ユーザーガイド

AWS セキュリティエージェント

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS セキュリティエージェント: ユーザーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

序章 1
AWS セキュリティエージェントとは 1
主な機能 1
利点 3
AWS セキュリティエージェントの機能 3
AWS セキュリティエージェントの仕組み 4
概要 5
リソース階層とライフサイクルを理解する 8
組織全体で共有される内容 8
エージェントスペースあたりのスコープ 9
GitHub リポジトリが階層にどのように収まるか 11
セキュリティ機能の主な違い 11
リソース関係について 13
はじめに 15
にサインアップする AWS 15
進めたい作業内容を選択します 15
クイックスタート: 侵入テストを実行する 16
前提条件 17
ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する 17
ステップ 2: ペネトレーションテストを有効にしてドメインを検証する 18
ステップ 3: ソースコードを接続する (オプション) 19
ステップ 4: 侵入テストを作成して実行する 19
ステップ 5: 侵入テストの結果を確認する 20
クイックスタート: コードレビューを実行する 21
前提条件 17
ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する 17
ステップ 2: コードレビューを有効にして設定する 22
ステップ 3: コードレビューを作成して実行する 24
ステップ 4: コードレビューの結果を確認する 24
クイックスタート: 脅威モデルを実行する 25
ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する 17
ステップ 2: ソースコードを接続する 26
ステップ 3: 脅威モデルを作成して実行する 27
ステップ 4: 脅威を確認する 28

管理者向け (コンソール) 29
設定例 29
エージェントスペースの設定と作成 29
AWS セキュリティエージェントをセットアップする 29
エージェントスペースを作成する 35
Connect 統合 37
エージェントスペースとの統合の仕組み 37
AWS セキュリティエージェントを GitHub リポジトリに接続する 39
GitHub 統合を削除する 45
AWS セキュリティエージェントを GitLab リポジトリに接続する 47
AWS セキュリティエージェントを GitLab セルフマネージドに接続する 51
GitLab 統合を削除する 54
AWS セキュリティエージェントを GitHub Enterprise Server に接続する 55
GitHub Enterprise Server 統合を削除する 59
Atlassian インストール ID を検索する 60
AWS セキュリティエージェントを Bitbucket リポジトリに接続する 60
Bitbucket 統合を削除する 64
AWS セキュリティエージェントを Confluence に接続する 65
Confluence 統合を削除する 69
プライベートにホストされたソースコントロールに接続する 70
エージェントをプライベート VPC リソースに接続する 75
機能を設定する 78
ペネトレーションテストを有効にする 79
GitHub リポジトリのプルリクエストコードレビューを有効にする 85
コードレビューを有効にする 85
脅威モデリングを有効にする 92
ユーザーが侵入テストとコードレビューの検出結果の修正を開始できるようにする 96
S3 バケットからエージェントリソースを提供する 97
ペネトレーションテスト用のアプリケーションドメインを有効にする 98
ペネトレーションテストに使用されるマネージドターゲットドメイン 100
セキュリティ要件を管理する 102
セキュリティ要件を管理する 102
AWS マネージドパック 109
ドキュメントからセキュリティ要件を生成する 110
要件生成用のドキュメントを準備する 112
ユーザーとアクセスを管理する 113

ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する 113
AWS セキュリティエージェントの IAM ロールを作成する 116
カスタマーマネージドキー 122
トラブルシューティング 146
アクセスが拒否されました: 選択した GitHub アカウントタイプが正しくないか、指定された組織名が正しくありません 147
アクセス拒否: GitHub アプリを組織にインストールするためのアクセス許可が不十分 147
エージェントがペネトレーションテスト中にエンドポイントに接続できない 148
追加のヘルプの取得 148
ユーザーと開発者向け (ウェブアプリと IDE) 149
AWS セキュリティエージェントウェブアプリにログインする 149
アクセス方法 149
IAM Identity Center (SSO) でログインする 150
管理者アクセス (IAM) を使用してログインする 151
設計レビューを作成する 152
前提条件 17
ステップ 1: 設計レビューの作成を開始する 152
ステップ 2: 設計レビューに名前を付ける 153
ステップ 3: 設計ファイルをアップロードする 153
ステップ 4: 設計レビューを開始する 154
次の手順 34
設計レビューの結果を確認する 154
脅威モデルを作成する 158
前提条件 17
AWS Security Agent が脅威モデルを実行する方法 160
脅威モデルページにアクセスする 160
脅威モデルを作成する 160
脅威モデルを実行する 163
脅威モデルの実行をモニタリングする 163
脅威モデルを編集する 165
脅威モデルを削除する 165
次の手順 34
脅威モデルからの脅威を確認する 166
プルリクエストでコードセキュリティの検出結果を確認する 171
プルリクエストでのコードレビューの仕組み 172
コードレビュー結果について 172

セキュリティ検出結果への対応 173
コードレビューの結果のフィルタリング 174
次の手順 34
コードレビューを作成する 176
前提条件 17
コードレビューページにアクセスする 177
コードレビューを作成する 177
コードレビューを実行する 183
コードレビューの実行をモニタリングする 183
次の手順 34
コードレビューの結果を確認する 185
コードレビューの検出結果の修正 191
S3 で差分コードスキャンを実行する 194
差分スキャンの仕組み 194
前提条件 17
ステップ 1: 差分ファイルを生成する 195
ステップ 2: 差分を S3 にアップロードする 196
ステップ 3: 差分コードレビュージョブを開始する 196
ステップ 4: スキャンをモニタリングする 197
ステップ 5: 検出結果を確認する 198
クォータと制限 198
次の手順 34
IDE からコードセキュリティスキャンを実行する 199
IDE 統合の仕組み 199
前提条件 17
MCP サーバーをインストールする 200
初回の設定 202
フルセキュリティスキャンを実行する 203
差分スキャンを実行する 203
脅威モデルレビューを実行する 204
検出結果を確認する 204
自動修正を適用する 205
自動スキャン提案 (Kiro) 205
使用可能なスキャンタイプ 206
環境変数 206
トラブルシューティング 207

クォータと制限 198
次の手順 34
侵入テストを作成する 208
前提条件 17
侵入テストの作成を開始する 208
侵入テストに名前を付ける 209
ペネトレーションテストスコープを設定する 209
IAM ロールを設定する 212
自動コード修復 192
VPC リソースを設定する (オプション) 213
認証情報を設定する (オプション) 215
追加のリソースをアタッチする (オプション) 217
侵入テストを作成する 220
ペネトレーションテストの結果を確認する 221
ペネトレーションテスト用の認証情報を提供する 233
ペネトレーションテストの検出結果の修正 238
セキュリティとリファレンス 240
セキュリティ 240
セキュリティに関する考慮事項 240
AWS マネージドポリシー 248
データ保護 251
ID とアクセス管理 254
インシデントへの対応 270
コンプライアンス検証 272
耐障害性 273
インフラストラクチャセキュリティ 274
インターフェイス VPC エンドポイント 275
設定と脆弱性の分析 280
サービス間の混乱した代理の防止 280
セキュリティのベストプラクティス 280
IAM アクションとリソースの移行 285
CloudTrail による口グ記録 289
サービスクォータ 291
オペレーションクォータ 291
設定クォータ 292
ドキュメント履歴 293

