes.diff` to apply the change to your repository locally.

# Review code security findings in GitHub

After enabling code review for your repositories, AWS Security Agent automatically analyzes pull requests and posts security findings directly in GitHub. This allows developers to address security issues within their normal workflow without leaving the pull request.

## How code review works in GitHub

When you submit a pull request in a repository with code review enabled, AWS Security Agent automatically begins analysis.

1. **Pull request analysis trigger** - Code review is triggered when a pull request is marked as "Ready for review" in repositories where you've enabled the code review capability. Draft pull requests are not analyzed.
2. **Analysis acknowledgment** - When AWS Security Agent begins analyzing your pull request, it posts an initial comment: "AWS Security Agent is analyzing your code…" This lets you know the analysis has started and is in progress.
3. **Review completion** - After analysis completes, AWS Security Agent posts a review to your pull request with the results. All security findings are batched together in a single review to keep your pull request organized and minimize notifications.

## Understanding code review results

AWS Security Agent provides different types of results depending on what it finds during analysis.

### When security issues are found

If AWS Security Agent identifies security issues in your code changes, it posts a review that includes:

- **Summary** - A high-level overview of all security findings at the top of the review, describing the types of issues identified and their potential impact
- **Individual findings** - Detailed security findings appear as threaded comments under the main review, with each finding including:
  - Description of the security issue
  - Location in your code where the issue was found

- Remediation guidance explaining how to address the issue
- Relevant context based on your code review settings (security requirement violations, common vulnerabilities, or both)

> **Note**
>
> The types of security issues analyzed depend on your code review settings. If you configured security requirement validation, findings will reference your organization's custom security requirements. If you configured security vulnerability findings, findings will identify common security vulnerabilities. For more information about code review settings, see the section called "Enable code review capability for a GitHub repository".

### When no security issues are found

If AWS Security Agent completes analysis and finds no security issues in your code changes, it posts a comment: "No issues identified." This confirms the review finished successfully and your code changes did not trigger any security findings based on your configured code review settings.

## Responding to security findings

After reviewing the security findings posted by AWS Security Agent, you can take action directly in GitHub.

- **Address findings** - Update your code based on the remediation guidance provided in the findings, then push new commits to the pull request. AWS Security Agent will analyze the updated code.
- **Resolve conversations** - After addressing a security finding, mark the conversation as resolved in GitHub to track your progress.

> **Tip**
>
> Each finding includes specific remediation guidance tailored to the security issue identified. Review this guidance carefully to understand the security risk and how to address it effectively.

# Filtering code review findings

You can customize how AWS Security Agent analyzes your code by adding a `filtering.md` file to your repository. This file allows you to reduce false positives by providing context about your codebase and excluding files or folders from analysis.

## Creating the filtering file

Create a file named `filtering.md` in the `.awssecurityagent` directory at the root of your repository:

```
.awssecurityagent/filtering.md
```

AWS Security Agent reads this file from the main branch of your repository (for example, `main` or `mainline`) when analyzing pull requests.

## File structure

The `filtering.md` file uses standard Markdown formatting with specific sections that AWS Security Agent recognizes. The file must include a `Code Review` heading followed by one or both of the following sections: `IgnorePatterns` and `ContextHints` (no space).

The following example shows the complete structure of a `filtering.md` file:

```
# filtering.md

## Code Review

### IgnorePatterns

**/*.md

/myapp/src/**/*.snap

/myapp/config/README

### ContextHints

- The backend is a trusted system and won't return non-standard protocols.
- URL is generated from server with presigned token, so no SSRF security
 vulnerabilities.
```

```
- AppSec has verified that we are allowed to use cache with an eviction policy.
```

## Ignore patterns

The `IgnorePatterns` section specifies files and folders that AWS Security Agent should skip during code review. Use `glob patterns` to define which paths to exclude from analysis.

Format requirements:

- Each pattern must be on its own line.
- Separate each pattern with an empty line between them. This ensures the file renders correctly when viewed in GitHub or code review tools.
- Patterns follow the standard glob format. For example, `**/*.md` matches all markdown files, and `/myapp/src/**/*.snap` matches all `.snap` files inside the `/myapp/src/` folder at the root.
- We support upto 1000 ignore patterns in this section.

## Context hints

The `ContextHints` section provides additional context about your codebase that helps AWS Security Agent make more accurate assessments. Use context hints to explain architectural decisions, security exceptions, or other information that might affect how findings are interpreted.

Format requirements:

- Each hint must start with a dash (-) followed by a space.
- Write each hint as a single line of free-form text limited to 500 characters.
- Each hint should describe one specific piece of context about your codebase.
- We support upto 20 context hints in this section.

Context hints are applied after AWS Security Agent completes its initial analysis, helping to filter findings that don't apply to your specific use case.

# Next steps

After reviewing code security findings:

- Update your code based on remediation guidance
- Push new commits to trigger re-analysis of your changes
- Adjust code review settings if needed (see the section called “Enable code review capability for a GitHub repository”)
- Review your organization’s security requirements to understand validation criteria
- Consider penetration testing for comprehensive security validation of deployed applications

# Security in AWS Security Agent

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 Security Agent in the AWS Cloud. This includes the service infrastructure, AI models, and penetration testing agents. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs.
- **Security in the cloud** – Your responsibility includes the following areas:
  - Managing access to AWS Security Agent through IAM policies and permissions
  - Protecting the content you provide to the service, including design documents, code repositories, and application URLs for penetration testing
  - Configuring which repositories and applications are monitored
  - Reviewing and acting on security findings provided by the service
  - Securing your applications based on the remediation guidance provided
  - 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 AWS Security Agent.

## Security Considerations for AWS Security Agent and AI assisted penetration testing

AWS Security Agent is a frontier agent that proactively secures your applications throughout the development lifecycle across all your environments. It conducts automated security reviews customized to your requirements, with security teams centrally defining standards that are automatically validated during reviews. Security Agent performs on-demand penetration testing customized to your application, discovering and reporting verified security risks. This approach scales security expertise across your applications to match development velocity while providing

comprehensive security coverage. By integrating security from design to deployment, it helps prevent vulnerabilities early and at scale.

Security teams define organizational security requirements once in the AWS Console: approved authorization libraries, logging standards, and data access policies. AWS Security Agent automatically enforces these security requirements throughout development, evaluating architectural documents and code against your standards and providing specific guidance when it detects violations. This delivers consistent security enforcement across teams and scales reviews to match development velocity.

For deployment validation, AWS Security Agent transforms penetration testing from a periodic bottleneck into an on-demand capability. Security teams provide target URLs, authentication details, source code and documentation. AWS Security Agent develops deep application understanding and executes sophisticated attack chains to discover and validate vulnerabilities, enabling teams to test whenever needed.

## Key capabilities

AWS Security Agent provides comprehensive security capabilities spanning the entire development lifecycle.

## Design security review

AWS Security Agent provides on-demand security feedback on design documents and assesses compliance with organizational security requirements before code is written. Security teams upload design documents through the web application, where the agent analyzes them against your security requirements and surfaces findings with remediation guidance. This accelerates hours-long manual reviews into focused analysis, enabling teams to address security concerns when remediation is most efficient.

## Code security review

AWS Security Agent analyzes pull requests or uploaded code for organizational security requirements and common security issues like missing input validation and SQL injection risks. The agent provides remediation guidance directly within your code repository platform. Security teams configure which repositories to monitor, scaling evaluation across all codebases while maintaining oversight on critical issues.

# On-demand penetration testing

AWS Security Agent provides on-demand penetration testing that discovers and reports validated security vulnerabilities through tailored multi-step attack scenarios. AWS Security Agent deploys specialized AI agents that develop application context from provided documentation and credentials, then execute sophisticated attack chains to identify complex vulnerabilities that conventional tools miss. It documents findings with impact analysis, reproducible attack paths, and ready-to-implement code fixes, accelerating penetration testing from weeks to hours and scaling validation across your application portfolio.

## FAQs

### Security & Control

#### How does AWS Security Agent authenticate and maintain access to systems?

Penetration testing is the only capability in AWS Security Agent that can authenticate to a user's system at runtime. The AWS Security Agent accepts credentials in the form of static username and password credentials (stored in Secrets Manager), or a credential vendor (as a Lambda Function) as configuration before starting the pen test. These credentials are used to exercise the normal functionality of the user's system/application through the lifecycle of the pen test. We encourage users to create new credentials with appropriately scoped permissions for the purposes of pentesting.

#### Can users control the scope and depth of testing to prevent unintended system impacts?

AWS Security Agent allows customers to select a specific category of vulnerability to explore in an endpoint. Users can specify out-of-scope URLs to prevent AWS Security Agent from performing penetration testing against those targets. https://docs.aws.amazon.com/securityagent/latest/userguide/perform-penetration-test.html

#### Can AWS Security Agent itself pose a security risk?

AWS Security Agent is instructed to discover security risks, but to do so using intentionally minimal impacting payloads (like extracting the SQL version instead of dropping a table when a SQL injection attack is discovered). AWS Security Agent is also confined to deterministic guardrails to prevent risky behavior like creating excessive load against the target application. While guardrails are in place, there could still be unintentional or non-obvious business logic interactions, therefore, we always recommend doing penetration testing against a pre-production environment.

**What data does AWS Security Agent collect and where is it stored?**

AWS Security Agent allows users to upload artifacts to provide context about their application being tested. For more information on data protection, see the section called "Data protection". AWS Security Agent will automatically select the optimal region within your geography to process your inference requests. This maximizes available compute resources, model availability, and delivers the best customer experience. Your data will remain stored only in the region where the request originated, however, input prompts and output results may be processed outside that region. All data will be transmitted encrypted across Amazon's secure network. For more information, see Cross Region Inference.

**What controls are present to block unauthorized testing against an endpoint?**

Endpoints that are specified as target URLs for pentesting will require DNS validation or HTTP validation as a measure of ownership. AWS Security Agent will ask the customer to add a TXT record to the endpoint's DNS or expose an HTTP Route returning validation string as proof of ownership. Only after demonstrating proof of ownership will the user be able to proceed with a pentest. Requests to URLs outside of the target and accessible URLs will be blocked by the network.

Customers are responsible for ensuring they have proper authorization to test all systems that may be affected by their penetration testing activities. All use of AWS Security Agent must comply with the AWS Acceptable Use Policy (https://aws.amazon.com/aup/).

**How do users block and report any abuse using AWS Security Agent?**

AWS Security Agent continuously monitors requests and attempts to access URLs that are outside of the target URLs. If abuse is detected, such as attempting to use AWS Security Agent to conduct unauthorized testing on a third party endpoint, any ongoing pentests in the account will be terminated. Customers can reach out to AWS Support or their AWS account team for help.