..... ccxcvi

序章

AWS セキュリティエージェントの機能、仕組み、リソースの連携について説明します。

AWS セキュリティエージェントとは

AWS セキュリティエージェントは、開発ライフサイクルを通じてアプリケーションをプロアクティブに保護するフロンティアエージェントです。組織の要件に合わせた自動セキュリティレビューを実施し、脅威モデルを構築してアプリケーションの攻撃方法を特定し、コンテキストに応じた侵入テストをオンデマンドで提供します。AWS セキュリティエージェントは、設計からデプロイまでセキュリティを継続的に検証することで、すべての環境で脆弱性を早期に防止できます。

セキュリティチームは、承認された認可ライブラリ、ログ記録標準、データアクセスポリシーなど、組織のセキュリティ要件を AWS コンソールで一度定義します。AWS セキュリティエージェントは、開発中にこれらのセキュリティ要件を自動的に適用し、アーキテクチャドキュメントとコードを標準に照らして評価し、違反を検出したときに特定のガイダンスを提供します。これにより、すべてのチームで設計レビューとコードレビューに一貫したセキュリティの適用が提供されます。

デプロイの検証のために、AWS セキュリティエージェントは侵入テストを定期的にボトルネックからオンデマンド機能に変換します。セキュリティチームは、ターゲット URLs、認証の詳細、ソースコード、アプリケーションドキュメントを提供します。AWS セキュリティエージェントは、アプリケーションの深い理解を深め、高度な攻撃チェーンを実行して脆弱性を検出して検証し、チームが必要に応じてテストできるようにします。

主な機能

AWS セキュリティエージェントは、開発ライフサイクル全体にわたる包括的なセキュリティ機能を提供します。

セキュリティレビューの設計

AWS セキュリティエージェントは、設計ドキュメントに関するリアルタイムのセキュリティフィードバックを提供し、コードが記述される前に組織のセキュリティ要件への準拠を評価することで、セキュリティを移行します。セキュリティチームはウェブアプリケーションを介してドキュメントをアップロードし、結果に優先順位を付けるための修復ガイダンスを受け取り、時間のかかる手動レビューを重点分析に迅速化します。セキュリティ標準をすべての設計レビューにプロアクティブに組み込むことで、後期段階のアーキテクチャの再作業を減らし、複数の開発チームに対応できます。

脅威モデリング

AWS セキュリティエージェントは、アプリケーションの脅威モデルを構築して、攻撃方法を特定します。技術的な設計ドキュメント (スコープドキュメント) を提供して、エージェントが何に焦点を当てているか、エージェントが既存のシステムを理解するためのコンテキストとしてソースコード、またはその両方を定義します。AWS セキュリティエージェントは、アプリケーションのアーキテクチャ、コンポーネント、信頼境界、データフロー、セキュリティ体制を説明するシステム概要と、STRIDE カテゴリ (スプーフィング、改ざん、否認、情報開示、サービス拒否、特権昇格) で分類された一連の脅威を、重要度レベルと実用的な推奨事項とともに生成します。脅威モデルは、コードと設計の進化に応じて再実行できる再利用可能な設定であり、開発ライフサイクル全体で反復的なセキュリティ評価を可能にします。

コードセキュリティレビュー

AWS セキュリティエージェントは、2 つの補完的なアプローチを通じてアプリケーションをプロアクティブに保護します。まず、ウェブアプリケーションでコードレビューを作成して、GitHub、GitLab、Bitbucket、GitHub Enterprise Server リポジトリまたは S3 バケットから完全なソースコードを包括的にスキャンし、セキュリティの脆弱性を特定し、コードベース全体の組織要件への準拠を検証できます。次に、接続されたリポジトリの自動プルリクエスト分析を有効にできます。AWS セキュリティエージェントはコードの変更を確認し、プルリクエストまたはマージリクエストのコメントとしてセキュリティ検出結果を直接投稿します。いずれの場合も、開発者は修復ガイダンスを受け取り、AWS セキュリティエージェントは、特定された脆弱性のコード修正を含むプルリクエストを自動的に生成したり、リクエストをマージしたりできます。これにより、すべてのリポジトリにセキュリティの専門知識が組み込まれ、開発パイプラインのセキュリティ関連の遅延が軽減され、すべてのコードベースで評価がスケーリングされます。

ペネトレーションテスト

AWS セキュリティエージェントは、カスタマイズされた複数ステップの攻撃シナリオを通じて、セキュリティの脆弱性を検出、検証、レポート、修正する専門の AI エージェントをデプロイすることで、オンデマンドの侵入テストを提供します。エージェントは、ソースコードとドキュメントを分析して、自動セキュリティスキャンツールが検出できない脆弱性を特定して悪用することで、アプリケーションのコンテキストを理解します。影響分析、再現可能な攻撃パスを使用して検出結果を文書化し、ready-to-implementコード修正を含むプルリクエストを作成します。これにより、定期的な評価が、重要な評価のみに制限されるのではなく、すべてのアプリケーションにまたがる継続的な検証に変換されます。

利点

オンデマンドアクセシビリティ

AWS セキュリティエージェントは、セキュリティの専門知識にすぐにアクセスできます。開発チームは、必要に応じて設計レビュー、脅威モデル、コード分析、侵入テストをオンデマンドで実行し、最新の開発サイクルのペースに合わせて、あらゆる段階でプロアクティブセキュリティを実現できます。

組織の要件を自動的に検証する

AWS コンソールで組織のセキュリティ要件を一度定義します。AWS セキュリティエージェントは、すべての設計とコードのセキュリティレビュー中にすべてのアプリケーションでこれらの要件を自動的に検証し、チームが一般的なチェックリストではなく特定の要件に対処できるようにします。

セキュリティの専門知識をスケールする

AWS セキュリティエージェントは、組織のセキュリティ要件の適用を自動化することで、開発速度に合わせてセキュリティレビューをスケーリングします。要件を一元的に設定し、検出結果分析で包括的なレビューを行い、AWS コンソールを通じて組織全体のペネトレーションテストの範囲を管理します。

確認された脆弱性に対する実用的な修正

AWS セキュリティエージェントは、プルーフベースのエクスプロイトを通じてペネトレーションテスト中に見つかったセキュリティ検出結果を検証し、再現可能なエクスプロイトパス、包括的な影響分析を提供し、すぐにready-to-implement修正を含むプルリクエストを開発者フレンドリーな言語で作成します。これにより、チームは誤検出に時間を費やすことなく、正当な影響の大きいセキュリティリスクに集中できます。

マルチクラウドとハイブリッドクラウドのサポート

AWS Security Agent は、AWS、オンプレミス、ハイブリッド、マルチクラウド、SaaS 環境間で動作し、インフラストラクチャやプラットフォームの選択に関係なく、一貫したセキュリティガイダンスとテストを実現します。