**Can AWS Security Agent replace pen testing workflow?**

AWS Security Agent is not a professional penetration testing service, and we encourage users to integrate AWS Security Agent into their security review workflow. AWS Security Agent can provide accessibility to penetration testing on-demand during the development phase of the software lifecycle when engaging with pentesting professionals would be too early, impractical, or need to be re-evaluated too frequently. Security professionals can review findings from AWS Security Agent to validate them, explain them, or extend upon them for new novel findings (if they exist).

### Can users set up role-based access control (RBAC) for different team members?

Yes. AWS Security Agent integrates with AWS IAM Identity Center, allowing admins to manage team members who can access the AWS Security Agent web application which allows users to create, manage and view design reviews and pentests.

## Testing Capabilities

### What types of vulnerabilities can AWS Security Agent detect?

AWS Security Agent detects vulnerabilities in the OWASP Top 10 for web applications. AWS Security Agent provides specific risk types that you can include or exclude in testing outlined below. Findings could arise within these risk categories or from novel findings discovered by following leads from a combination of these risk categories.

- Arbitrary File Upload
  - Arbitrary File Upload confirms that the application should be able to fend off bogus and malicious files in a way to keep the application and the users safe
- Code Injection
  - Code Injection is the general term for attack types which consist of injecting code that is then interpreted/executed by the application
- Command Injection
  - Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application
- Cross-Site Scripting (XSS)
  - Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites
- Insecure Direct Object Reference
  - Insecure Direct Object References (IDOR) occur when an application provides direct access to objects based on user-supplied input
- JSON Web Token Vulnerabilities
  - JWTs are a common source of vulnerabilities, both in how they are implemented in applications, and in the underlying libraries
- Local File Inclusion
  - File Inclusion vulnerability allows an attacker to include a file, usually exploiting a “dynamic file inclusion” mechanisms implemented in the target application

- Path Traversal
  - Path Traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder
- Privilege Escalation
  - Privilege escalation occurs when a user gets access to more resources or functionality than they are normally allowed, and such elevation or changes should have been prevented by the application
- Server-Side Request Forgery (SSRF)
  - Server-Side Request Forgery (SSRF) occurs when the attacker can abuse functionality on the server to read or update internal resources
- Server-Side Template Injection
  - Server Side Template Injection vulnerabilities (SSTI) occur when user input is embedded in a template in an unsafe manner and results in remote code execution on the server
- SQL Injection
  - SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application
- XML External Entity
  - XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser

**What authentication methods does AWS Security Agent support?**

AWS Security Agent supports common authentication methods, including OAuth and JWT. For more information, see the documentation.

**How does AWS Security Agent handle rate limiting and Denial of Service (DOS) prevention?**

AWS Security Agent has guardrails to prevent it from disrupting or taking down endpoints under test, including DOS. It has internal velocity controls to detect and handle unexpected traffic patterns.

### Can AWS Security Agent test both REST and GraphQL APIs?

Yes, AWS Security Agent can test API endpoints. We encourage customers to provide API documentation as **Additional Learning Resources** allowing AWS Security Agent to have better context on the shape and functionality of each API being tested.

### How can users verify that AWS Security Agent has covered all critical application logic and endpoints?

AWS Security Agent will do a breadth-first exploration of the target application(s) and attempt to exercise it normally before attempting any exploits. This allows it to build a working understanding of the application at runtime and discover critical application logic and endpoints. Given its stochastic nature, AWS Security Agent is not guaranteed to discover and test all critical applications and endpoints for any target application. The AWS Security Agent web application provides visibility into all discovered endpoints and actions taken in the Penetration test logs.

## Accuracy & Reliability

### How does AWS Security Agent validate findings before reporting?

AWS Security Agent uses deterministic validators to help validate the reported finding. In the risk types where it is not possible to use deterministic validators, AWS Security Agent will independently replay the finding steps to gain confidence in the validity of the finding. AWS Security Agent only reports the high or medium confidence findings and hides the unverified findings by default.

### Can AWS Security Agent adapt to custom application logic?

AWS Security Agent optionally accepts source code, threat model, design documents, and API documentation as **Additional Learning Resources** to gain user-directed context on the target application used in the lifecycle of a pentest.

### Can users review AWS Security Agent testing methodology before execution?

Currently there is no way to preview AWS Security Agent's course of action. The AWS Security Agent plan is dynamic in nature based on its exploration of the target application. Customers can monitor AWS Security Agent as it goes through its exploration in real time by observing the penetration test logs. If logs show an invalid or undesirable trajectory, customers can stop ongoing pentest run.

## Integration & Deployment

### Does AWS Security Agent integrate with security tools (SIEM, vulnerability management) or CI/CD pipelines?

AWS Security Agent does not integrate with any existing security tools or CI/CD pipelines.

### How does AWS Security Agent handle environment-specific configurations?

AWS Security Agent can be configured to run with specific IAM roles, inside VPCs, with customer-specified application relevant credentials, and with Github source repositories as source code reference to the target application.

### Can AWS Security Agent run in air-gapped or isolated environments?

AWS Security Agent can be configured to have connectivity to VPCs, including ones that do not have outbound internet access.

### Can multiple team members run tests simultaneously?

AWS Security Agent supports 5 concurrent pentest runs per account, independent of who starts the test. Customers can create a maximum of 100 Agent Spaces and 1,000 Pentest projects.

## Operational Impact

### What's the performance impact on tested systems?

AWS Security Agent has guardrails to prevent it from disrupting or taking down endpoints under test. This includes velocity controls on number of calls that AWS Security Agent can make to an endpoint. System or the endpoint under test should expect some increase in traffic and potential monitoring alerts being triggered due to the pen test activity. Our recommendation is to only run AWS Security Agent or any pen testing activity in pre-production environment.

### Can users schedule or throttle AWS Security Agent?

AWS Security Agent does not have public APIs or the ability to schedule the pen test runs. AWS Security Agent also does not offer a concurrency control on requests to the target endpoint when starting the pen test run. If AWS Security Agent is causing problems for target endpoints, customers can stop the ongoing pentest(s).

**What's the typical duration for a complete security assessment?**

The runtime for each pentest depends on the breadth of the target application, and the risk types configured to be assessed. Typical pentest runs can take 12 hours long on configurations that include all risk types.

# AWS managed policies for AWS Security Agents

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

## AWS managed policy: SecurityAgentWebAppAPIPolicy

Grants permissions to interact with the Security Agent Web Application API. This policy enables users to configure and execute automated security penetration tests, manage test executions, view security findings, and access Security Agent resources. This policy references the legacy Agent Instance resource type and specific legacy IAM actions.

**Permissions details**

This policy grants permissions to interact with the Security Testing Control service (securityagent:*) for:

- Pentest Management: Create, update, delete, and list penetration tests and their execution jobs
- Security Findings: View, describe, and update security findings from completed tests, including related content and metadata.

- Task Management: List and retrieve code review and documentation review tasks
- Resource Discovery: List and view agent instances, artifacts, integrations, and discovered endpoints
- Test Execution: Start and stop pentest executions with real-time monitoring capabilities
- Code Remediation: Start automated code remediation for security findings

To view the latest version of the JSON policy document, see SecurityAgentWebAppAPIPolicy in the AWS Managed Policy Reference Guide.

## AWS managed policy: AWSSecurityAgentWebAppPolicy

Grants permissions to interact with the Security Agent Web Application API. This policy enables users to configure and execute automated security penetration tests, manage test executions, view security findings, and access Security Agent resources.

**Permissions details**

This policy grants permissions to interact with the Security Testing Control service (securityagent:*) for:

- Pentest Management: Create, update, delete, and list penetration tests and their jobs
- Security Findings: View, describe, and update security findings from completed tests, including related content and metadata.
- Task Management: List and retrieve code review and design review tasks
- Resource Discovery: List and view agent spaces, artifacts, integrations, and discovered endpoints
- Test Execution: Start and stop pentest jobs with real-time monitoring capabilities
- Code Remediation: Start automated code remediation for security findings

To view the latest version of the JSON policy document, see AWSSecurityAgentWebAppPolicy in the AWS Managed Policy Reference Guide.

## AWS Security Agents updates to AWS managed policies

View details about updates to AWS managed policies for AWS Security Agents since this service began tracking these changes.

To receive notifications of all source file changes to this specific documentation page, you can subscribe to the following URL with an RSS reader:

| Change | Description | Date |
| --- | --- | --- |
| Added permissions to AWSSecurityAgentWebAppPolicy. | Added `TargetDomain` and `DesignReviewFeedback` resource permissions for the new resource types. | March 31, 2026 |
| Added a new managed policy AWSSecurityAgentWebAppPolicy. | Added managed policy `AWSSecurityAgentWebAppPolicy` for the new AgentSpace resource type and IAM action name changes. | February 9, 2026 |
| Added permissions to SecurityAgentWebApAPIPolicy. | Added `securityagent:StartCodeRemediation` to allow users to start automated code remediation for security findings. | January 20, 2026 |
| Added permissions to SecurityAgentWebApAPIPolicy. | Added `securityagent:BatchGetSecurityTestContentMetadata` to allow users to view images in the console. | December 5, 2025 |
| AWS Security Agents started tracking changes. | AWS Security Agents started tracking changes for its AWS managed policies. | December 2, 2025 |

# Data protection in AWS Security Agent

The AWS shared responsibility model applies to data protection in AWS Security Agent. 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 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. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 140-3 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) 140-3.

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 AWS Security Agent 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

AWS Security Agent encrypts all data at rest using AWS-managed encryption keys by default. This includes:

- **Design documents and code** – All design documents, code repositories, and application artifacts you provide for security reviews are encrypted using AES-256 encryption.
- **Security findings** – All security findings, vulnerability reports, and remediation recommendations are encrypted at rest.
- **Configuration data** – Security requirements, custom policies, and service configurations are encrypted.
- **Audit logs** – All service activity logs and audit trails are encrypted.

AWS Security Agent uses AWS Key Management Service (AWS KMS) to manage encryption keys. You can optionally use a customer managed key to encrypt your data, giving you full control over the encryption keys that protect your resources. For more information, see the section called "Customer managed keys".

## Encryption in transit

AWS Security Agent encrypts all data in transit using Transport Layer Security (TLS) 1.2 or higher. This applies to:

- **API communications** – All API calls between your applications and AWS Security Agent use HTTPS with TLS encryption.
- **Console access** – The AWS Security Agent console is accessed over HTTPS.
- **Repository connections** – Connections to GitHub and other code repositories use encrypted protocols.
- **Agent communications** – All communications between the AWS Security Agent service and penetration testing agents use encrypted channels.

## Key management

AWS Security Agent uses AWS Key Management Service (AWS KMS) to manage encryption keys. By default, data is encrypted using AWS-managed keys. You can optionally specify a customer managed key when creating resources such as Agent Spaces and integrations. For more information, see the section called "Customer managed keys".

## Internetwork traffic privacy

AWS Security Agent uses the public internet to communicate with GitHub.