AWS セキュリティエージェントの機能

AWS セキュリティエージェントは、開発ライフサイクル全体で 4 つのコアセキュリティ機能を提供します。

- 設計セキュリティレビュー — 設計とアーキテクチャのドキュメントを確認して、セキュリティリスクを特定します。ウェブアプリケーションを介してドキュメントをアップロードすると、サービスは組織のセキュリティ要件に照らしてドキュメントを分析します。AWS セキュリティエージェントは、承認された認可ライブラリ、ログ記録標準、データアクセスポリシーなど、定義されたセキュリティ要件への準拠を評価します。これにより、安全でない設計やポリシー違反を開発プロセスの早い段階で発見できます。
- 脅威モデリング — アプリケーションの脅威モデルを構築し、攻撃方法を特定します。設計ドキュメントをスコープドキュメントとして提供して分析の焦点を定義するか、ソースコードをコンテキストとして提供して、エージェントが既存のシステムを理解できるようにします。AWS セキュリティエージェントは、アプリケーションのアーキテクチャ、コンポーネント、信頼境界、データフロー、セキュリティ体制を説明するシステム概要と、STRIDE カテゴリ (スプーフィング、改ざん、否認、情報開示、サービス拒否、特権昇格) で分類された一連の脅威を、重要度レベルと実用的な推奨事項とともに示します。各脅威は、ソースコードの証拠にリンクされます。
- コードセキュリティレビュー — リポジトリと S3 ソース内のコード を分析して、言語、フレームワーク、アーキテクチャ全体の組織のセキュリティ要件に対するセキュリティの脆弱性と違反を特定します。ウェブアプリケーションでコードレビューを作成して完全なソースコードの包括的なスキャンを実行するか、プルリクエストコメントを有効にして GitHub でコード変更を自動的にレビューします。AWS セキュリティエージェントは、特定の修復ガイダンスを提供し、特定された脆弱性のコード修正を含むプルリクエストを自動的に生成できます。これにより、すべての開発チームでセキュリティポリシーが一貫して適用されます。
- オンデマンド侵入テスト — カスタマイズされた複数ステップの攻撃シナリオを通じて、ライブウェブアプリケーションと APIs のセキュリティ脆弱性 を検出、検証、レポート、修正します。テストの範囲、認証の詳細、リソースを指定して、ウェブアプリケーションを介してペンテストを作成するようにサービスを設定します。AWS セキュリティエージェントは、提供されたソースコードとドキュメントからアプリケーションコンテキストを開発し、高度な攻撃チェーンを実行して、静的分析と従来のツールが見逃す悪用可能な脆弱性を特定します。また、ready-to-implementコード修正を提供し、コードリポジトリに直接プルリクエストを作成するため、脆弱性をより迅速に解決できます。

AWS セキュリティエージェントの仕組み

AWS セキュリティエージェントは 3 つのインターフェイスで動作し、開発ライフサイクルを通じてプロアクティブなアプリケーションセキュリティを提供します。セキュリティ設定は AWS マネジメントコンソールで管理され、セキュリティレビューはセキュリティエージェントウェブアプリケーションで実施され、コードとセキュリティ検出結果の自動修復は GitHub などのコードリポジトリプラットフォームで直接行われます。

概要

AWS セキュリティエージェントは、次の 3 つの主要コンポーネントで構成されています。

- AWS マネジメントコンソール - エージェントスペースの設定、セキュリティ要件の定義、ペネトレーションテストの設定、コードリポジトリの接続、ユーザーアクセスの管理
- セキュリティエージェントウェブアプリケーション - 設計レビューの実行、脅威モデルの作成と実行、ペネトレーションテストの実行、コードレビューの作成と実行、割り当てられたエージェントスペース内のセキュリティ検出結果の確認
- コードリポジトリの統合 - プルリクエストとペネトレーションテスト修復プルリクエストに関する自動コードレビューを受け取ります。現在、AWS セキュリティエージェントは GitHub への接続をサポートしています。

AWS セキュリティエージェントは、使用するインターフェイスで識別できる 3 つのロールのいずれかで使用できます。

ロール	作業場所と作業内容
管理者	AWS マネジメントコンソールで動作 — エージェントスペースの設定、セキュリティ要件の定義、機能の設定、ユーザーアクセスの管理を行います。
ユーザー	セキュリティエージェントウェブアプリケーションで動作 — 設計レビュー、脅威モデル、侵入テスト、コードレビューを実行し、結果を確認します。
デベロッパー	GitHub または IDE (Kiro や Claude Code など) で動作 — プルリクエストで自動セキュリティ検出結果を受け取り、開発環境を離れることなくコードスキャンを実行します。

全体でこれらのロール名を使用して、各タスクを実行するユーザーを示します。1 人のユーザーが複数のロールを保持できます。

コンソール設定

管理者は、AWS マネジメントコンソールを使用して AWS セキュリティエージェントを設定します。

エージェントスペース - AWS マネジメントコンソールで最初のエージェントスペースを作成すると、AWS セキュリティエージェントはアカウントのウェブアプリケーションを作成します。作成す

る各エージェントスペースは、保護する個別のアプリケーションまたはプロジェクトを表します。ウェブアプリケーションで、ユーザーはセキュリティ評価を実行するときに作業するエージェントスペースを選択します。

セキュリティ要件 - AWS Security Agent が設計とコードのレビュー中に評価するセキュリティ要件を、セキュリティ要件パックに整理して定義します。業界標準に基づいて AWS マネージドパックを有効にしたり、組織のポリシー用にカスタムパックを作成したり、セキュリティドキュメントをアップロードして要件を生成したりできます。要件はすべてのエージェントスペースに適用されます。

ペネトレーションテスト設定 - 次の方法で各エージェントスペースのペネトレーションテスト機能を設定します。

- DNS または HTTP 検証によるテストのターゲットドメインの検証
- プライベートアプリケーションをテストするための VPC アクセスの設定
- ペネトレーションテスト実行用の CloudWatch ログ記録の設定
- テスト認証情報用の AWS Secrets Manager または Lambda 関数の設定
- 追加のアプリケーションコンテキストの S3 バケットの指定

コードレビュー設定 - 次の方法で各エージェントスペースのコードレビュー機能を設定します。

- ソースコードを含む GitHub リポジトリまたは S3 バケットの接続
- コードレビュー設定 (セキュリティの脆弱性、カスタム要件、またはその両方) の選択
- GitHub でのコード変更の自動レビューのためのプルリクエストコメントの有効化
- コードレビュー実行用の CloudWatch ログ記録の設定

脅威モデリング設定 - 次の方法で各エージェントスペースの脅威モデリング機能を設定します。

- ソースコードを含むソースコードリポジトリまたは S3 バケットの接続
- スコープドキュメント (特徴量設計ドキュメント) を追加して、分析を特定の特徴量に絞る
- 脅威モデル実行の CloudWatch ログ記録の設定
- AWS リソースアクセスのサービスロールの設定

統合 - GitHub リポジトリを各エージェントスペースに接続して、主要な機能を有効にします。

- より正確なペネトレーションテストのためのアプリケーションコンテキストを提供する

- 完全なリポジトリスキャンとプルリクエスト分析のコードレビューを有効にする
- コードレビューとペネトレーションテストの検出結果のプルリクエストによる自動コード修復を有効にする