In the default configuration, AWS Security Agent uses the public internet to reach your app for penetration testing. You can optionally configure penetration tests to use a VPC to access your application. For more information, see the section called “Connect agent to private VPC resources”.

## Data deletion

When you delete data from AWS Security Agent:

- The data is marked for deletion and is no longer accessible through the service.
- The data is deleted from all AWS Security Agent systems within 30 days.

To delete your data

1. In the AWS console, navigate to AWS Security Agent.
2. Choose the data you want to delete (security reviews, findings, or custom requirements).
3. Choose **Delete** and confirm the deletion.

# Identity and access management for AWS Security Agent

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 AWS Security Agent resources. IAM is an AWS service that you can use with no additional charge.

## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in AWS Security Agent.

**Service user** – If you use the AWS Security Agent service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Security Agent 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 AWS Security Agent, see the section called “Troubleshooting AWS Security Agent identity and access”.

**Service administrator** - If you're in charge of AWS Security Agent resources at your company, you probably have full access to AWS Security Agent. It's your job to determine which AWS Security Agent 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 AWS Security Agent, see the section called "How AWS Security Agent 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 AWS Security Agent. To view example AWS Security Agent identity-based policies that you can use in IAM, see the section called "AWS Security Agent identity-based policy examples".

## 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, an IAM user, or by assuming an IAM role. AWS suggests using an IAM role, and avoiding using the root user directly.

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 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 Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

### AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform 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 *Account Management Reference 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 (successor to AWS Single Sign-On) 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 (successor to AWS Single Sign-On) User Guide*.
- **Temporary IAM user permissions** – An IAM user 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.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services,

you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.

- **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 Amazon EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the Amazon EC2 instance. To assign an AWS role to an Amazon 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 Amazon 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, 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**.

Every IAM entity (user or role) starts with no permissions. By default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that

has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. 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 such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

### Access control lists (ACLs)

Access control lists (ACLs) are a type of policy that controls which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

### Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

### Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

## How AWS Security Agent works with IAM

Before you use IAM to manage access to AWS Security Agent, learn what IAM features are available to use with AWS Security Agent.

| IAM feature | AWS Security Agent support |
| --- | --- |
| the section called “Identity-based policies for AWS Security Agent” | Yes |
| the section called “Resource-based policies within AWS Security Agent” | No |
| the section called “Policy actions for AWS Security Agent” | Yes |
| the section called “Policy resources for AWS Security Agent” | Partial |
| the section called “Policy condition keys for AWS Security Agent” | Yes |
| the section called “Access control lists (ACLs) in AWS Security Agent” | No |
| the section called “Attribute-based access control (ABAC) with AWS Security Agent” | No |
| the section called “Using temporary credentials with AWS Security Agent” | Yes |
| the section called “Forward access sessions for AWS Security Agent” | Yes |
| the section called “Service roles for AWS Security Agent” | No |
| the section called “Service-linked roles for AWS Security Agent” | Yes |

To get a high-level view of how AWS Security Agent and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

## Identity-based policies for AWS Security Agent

**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 Define custom IAM permissions with customer managed 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 use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

### Identity-based policy examples for AWS Security Agent

To view examples of AWS Security Agent identity-based policies, see the section called "AWS Security Agent identity-based policy examples".

### Resource-based policies within AWS Security Agent

**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 Cross account resource access in IAM in the IAM User Guide.

**Policy actions for AWS Security Agent**

**Supports 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 an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in AWS Security Agent use the following prefix before the action: `securityagent:`. For example, to grant someone permission to create an environment with the AWS Security Agent `CreateEnvironment` API operation, you include the `securityagent:CreateEnvironment` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS Security Agent defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "securityagent:action1",
      "securityagent:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

```
"Action": "securityagent:List*"
```

**Policy resources for AWS Security Agent**

**Supports policy resources:** Partial

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": "*"
```

Some AWS Security Agent API actions support multiple resources. For example, multiple environments can be referenced when calling the `ListEnvironments` API action. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

For example, the AWS Security Agent environment resource has the following ARN:

```
arn:${Partition}:securityagent:${Region}:${Account}:environment/${EnvironmentId}
```

To specify the environments `my-environment-1` and `my-environment-2` in your statement, use the following example ARNs:

```
"Resource": [
        "arn:aws:securityagent:us-east-1:123456789012:environment/my-environment-1",
        "arn:aws:securityagent:us-east-1:123456789012:environment/my-environment-2"
```

To specify all environments that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:securityagent:us-east-1:123456789012:environment/*"
```

**Policy condition keys for AWS Security Agent**

**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 Security Agent defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

## Access control lists (ACLs) in AWS Security Agent

**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 AWS Security Agent

**Supports ABAC (tags in policies):** No

## Using temporary credentials with AWS Security Agent

**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 Switch from a user to an IAM 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.

### Forward access sessions for AWS Security Agent

**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 AWS Security Agent

**Supports service roles:** No

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 Create a role to delegate permissions to an AWS Service in the *IAM User Guide*.

### Service-linked roles for AWS Security Agent

**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.

## AWS Security Agent identity-based policy examples

By default, IAM users and roles don't have permission to create or modify AWS Security Agent resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS

API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies using the JSON editor in the *IAM User Guide*.

## Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Security Agent 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 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 root users in your 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*.

### Using the AWS Security Agent console

To access the AWS Security Agent console, an IAM principal must have a minimum set of permissions. These permissions must allow the principal to list and view details about the AWS Security Agent 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 principals with that policy attached to them.

To ensure that your IAM principals can still use the AWS Security Agent console, create a policy with your own unique name. Attach the policy to the principals. For more information, see Adding permissions to a user in the *IAM User Guide*:

For policy details, see the section called "AWS managed policy: SecurityAgentWebAppAPIPolicy".

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

## Troubleshooting AWS Security Agent identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Security Agent and IAM.

### AccessDeniedException

If you receive an `AccessDeniedException` when calling an AWS API operation, then the IAM principal credentials that you're using don't have the required permissions to make that call.

```
An error occurred (AccessDeniedException) when calling the CreateEnvironment operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
securityagent:CreateEnvironment on resource:
 arn:aws:securityagent:region:111122223333:environment/my-env
```

In the previous example message, the user does not have permissions to call the AWS Security Agent `CreateEnvironment` API operation. To provide AWS Security Agent admin permissions to an IAM principal, see the section called "AWS Security Agent identity-based policy examples".

For more general information about IAM, see Control access to AWS resources using policies in the *IAM User Guide*.

### I want to allow people outside of my AWS account to access my AWS Security Agent 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 AWS Security Agent supports these features, see the section called "How AWS Security Agent 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*.

# Incident response

AWS Security Agent is a proactive security testing service designed to identify and prevent vulnerabilities before they can be exploited. The service focuses on preventative security validation through design reviews, code analysis, and penetration testing rather than reactive incident detection or response.

## Incident detection

AWS Security Agent does not provide incident detection capabilities. The service operates as a proactive security testing tool that validates applications for vulnerabilities during development and before deployment. For runtime security monitoring and incident detection, use services such as Amazon GuardDuty, AWS Security Hub, or Amazon CloudWatch.

## Incident alerting

AWS Security Agent does not generate real-time alerts for security incidents. The service delivers security findings through the AWS Console after completing design reviews, code analyses, or penetration testing engagements. These findings represent potential vulnerabilities discovered during testing rather than active security incidents.

## Incident remediation

AWS Security Agent does not provide automated incident remediation. The service identifies security vulnerabilities and provides remediation guidance, including:

- Detailed descriptions of identified vulnerabilities
- Reproducible exploit paths for validated findings
- Specific code fixes and implementation guidance
- Impact analysis for discovered issues

Development and security teams use this guidance to manually address vulnerabilities before they reach production environments.

## Supporting incident response activities

While AWS Security Agent is not designed for incident response, security teams can use the service to support post-incident activities:

**Vulnerability validation**

After a security incident, use AWS Security Agent to test whether similar vulnerabilities exist in other applications or environments.

**Security posture assessment**

Conduct penetration testing to validate security improvements implemented as part of incident remediation.

**Root cause analysis**

Use code security review capabilities to identify how similar vulnerabilities might exist in other codebases.

# Compliance validation for AWS Security Agent

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 Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- 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.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- 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 AWS Security Agent

> **Note**
>
> AWS Security Agent is available in `us-east-1`, `us-west-2`, `eu-west-1`, `eu-central-1`, `ap-southeast-2`, and `ap-northeast-1`. It uses geographic cross region inference. AWS Security Agent is a tool used during the development of your application, and should not be deployed as critical or customer facing infrastructure.

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with 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 AWS Security Agent

As a managed service, AWS Security Agent 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 Security in the *AWS Well-Architected Framework*.

## Network isolation

AWS Security Agent is a fully managed service accessed through the AWS Console and AWS Security Agent Web Application. Access to the service is controlled through AWS Identity and Access Management (IAM) or AWS IAM Identity Center, which can integrate with your identity provider.

The service does not support VPC endpoints or deployment within customer VPCs, and cannot be restricted to specific subnets through IAM or SCP policies.

AWS Security Agent requires internet access to perform penetration testing on target applications and for control plane operations. The service does not create customer-owned resources with public IP addresses.

### Multi-tenancy and resource isolation

AWS Security Agent is a multi-tenant service. Security reviews, findings, and customer data are isolated to individual AWS accounts and encrypted at rest. AWS applies standard infrastructure isolation controls to ensure that one customer's security testing activities do not impact another customer's performance or confidentiality.

## Configuration and vulnerability analysis in AWS Security Agent

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model.

## Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account. For more information, see Cross-service confused deputy prevention in the AWS Identity and Access Management User Guide.

## Security best practices for AWS Security Agent

AWS Security Agent provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

## Use non-production environments for penetration testing

AWS Security Agent uses a comprehensive suite of penetration testing tools from the Kali Linux distribution. These tools are designed to identify security vulnerabilities and may perform actions that modify application state, data, or system configurations.

**Best practice:** Conduct penetration tests against non-production environments that mirror your production setup. These test environments should:

- Not contain live customer data or sensitive production information
- Be isolated from production systems
- Have similar configurations and security controls as production
- Not use credentials with access to production systems

Testing in production environments may result in:

- Data modification or deletion
- Service disruptions or performance degradation
- Unintended state changes
- Triggering of security alerts or incident response procedures

## Validate AI-generated security findings

AWS Security Agent performs security analysis using AI agents. Due to the non-deterministic nature of AI systems, penetration test runs may produce varying results across different executions.

**Best practice:** Validate security findings before taking remediation action:

- Review the verification scripts generated by AWS Security Agent for each finding
- Execute verification scripts in your test environment to confirm the vulnerability
- Consider running multiple penetration tests to ensure comprehensive coverage
- Apply professional security judgment to assess finding severity and exploitability

Not all identified issues may represent exploitable vulnerabilities in your specific deployment context.

## Review and test generated remediation code

AWS Security Agent can generate code fixes and security improvements for identified vulnerabilities. These AI-generated fixes require verification before deployment.