ユーザーアクセス - ユーザーがセキュリティエージェントウェブアプリケーションにアクセスする方法を管理します。IAM Identity Center (SSO) を有効にしている場合は、AWS セキュリティエージェントコンソールでユーザーを割り当てて、ウェブアプリケーションへの直接 SSO アクセスを提供します。IAM 専用アクセスを使用している場合、AWS コンソールにアクセスできるユーザーは、任意のエージェントスペースのコンソールの管理者アクセスリンクからウェブアプリケーションを起動できます。

ウェブアプリケーションアクティビティ

ユーザーはセキュリティエージェントウェブアプリケーションにアクセスして、割り当てられたエージェントスペース内でセキュリティ評価を実行します。

エージェントスペースの選択 - ウェブアプリケーションにログインすると、ユーザーは作業するエージェントスペースを選択します。ユーザーは、割り当てられたエージェントスペースのみを表示してアクセスできます。

設計レビュー - 組織のセキュリティ要件に照らして分析するための設計ドキュメントとアーキテクチャ仕様をアップロードします。修復ガイダンスを使用して結果を確認します。

脅威モデル - ソースコード、技術設計ドキュメント (スコープドキュメント) 、またはその両方を指定して、脅威モデルを作成して実行します。スコープドキュメントは、エージェントが分析に焦点を当てる内容を定義します。ソースコードは、既存のシステムに関するコンテキストを提供します。各実行では、アプリケーションのアーキテクチャ、信頼境界、データフロー、セキュリティ体制を説明するシステム概要と、重要度レベルと実用的な推奨事項を含む STRIDE カテゴリ別に分類された一連の脅威が表示されます。

コードレビュー - 完全なソースコード全体で包括的な静的分析を実行するコードレビューを作成して実行します。GitHub リポジトリまたは S3 ソースを選択し、スキャン設定を設定し、修復ガイダンスを使用して詳細なセキュリティ検出結果を確認します。AWS セキュリティエージェントは、セキュリティの脆弱性を特定し、コードベース全体で組織のカスタムセキュリティ要件への準拠を検証します。

侵入テスト - ターゲット URLs、およびドキュメントを指定して、侵入テストを設定して実行します。AWS セキュリティエージェントは、自動テストを実行して、複数ステップの攻撃シナリオを通じて悪用可能な脆弱性を検出します。

検出結果の確認 - 影響分析、再現可能な攻撃パス、修復ガイダンスなど、設計レビュー、脅威モデル、コードレビュー、侵入テストからの詳細なセキュリティ検出結果を検証します。

修正の検証 - 修復を実装した後にセキュリティ評価を再実行し、脆弱性が対処されたことを確認します。

GitHub 統合

AWS セキュリティエージェントは GitHub と直接統合され、デベロッパーのワークフローで自動化されたセキュリティフィードバックを提供します。

プルリクエストコメント - 管理者が AWS セキュリティエージェント GitHub アプリをインストールし、エージェントスペースのコードレビューコメントを使用してコードレビューを有効にすると、AWS セキュリティエージェントは接続されたリポジトリ内のプルリクエストを自動的に分析します。開発者は、プルリクエストコメントでセキュリティ検出結果と修復ガイダンスを直接受け取り、組織のセキュリティ要件と一般的な脆弱性に照らしてコードの変更を検証します。

自動修復 - ユーザーがコードレビューの自動コード修復を有効にすると、AWS セキュリティエージェントは特定された脆弱性の修正を生成し、関連する GitHub リポジトリにプルリクエストを送信します。

侵入テストの修復 - 管理者がコンソールで検出結果の修復を有効にすると、ユーザーはウェブアプリケーションから侵入テストの検出結果の自動修復をリクエストできます。AWS セキュリティエージェントは、脆弱性に対処するためのコード修正を含む、関連付けられた GitHub リポジトリへのプルリクエストを開きます。

リソース階層とライフサイクルを理解する

AWS セキュリティエージェントは、セキュリティテストリソースを階層構造に整理し、組織全体で共有されるものと、アプリケーションごとにスコープされるものを決定します。この構造を理解することで、AWS セキュリティエージェントを効果的に設定し、さまざまなリソースを見つけて管理する場所を把握できます。

組織全体で共有される内容

AWS セキュリティエージェントの一部のリソースは、組織レベルで 1 回設定され、すべてのアプリケーションとエージェントスペースに適用されます。これらのテナントレベルのリソースは一貫性を提供し、重複する設定作業を減らします。

[リソース]	その内容とは	共有される理由
セキュリティ要件	AWS セキュリティエージェントが設計およびコードレビュー中に検証する内容を定義する組織のセキュリティ標準	セキュリティポリシーはすべてのアプリケーションに適用されます。それらを 1 回定義すると、AWS セキュリティエージェントはそれらをどこにでも適用します。
GitHub 統合	AWS セキュリティエージェントへの接続を許可された登録済みの GitHub 組織またはユーザーアカウント	GitHub 組織を一度登録し、必要に応じて特定のリポジトリを任意のエージェントスペースに接続します。
IAM Identity Center の設定	ユーザーが AWS セキュリティエージェントにアクセスする方法を制御する SSO 設定	一元化された ID 管理は、組織内のすべてのエージェントスペースに適用されます。

> **Important**
>
> セキュリティ要件の変更は、すべてのエージェントスペースで今後のすべての設計レビューとコードレビューに影響します。既存のレビューは影響を受けません。

エージェントスペースあたりのスコープ

各エージェントスペースは、保護する個別のアプリケーションまたはプロジェクトを表します。エージェントスペースレベルのリソースは、その特定のアプリケーションに限定されるため、さまざまなチームが独自の設定と評価を個別に操作できます。

[リソース]	その内容とは	アプリケーションごとにスコープが設定される理由
侵入テスト設定	アプリケーション内の特定の機能、API エンドポイント、または機能のテスト設定	各アプリケーションには、そのアプリケーションに固有の一意のターゲット、認証方法、スコープの境界があります。

[リソース]	その内容とは	アプリケーションごとにスコープが設定される理由
設計レビュー	設計ドキュメントの個々のアーキテクチャセキュリティ評価	各アプリケーションには、個別に評価される独自のアーキテクチャおよび設計ドキュメントがあります。
脅威モデル	システム概要を構築し、ソースコード、設計ドキュメント、またはその両方からの脅威を特定する脅威モデリング評価	各アプリケーションには独自のコードと設計があり、脅威は個別にモデル化されています。脅威モデルは、コードと設計の進化に応じて再実行できる再利用可能な設定です。
統合	このエージェントスペースに接続されたソースプロバイダーとドキュメントプロバイダー (GitHub、GitLab、Bitbucket、GitHub Enterprise Server、Confluence)	さまざまなアプリケーションは、さまざまなソースとドキュメントに依存します。エージェントスペースレベルで接続すると、アプリケーションの境界が明確になります。
コードレビュー設定	接続されたソース、スキャン設定、PR コメント有効化などのコードレビュー機能の設定	各アプリケーションには、独自のリポジトリとセキュリティレビューのニーズが個別に設定されています。
侵入テスト修復設定	ペネトレーションテストの検出結果に対する自動修正プルリクエストを受信できる接続されたリポジトリの設定	チームは、AWS セキュリティエージェントがアプリケーションのワークフローに基づいてコード変更を送信できる場所を制御します。
ユーザー割り当て	この特定のエージェントスペースにアクセスできるユーザー	チームには、担当するアプリケーションのセキュリティ評価のみが表示され、作業の整理と集中が維持されます。

> **Tip**
>
> チーム間の明確な境界を維持し、セキュリティ評価を効果的に整理するために、アプリケーションまたはプロジェクトごとに 1 つのエージェントスペースを作成することをお勧めします。

GitHub リポジトリが階層にどのように収まるか

GitHub リポジトリは、組織リソースを特定のアプリケーションに接続する複数ステップのプロセスを通じて統合されます。

1. テナントレベルで登録する - GitHub 組織またはユーザーアカウントの AWS セキュリティエージェント GitHub アプリを一度承認する
2. エージェントスペースレベルで接続 - 特定のリポジトリを選択して各エージェントスペースに接続します。
3. リポジトリごとに使用量を設定する - 接続されたリポジトリごとに特定の機能を有効にします。
   - コードレビュー - フルソースコードスキャンと自動プルリクエスト分析
   - 侵入テストコンテキスト - 侵入テスト中のソースコードからのアプリケーションの理解
   - 自動コード修復 - コードレビューと侵入テストの検出結果の脆弱性修正を含む自動プルリクエスト

1 つのリポジトリを複数のエージェントスペースに接続し、それぞれで異なる機能を有効にできます。

セキュリティ機能の主な違い

AWS セキュリティエージェントの各セキュリティ機能は、セキュリティチームがそれを使用する方法に基づいて異なるワークフローモデルに従います。

侵入テスト: 独立した実行による再利用可能な設定

ペネトレーションテストでは、反復セキュリティテストをサポートするconfiguration-and-runモデルを使用します。

- 1 回作成し、複数回実行する - スコープの境界、認証、テストパラメータを使用して、特定のターゲット (API エンドポイント、機能エリア) の設定を定義します。

- 独立した実行 - セキュリティを向上させるのと同じ設定を複数回実行します。各実行は独立しており、新しい検出結果を生成します。

このモデルは、改善を開発およびデプロイする際の継続的なセキュリティ検証をサポートします。

設計レビュー: クローン作成による 1 回限りの評価

設計レビューは、再利用可能な設定モデルに従わない独立した評価です。

- 単一の評価 - 各設計レビューは、アップロードされたドキュメントを組織のセキュリティ要件に照らして 1 回分析します。
- 再実行できない - 設計レビューは再利用できません。同じレビューを再実行することはできません
- 更新のクローン - 既存の設計レビューのクローンを作成して、元のドキュメントがプリロードされた新しいレビューを作成します。これにより、ドキュメントを更新して新しい分析を実行できます。

このモデルはpoint-in-timeアーキテクチャセキュリティ評価をサポートしています。

コードレビュー: オンデマンドスキャンと自動 PR 分析による再利用可能な設定

コードレビューには、ソースコードを保護するための 2 つのオペレーションモードが用意されています。

- フルコードレビュー (ウェブアプリケーション) - GitHub リポジトリまたは S3 ソースを選択するコードレビュー設定を作成し、オンデマンドで包括的なスキャンを実行します。各実行は、完全なソースコードにわたって静的分析を実行し、修復ガイダンスを含む結果を生成します。コードの進化と同じコードレビュー設定を再実行できます。
- プルリクエストコメント (GitHub) - 接続された GitHub リポジトリの自動分析を有効にします。AWS セキュリティエージェントは、プルリクエストがレビュー準備完了としてマークされると自動的にレビューし、セキュリティ検出結果をコメントとして GitHub に直接投稿します。

どちらのモードも、設定されたコードレビュー設定 (セキュリティの脆弱性、カスタム要件、またはその両方) を使用し、プルリクエストによるコードの自動修復をサポートします。

脅威モデル: オンデマンド実行による再利用可能な設定

脅威モデルは、アーキテクチャの反復評価をサポートするconfiguration-and-runモデルを使用します。

- 一度作成して複数回実行する – ソースコードをソースとして選択するか、設計ドキュメントをスコープドキュメントとしてアップロードするか、またはその両方を行うことで、脅威モデルを定義します。オンデマンドで実行し、コードと設計の進化に合わせて再実行します。
- 柔軟な入力 – ソースコードのみ、設計ドキュメントのみ、またはその両方で脅威モデルを実行します。スコープドキュメントは、エージェントが分析に焦点を当てる内容を定義します。ソースコードは、既存のシステムに関するコンテキストを提供します。
- システムの概要と脅威 – 各実行では、アプリケーションのアーキテクチャ、信頼境界、データフロー、セキュリティ体制を説明するシステムの概要と、重要度、証拠、実用的な推奨事項を含む STRIDE カテゴリ別に分類された一連の脅威が表示されます。

リソース関係について

階層によって、さまざまなリソースを設定してアクセスする場所が決まります。

AWS マネジメントコンソールで:

- テナントレベルのリソースを設定する (セキュリティ要件、GitHub 統合、IAM Identity Center)
- エージェントスペースの作成と管理
- エージェントスペースの設定を構成する (接続されたリポジトリ、コードレビューの有効化、ペネトレーションテストの修正)

セキュリティエージェントウェブアプリケーションの場合:

- 侵入テスト設定とテスト実行の作成と管理
- 設計レビューの作成と管理
- 接続されたリポジトリと S3 ソースに対してコードレビューを作成、管理、実行する
- ソースコード、スコープドキュメント、またはその両方に対して脅威モデルを作成、管理、実行する
- 侵入テスト、コードレビュー、設計レビュー、脅威モデルの結果を表示する

GitHub の場合:

- プルリクエストコードレビューの結果をプルリクエストコメントとして表示する
- コードレビューとペネトレーションテストの検出結果に対する自動修復プルリクエストを受信する (エージェントスペースで有効になっている場合)

> **Note**
>
> プルリクエストコードレビューの結果が GitHub に表示されます。完全なコードレビュー、ペネトレーションテスト、設計レビューの結果は、 セキュリティエージェントウェブアプリケーションに表示されます。

はじめに

サインアップし、目標に適した開始点を見つけ、クイックスタートで最初の評価を実行します。

AWS アカウントにサインアップする

の使用を開始するには AWS、 AWS アカウントが必要です。 AWS アカウントの作成の詳細については、「 AWS アカウント管理リファレンスガイド」の「 アカウントの開始方法」を参照してください。 AWS

進めたい作業内容を選択します

AWS セキュリティエージェントで実行する適切な開始点を見つけます。以下の表を使用して、目標とタスクを照合し、そのページに直接移動します。AWS セキュリティエージェントを初めて使用する場合は、the section called “AWS セキュリティエージェントとは”まず「」を読んで概要を確認してください。