**Best practice:** Review all generated code changes:

- Examine proposed fixes for completeness and correctness
- Test fixes thoroughly in non-production environments
- Verify that fixes don't introduce new vulnerabilities or break functionality
- Use AWS Security Agent to retest after applying fixes, or run the provided verification scripts
- Follow your organization's code review and approval processes

## Code repository access

AWS Security Agent can provide security guidance on code changes through pull request comments and code review integration. To protect sensitive security information, this functionality operates under specific constraints.

**Limitation:** Code security guidance is restricted to private repositories only. This ensures that:

- Security findings remain confidential to your organization
- Potential vulnerabilities are not publicly disclosed before remediation
- Generated fix recommendations don't expose exploitation details

AWS Security Agent does not provide code security guidance for public repositories. It will not comment on public repositories or open-source projects where security findings would be publicly visible.

AWS Security Agent penetration tests can review and remediate private and public repositories that you configured for the pentest. If the repository is public, the remediation code will be provided as a downloadable diff file instead of a pull request.

## Accessible URLs

Accessible URLs specify additional endpoints that the penetration testing environment can access during testing. These are necessary when your application depends on external services such as third-party authentication providers or CDNs. All network dependencies required for testing must

be specified as either target URLs or accessible URLs. The network blocks access to any unspecified endpoints.

**Security implications:** AWS Security Agent is not instructed to perform security testing on accessible URLs. By specifying accessible URLs, you indicate trust in these dependencies. Penetration test data, including credentials, may be transmitted to these accessible URL endpoints during testing.

## Cross Region Inference

AWS Security Agent will automatically select the optimal region within your geography to process your inference requests. This maximizes available compute resources, model availability, and delivers the best customer experience. Your data will remain stored only in the region where the request originated, however, input prompts and output results may be processed outside that region. All data will be transmitted encrypted across Amazon's secure network.

AWS Security Agent will securely route your inference requests to available compute resources within the geographic area where the request originated, as follows:

- Inference requests originating in the European Union will be processed within the European Union.
- Inference requests originating in the United States will be processed within the United States.
- Inference requests originating in Australia will be processed within Australia.
- Inference requests originating within Japan will be processed within Japan.

Cross-Region inference is always enabled and cannot be opted out of.

- **Data residency** – All data processing occurs within your geographic area. If your organization requires data processing within a specific single region, evaluate whether processing across the broader geographic boundary satisfies your compliance obligations.
- **IAM and Service Control Policies** – Ensure your IAM policies and Service Control Policies (SCPs) allow access to regions within your geography used for cross-Region inference operations.

# IAM actions and resources migration

AWS Security Agent is a frontier agent that proactively secures your applications throughout the development lifecycle across all your environments. If you onboarded to AWS Security Agent prior

to February 9, 2026, you will be impacted by upcoming changes on March 9, 2026 to your existing Agent Instance resources and AWS Security Agent IAM actions. In preparation for releasing public API/SDK support, the Agent Instance resource is being renamed to Agent Space, and specific IAM actions are being renamed. These changes will affect any Application or Agent Instances IAM roles you have created prior to March 9, 2026. In order to avoid seeing authentication issues after March 9, 2026, you will need to follow the steps under Preparing for Migration.

> **Note**
>
> If you create any new Agent Instances after February 9, 2026, the new Agent Instance will be created as an Agent Space and no migration steps will be required.

## Planned Changes

AWS Security Agent is renaming the Agent Instance resource to Agent Space: `arn:aws:securityagent:us-east-1:{{accountId}}:agent-instance/*` renamed to `arn:aws:securityagent:us-east-1:{{accountId}}:agent-space/*`. Additionally, the following IAM actions are being renamed:

- `securityagent:ListAgentInstances` renamed to `securityagent:ListAgentSpaces`
- `securityagent:ListControls` renamed to `securityagent:ListSecurityRequirements`
- `securityagent:BatchGetAgentInstances` renamed to `securityagent:BatchGetAgentSpaces`
- `securityagent:BatchGetSecurityTestContentMetadata` renamed to `securityagent:BatchGetPentestJobContentMetadata`
- `securityagent:BatchGetTasks` renamed to `securityagent:BatchGetPentestJobTasks`
- `securityagent:CreateDocumentReview` renamed to `securityagent:CreateDesignReview`
- `securityagent:GetDocumentReview` renamed to `securityagent:GetDesignReview`
- `securityagent:GetDocumentReviewArtifact` renamed to `securityagent:GetDesignReviewArtifact`
- `securityagent:ListDocumentReviews` renamed to `securityagent:ListDesignReviews`

- `securityagent:ListDocumentReviewComments` renamed to `securityagent:ListDesignReviewComments`
- `securityagent:ListTasks` renamed to `securityagent:ListPentestJobTasks`
- `securityagent:StartPentestExecution` renamed to `securityagent:StartPentestJob`
- `securityagent:StopPentestExecution` renamed to `securityagent:StopPentestJob`
- `securityagent:DeleteDocumentReview` renamed to `securityagent:DeleteDesignReview`

## Preparing for Migration

In order to avoid seeing issues after March 9, 2026 while continuing to use AWS Security Agent prior to March 9, 2026, you will need to **trust both the new and old resources and IAM actions in your IAM roles/policies** until March 9, 2026. The below instructions will provide a guide for migrating to the new resource and action formats:

1. Log in to your AWS account and navigate to the **AWS Security Agent console**
2. In the left hand panel, select **Settings** and click the role under **Service role**
3. In the IAM console for the associated role, select **Add permissions** and **Attach policies**
4. Select **AWSSecurityAgentWebAppPolicy** and click **Add permissions**
   a. **Important Note:** Verify that you have selected **AWSSecurityAgentWebAppPolicy** as the new policy and not **SecurityAgentWebAppAPIPolicy**
5. Verify that your IAM role now has both **AWSSecurityAgentWebAppPolicy** and **SecurityAgentWebAppAPIPolicy** under **Permissions policies**
6. In the same IAM role console, select **Trust relationships** then **Edit trust policy**
7. Update your trust policy to the following format, replacing **{{accountId}}** with your AWS account ID

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{{accountId}}"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:securityagent:us-east-1:{{accountId}}:application/*",
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-space/*",
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-instance/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "sts:SetContext",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{{accountId}}"
        },
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": "arn:aws:iam::aws:contextProvider/
IdentityCenter"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:securityagent:us-east-1:{{accountId}}:application/*",
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-space/*",
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-instance/*"
          ]
        }
      }
    }
  ]
}
```

8. Navigate back to the **AWS Security Agent console**. From the left-hand panel, select **Agent Spaces**

9. For each Agent Space you have with penetration testing enabled, perform the following steps
   a. Navigate to the Agent Space and select **Penetration test**
   b. Scroll down to **Service access** and click the role under **Service role name**
   c. In the IAM console for the associated role, select **Trust relationships** then **Edit trust policy**
   d. Update your trust policy to the following format, replacing **{{accountId}}** with your AWS account ID

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{{accountId}}"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-space/*",
            "arn:aws:securityagent:us-east-1:{{accountId}}:agent-instance/*"
          ]
        }
      }
    }
  ]
}
```

# Logging AWS Security Agent API calls with AWS CloudTrail

AWS Security Agent is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Security Agent. CloudTrail captures all API calls for AWS Security Agent as events. The calls captured include calls from the AWS Security Agent console and code calls to the AWS Security Agent API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for

AWS Security Agent. 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 AWS Security Agent, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

## AWS Security Agent information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Security Agent, 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 AWS Security Agent, 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 AWS 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 the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All AWS Security Agent actions are logged by CloudTrail. For example, calls to the `CreatePentest`, `BatchGetPentestJobs`, and `ListFindings` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

# Example: AWS Security Agent log file entries

## Understanding AWS Security Agent 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 `CreatePentest` action:


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2025-01-15T10:30:00Z",
    "eventSource": "securityagent.amazonaws.com",
    "eventName": "CreatePentest",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "aws-cli/2.13.0",
    "requestParameters": {
        "pentestName": "WebApp-Security-Test",
        "targetUrl": "https://example.com",
        "testScope": "OWASP-Top-10"
    },
    "responseElements": {
        "pentestId": "pt-1234567890abcdef0",
        "pentestArn": "arn:aws:securityagent:us-east-1:123456789012:pentest/
pt-1234567890abcdef0"
    },
    "requestID": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
```

```
    "eventID": "12345678-1234-1234-1234-123456789012",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

# Service Quotas

AWS Security Agent has quotas that limit the number of resources you can create and the rate at which you can perform operations. Quotas marked as adjustable can be increased by submitting a request through AWS Support. For more information, see Creating support cases and case management.

## Operations Quotas

Operations quotas limit the monthly usage of security testing and review features to help manage service capacity.

| Resource | Scope | Quota | Adjustable |
|---|---|---|---|
| Pentest hours | Per month per account per region | 80 | Yes |
| Design reviews | Per month per account per region | 200 | Yes |
| Code reviews | Per month per account per region | 1,000 | Yes |

## Configuration Quotas

Configuration quotas limit the number of resources and settings you can configure in your AWS Security Agent environment.

| Resource | Scope | Quota | Adjustable |
|---|---|---|---|
| Agent Spaces | Per account per region | 100 | Yes |
| Integrations | Per account per region | 20 | No |
| Integrated resources per integration | Per integration | 20 | No |

| Resource | Scope | Quota | Adjustable |
|---|---|---|---|
| Custom security requirements | Per account per region | 20 | No |
| Pentest projects | Per account per region | 1,000 | Yes |
| Concurrent pentest runs | Per account per region | 5 | Yes |

# Document history

The following table describes some of the major updates and new features for the AWS Security Agents User Guide.

| Change | Description | Date |
| --- | --- | --- |
| Customer managed key support | Added documentation for customer managed key (CMK) support. You can now specify a customer managed KMS key when creating Agent Spaces and integrations to encrypt your data with keys you control. | March 31, 2026 |
| AWS managed policy updates | Added AWSSecurityAgentWebAppPolicy managed policy for the new TargetDomain and DesignReviewFeedback resource types. | March 31, 2026 |
| AWS managed policy updates | Added AWSSecurityAgentWebAppPolicy managed policy for the new AgentSpace resource type and IAM action name changes. | February 9, 2026 |
| AWS managed policy updates | Updated SecurityAgentWebAppAPIPolicy to allow customers to delete design reviews. | January 28, 2026 |
| AWS managed policy updates | Updated SecurityAgentWebAppAPIPolicy to allow customers to start | January 20, 2026 |

| | | |
|---|---|---|
| | automated code remediation for security findings. | |
| AWS managed policy updates | Updated to SecurityAgentWebAppAPIPolicy to allow customers to view images in the console. | December 5, 2025 |
| AWS Security Agents initial release | Initial documentation for service launch | December 2, 2025 |