次の操作を行います。	誰が	ここから開始する
AWS セキュリティエージェントが設定する前に行うことを理解する	[全員]	the section called “AWS セキュリティエージェントとは”
サービスを設定し、エージェントスペースを作成する	管理者	the section called “AWS セキュリティエージェントをセットアップする”
ライブアプリケーションまたはデプロイ済みアプリケーションをテストして悪用可能な脆弱性がないか調べる	ユーザー	the section called “クイックスタート: 侵入テストを実行する”
ソースコードをスキャンして脆弱性やポリシー違反がないか確認する	ユーザー	the section called “クイックスタート: コードレビューを実行する”
アプリケーションの攻撃方法のモデル化 (STRIDE)	ユーザー	the section called “クイックスタート: 脅威モデルを実行する”

次の操作を行います。	誰が	ここから開始する
コードを記述する前に、設計に関するセキュリティフィードバックを取得する	ユーザー	the section called “設計レビューを作成する”
IDE を離れずにコードをスキャンする (Kiro または Claude コード)	開発者	the section called “IDE からコードセキュリティスキャンを実行する”
マージする前に、変更された行のみをスキャンする	開発者	the section called “S3 で差分コードスキャンを実行する”
プルリクエストとマージリクエストに関する自動セキュリティコメントを取得する	管理者	the section called “コードレビューを有効にする”
結果を確認し、最初に修正すべき点を決定する	ユーザー	侵入テスト、コードレビュー、脅威モデル、設計レビュー

> **Note**
>
> AWS セキュリティエージェントは、作業するインターフェイスで識別できる 3 つのロールを使用します。管理者は AWS マネジメントコンソールで作業し (設定と設定)、ユーザーはウェブアプリケーションで作業し (評価の実行と結果の確認)、開発者は GitHub または Kiro や Claude Code などの IDE で作業します。1 人のユーザーが複数のロールを持つことができます。定義の詳細については、「」を参照してくださいthe section called “AWS セキュリティエージェントの仕組み”。

クイックスタート: 侵入テストを実行する

このクイックスタートでは、AWS セキュリティエージェントで最初のペネトレーションテスト (ペンテスト) を実行する手順を説明します。ペネトレーションテストでは、デプロイされたアプリケーションを検証済みターゲットドメインに対して演習し、セキュリティ検出結果を返します。各検出結果には、重要度評価と証拠が示されます。パブリックにアクセス可能なアプリケーションについて説明します。プライベート VPC でホストされているアプリケーションをテストするには、「」を参照してくださいthe section called “ペネトレーションテストを有効にする”。

> **Note**
>
> AWS セキュリティエージェントをセットアップしてテストスコープを定義するには AWS マネジメントコンソールにアクセスし、ペネトレーションテストを作成して実行するにはウェブアプリケーションにアクセスする必要があります。

前提条件

開始する前に、以下の準備が整っていることを確認します。

- AWS セキュリティエージェントをセットアップするアクセス許可を持つ AWS マネジメントコンソールへのアクセス。
- テストするアプリケーションをホストするライブターゲットドメイン。
- 所有権を検証できるように、そのドメインの DNS レコード、または同じ AWS アカウントの Route 53 ホストゾーンを追加する機能。
- (オプション) アプリケーションのソースコードリポジトリ (GitHub、GitLab、または Bitbucket)。エージェントにコンテキストを提供します。

ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する

AWS セキュリティエージェントをまだセットアップしていない場合は、初期設定を完了します。

1. AWS マネジメントコンソールで AWS セキュリティエージェントに移動します。
2. AWS セキュリティエージェントのセットアップ を選択します。
3. エージェントスペースを作成します。エージェントスペースは複数のユーザーが使用でき、テストするすべてのアプリケーションに固有である必要があります。名前と説明を入力します。名前は、ペネトレーションテストするアプリケーションを識別する必要があります。
4. ユーザーアクセス設定で IAM のみのアクセスを選択します。
   - このクイックスタートでは、ユーザーが AWS コンソールから直接ウェブアプリケーションにアクセスできるようにする IAM アイデンティティセンターでのシングルサインオン (SSO) の有効化については説明していません。
   - AWS マネジメントコンソール を使用しないユーザーがペネトレーションテストを開始できるようにするには、IAM Identity Center 統合を有効にします。詳細については、「the section

called “ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する”」を参照してください。

5. AWS セキュリティエージェントのセットアップ を選択します。

> **Note**
>
> セットアップを選択すると、AWS セキュリティエージェントはエージェントスペースを作成し、ユーザーが侵入テスト、コードレビュー、脅威モデル、設計レビューを実行できるウェブアプリケーションを確立します。

ステップ 2: ペネトレーションテストを有効にしてドメインを検証する

> **Note**
>
> AWS コンソールでは、テストできる範囲を定義します。その後、ユーザーはウェブアプリケーションからその範囲内で特定の侵入テストを実行します。

1. 左側のサイドバーから、エージェントスペースを選択し、ステップ 1 で作成したエージェントスペースを選択します。
2. ペネトレーションテストタブを選択し、ペネトレーションテストの設定を選択してウィザードを開きます。
3. ドメインの設定 - テストするターゲットドメインを入力し、検証方法、DNS TXT レコード、または HTTP ルートを選択します。ドメインはライブで、テストするアプリケーションをホストする必要があります。[次へ] を選択します。
4. ドメインの検証 - ターゲットドメインテーブルの各ドメインの所有権を検証します。AWS セキュリティエージェントは、検証済みドメインに対してのみテストを実行します。コピーする詳細な手順と正確な値については、「」を参照してくださいthe section called “ペネトレーションテスト用のアプリケーションドメインを有効にする”。
   - 同じ AWS アカウントの Route 53 ドメイン - ドメインを選択し、ワンクリック検証を選択します。AWS セキュリティエージェントは DNS レコードを作成し、検証を完了します。
   - DNS TXT レコード (他の DNS プロバイダー) - ターゲットドメインテーブルで、レコード名とレコード値をコピーし、DNS プロバイダーで TXT レコードとして追加してから、ドメイ

ンを選択して検証を選択します。DNS の変更が反映されるまでに時間がかかる場合があるため、検証がすぐに完了しない場合があります。

- HTTP ルート - ウェブサーバー.well-known/aws/securityagent-domain-verification.jsonで にファイルを作成し、 形式で検証トークンを追加し{"tokens": ["<token>"]}、ドメインを選択して検証を選択します。

5. (オプション) 追加機能を設定する - CloudWatch ロググループや認証情報などのオプションリソースを追加します。サービスアクセスセクションは事前設定されています。AWS セキュリティエージェントは、既存の IAM ロールを選択しない限り、必要なアクセス許可を持つサービスロールを作成します。

ステップ 3: ソースコードを接続する (オプション)

ソースコードプロバイダーを接続すると、AWS セキュリティエージェントにアプリケーションに関するコンテキストが提供され、ペネトレーションテストのカバレッジが向上します。この手順は省略可能です。

1. 侵入テストタブで、ページ上部の侵入テストバナーのソースコードプロバイダーを接続するで追加を選択します。
2. プロバイダーを登録して接続し、ペネトレーションテストに使用できるリポジトリを選択します。

完全な統合フローについては、「AWS セキュリティエージェントを GitHub リポジトリに接続する」またはプロバイダーの接続トピックを参照してください。

ステップ 4: 侵入テストを作成して実行する

> **Note**
>
> AWS Security Agent ウェブアプリケーションでペネトレーションテストを作成して実行します。本番環境に近いテスト環境に対してテストを実行することをお勧めします。

1. ウェブアプリケーションを起動します。ウェブアプリケーションタブを選択し、管理者アクセスを選択します。
2. 左側のサイドバーで、ペネトレーションテストを選択し、ペネトレーションテストを作成します。

3. 侵入テストの詳細 - テストスコープとログソースを設定します。
   a. Pentest 名を入力します。
   b. ターゲット URLs で、テストする検証済みドメイン (など) を選択または入力しますhttps://example.com。検証済みドメインのみがテストでき、サブドメインは自動的にカバーされます。
   c. (オプション) アクセス可能な URLs で、エージェントがログインとナビゲーションのために到達する必要があるが、ターゲットドメイン外の ID プロバイダー、CDNs、APIsなど、攻撃すべきではないドメインを追加します。AWS セキュリティエージェントはこれらのドメインに到達できますが、テストは行いません。ターゲットドメインのみが攻撃されます。
   d. ネットワーク設定ルールを確認して、テスト中にトラフィックを受信できるエンドポイントと受信できないエンドポイントを確認します。
   e. アクセス許可で、サービスロールを選択し、オプションで CloudWatch ロググループを選択します。[次へ] を選択します。
4. (オプション) VPC リソース - ターゲットがパブリックにアクセスできないプライベートネットワーク上にある場合は、VPC を設定します。「the section called “ペネトレーションテストを有効にする”」を参照してください。
5. (オプション) 認証リソース - エージェントが認証された非公開のパスに到達できるように認証情報を提供します。これらを追加してカバレッジを広げ、ログインの背後にある脆弱性を明らかにすることをお勧めします。
6. (オプション) 追加設定 - ファイル、GitHub リポジトリ、S3 ソースなどのアプリケーションコンテキストを追加して、検出結果の品質を向上させます。自動コード修復を有効にし、ここで他の実行オプションを設定することもできます。
7. Create and execute を選択して今すぐテストを開始するか、Create pentest を選択して保存し、後で実行します。

ステップ 5: 侵入テストの結果を確認する

1. ペネトレーションテストが完了するまでに数時間かかる場合があります。ほとんどは、アプリケーションのサイズと複雑さに応じて、16 時間以内に完了します。
2. 実行は、テストを開始する前にセットアップを検証するプリフライトフェーズから始まります。ロギングインフラストラクチャを設定し、ターゲットエンドポイントに到達可能かどうかをチェックし、テスト環境を準備します。プリフライトチェックが失敗した場合 (解決できないターゲットドメインなど) 、テストを開始する前に実行が停止します。プリフライトタブを開いて、失敗したチェックを確認し、解決して、新しい実行を開始します。

3. 実行が完了したら、実行を開き、実行の概要、ログ、結果タブを確認します。
4. 検出結果タブで、検出結果を選択して、その説明、重要度、リスクタイプ、および補足証拠を表示します。

詳細については、「the section called “侵入テストを作成する”」および「the section called “ペネトレーションテストの結果を確認する”」を参照してください。

クイックスタート: コードレビューを実行する

このクイックスタートでは、AWS セキュリティエージェントで最初のコードレビューを実行する手順を説明します。AWS Security Agent は、ソースコードリポジトリをスキャンして、セキュリティの脆弱性と組織のセキュリティ要件への準拠を確認します。

> **Note**
>
> AWS セキュリティエージェントをセットアップするには AWS マネジメントコンソールにアクセスし、コードレビューを作成して実行するにはウェブアプリケーションにアクセスする必要があります。

前提条件

開始する前に、以下の準備が整っていることを確認します。

- AWS セキュリティエージェントをセットアップするアクセス許可を持つ AWS マネジメントコンソールへのアクセス。
- レビューするコードを含むソースコードリポジトリ (GitHub、GitLab、または Bitbucket) または Amazon S3 ソース。

ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する

AWS セキュリティエージェントをまだセットアップしていない場合は、初期設定を完了します。

1. AWS マネジメントコンソールで AWS セキュリティエージェントに移動します。
2. AWS セキュリティエージェントのセットアップ を選択します。

3. エージェントスペースを作成します。エージェントスペースは複数のユーザーが使用でき、テストするすべてのアプリケーションに固有である必要があります。最初のエージェントスペースの名前と説明を入力します。この名前は、ウェブアプリケーションのユーザーに表示されます。名前は、コードを確認するアプリケーションを識別する必要があります。
4. ユーザーアクセス設定で IAM のみのアクセスを選択します。
   - このクイックスタートでは、IAM Identity Center でのシングルサインオン (SSO) の有効化については説明していません。これにより、ユーザーは AWS コンソールから AWS セキュリティエージェントのウェブアプリケーションに直接アクセスできます。
   - AWS マネジメントコンソール にアクセスしないユーザーがコードレビューを実行できるようにするには、IAM Identity Center 統合を有効にします。詳細については、「the section called “ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する”」を参照してください。
5. AWS セキュリティエージェントのセットアップ を選択します。

> **Note**
>
> セットアップを選択すると、AWS セキュリティエージェントはエージェントスペースを作成し、ユーザーが侵入テスト、コードレビュー、脅威モデル、設計レビューを実行できるウェブアプリケーションを確立します。

ステップ 2: コードレビューを有効にして設定する

> **Note**
>
> GitHub リポジトリまたは S3 バケットが既にエージェントスペースに接続されている場合 (侵入テストのセットアップなど) 、コードレビューは既に有効になっています。このステップをスキップして、ウェブアプリケーションに直接移動できます。

コードレビュー設定ウィザードを開きます。

1. 左側のサイドバーから、エージェントスペースを選択し、エージェントスペースを選択します。
2. ヘッダーまたはコードレビュータブからコードレビューを有効にするを選択します。

統合、リポジトリ、バケットを接続する

1. (GitHub 統合がまだない場合) GitHub 登録を作成します。すでにある場合は、次のステップに進みます。
   a. 「Connected integrations」セクションで、「追加」と「新規登録の作成」を選択します。
   b. GitHub を選択し、次へを選択します。
   c. インストールと認可を選択し、GitHub でインストールを完了します。
      i. レビューするリポジトリを所有する GitHub ユーザーまたは組織を選択します。
      ii. すべてのリポジトリを選択するか、リポジトリのみを選択します。
      iii. Install & Authorize を選択し、GitHub 認証を完了します。
   d. AWS マネジメントコンソールに戻り、登録名を入力し、アカウントタイプが GitHub アプリをインストールした場所と一致することを確認します。
   e. Connect を選択して登録を保存します。

   GitHub 統合フローの詳細については、「AWS セキュリティエージェントを GitHub リポジトリに接続する」を参照してください。
2. GitHub リポジトリを接続します。接続された統合セクションで、追加を選択し、GitHub 登録を選択します。2 ステップの Connect GitHub ウィザードが開きます。
   a. Connect GitHub リポジトリで、含めるリポジトリを選択し、次へを選択します。
   b. 管理機能で、リポジトリごとに以下を切り替えます。
      - コードレビューコメント – AWS セキュリティエージェントがセキュリティ検出結果をリポジトリ内のプルリクエストに対するコメントとして投稿できるようにします。
      - 自動修復 – AWS Security Agent ウェブアプリケーションのユーザーが、検出結果を修正するリクエストをプルできるようにします。
   c. 保存を選択してセットアップウィザードに戻ります。
3. (オプション) S3 ソースを接続します。S3 バケットセクションで、S3 リソースの追加を選択し、ソースコードを含むバケットの S3 URI を入力するか、参照を選択して選択します。
4. コードレビュー設定を選択します。デフォルトのセキュリティ要件と脆弱性の検出結果は、有効にしたセキュリティ要件と一般的な脆弱性の両方への準拠についてコードを分析します。
5. [次へ] を選択します。

オプションの設定

1. オプションの CloudWatch ロググループとサービスアクセスを設定します。デフォルトのサービスロールは、必要なアクセス許可で事前設定されています。
2. [保存] を選択します。

ステップ 3: コードレビューを作成して実行する

> **Note**
>
> コードレビューを作成および実行するのは、AWS セキュリティエージェントのウェブアプリケーションのみです。

1. ウェブアプリタブを選択し、管理者アクセスを選択して AWS セキュリティエージェントのウェブアプリケーションを起動します。
2. 左側のサイドバーで、コードレビューを選択します。
3. コードレビューの作成 を選択します。
4. コードレビューを設定します。
   a. このレビューの範囲 (billing-service-security-review」など) を識別するタイトルを入力します。
   b. ソースで、ステップ 2 でエージェントスペースに接続した GitHub リポジトリを選択するか、スキャンする S3 ソースを入力します。
   c. 設定したロールからサービスロールを選択します。
   d. (オプション) 自動コード修復を有効にするを選択すると、AWS セキュリティエージェントがすべての検出結果の修正を含むプルリクエストを自動的に送信します。
5. コードレビューの作成 を選択します。
6. コードレビューの詳細ページで、レビューの開始を選択します。

ステップ 4: コードレビューの結果を確認する

1. コードレビューには通常、コードベースのサイズに応じて 30～60 分かかります。
2. 実行は、スキャンを開始する前にセットアップを検証するプリフライトフェーズから始まります。AWS セキュリティエージェントがリポジトリまたは S3 ソースからソースコードをプルし、スキャン環境をセットアップできることを確認します。プリフライトチェックが失敗した場合 (た

とえば、ソースにアクセスできない場合) 、実行は静的分析の前に停止します。プリフライトタブを開いて、失敗したチェックを確認し、解決して、新しい実行を開始します。

3. 完了したら、完了した実行に移動し、検出結果タブを選択します。
4. list-detail ビューで結果を確認します。
   a. 左側のパネルから検出結果を選択すると、その詳細が表示されます。
   b. 説明、コードの場所、証拠、推奨される修正セクションを確認します。
   c. 修正コードを使用して修正を含むプルリクエストを生成するか、そのオプションを有効にした場合は自動修復 PRsを確認します。

詳細については、「the section called “コードレビューを作成する”」および「the section called “コードレビューの結果を確認する”」を参照してください。

クイックスタート: 脅威モデルを実行する

このクイックスタートでは、AWS セキュリティエージェントで最初の脅威モデルを実行する手順を説明します。脅威モデルは、アプリケーションのアーキテクチャを分析し、システムの概要 (AWS Security Agent がシステムを理解する方法) と一連の脅威 (それぞれが重要度レベル、STRIDE 分類、推奨事項を持つ攻撃方法) を生成します。設計ドキュメント (スコープドキュメント) で脅威モデルを実行して、フォーカス、ソースコード (ソース) を定義し、既存のシステムに関するコンテキスト、またはその両方を提供できます。

> **Note**
>
> AWS セキュリティエージェントをセットアップするには AWS マネジメントコンソールにアクセスし、脅威モデルを作成して実行するにはウェブアプリケーションにアクセスする必要があります。

ステップ 1: AWS コンソールで AWS セキュリティエージェントを設定する

AWS セキュリティエージェントをまだセットアップしていない場合は、初期設定を完了します。

1. AWS マネジメントコンソールで AWS セキュリティエージェントに移動します。
2. AWS セキュリティエージェントのセットアップ を選択します。

3. エージェントスペースを作成します。エージェントスペースは複数のユーザーが使用でき、保護するアプリケーションごとに固有である必要があります。最初のエージェントスペースの名前と説明を入力します。名前は、脅威モデルにするアプリケーションを識別する必要があります。
4. ユーザーアクセス設定で IAM のみのアクセスを選択します。
   - このクイックスタートでは、IAM Identity Center でのシングルサインオン (SSO) の有効化については説明していません。これにより、ユーザーは AWS コンソールから AWS セキュリティエージェントのウェブアプリケーションに直接アクセスできます。
   - AWS マネジメントコンソールにアクセスできないユーザーが脅威モデルの開始などのタスクを実行できるようにする場合は、IAM アイデンティティセンターの統合を有効にする必要があります。
5. AWS セキュリティエージェントのセットアップ を選択します。

> **Note**
>
> セットアップを選択すると、AWS セキュリティエージェントはエージェントスペースを作成し、ユーザーが侵入テスト、コードレビュー、脅威モデル、設計レビューを実行できるウェブアプリケーションを確立します。

ステップ 2: ソースコードを接続する

> **Note**
>
> エージェントスペースに接続されているリポジトリまたは S3 バケットが既にある場合 (コードレビューや侵入テストのセットアップなど) 、このステップをスキップしてウェブアプリケーションに直接移動できます。ソースコードを接続せずに、アップロードされたスコープドキュメントで脅威モデルをいつでも実行できます。

既存のシステムを理解するためのコンテキストとしてエージェントが使用するソースコードを含むリポジトリまたは S3 バケットを接続します。

1. 左側のサイドバーから、エージェントスペースを選択し、エージェントスペースを選択します。
2. 統合セクションに移動して、ソースコードプロバイダーを接続します。
3. または、ソースコードを含む S3 バケットを接続します。

完全な統合フローについては、「AWS セキュリティエージェントを GitHub リポジトリに接続する」を参照してください。

ステップ 3: 脅威モデルを作成して実行する

> **Note**
>
> AWS Security Agent ウェブアプリケーションで脅威モデルを作成して実行します。

1. AWS マネジメントコンソールのウェブアプリタブからウェブアプリケーションを起動します。または、IAM Identity Center を設定している場合は、直接ログインします。
2. 左側のサイドバーで、脅威モデルを選択します。
3. 「脅威モデルの作成」を選択します。
4. 脅威モデルを設定します。
   a. この脅威モデルの範囲 (「billing-service」や「checkout-api」など) を識別するタイトルを入力します。
   b. 少なくとも 1 つの入力を指定します。
      - スコープドキュメントで、設計ドキュメント (DOC、DOCX、JPEG、MD、PDF、PNG、TXT) をアップロードするか、S3 URIs を入力するか、Confluence ページを選択します。
      - ソース で、接続された統合からリポジトリを選択するか、ソースコードを含む S3 URIs を入力します。

      > **Tip**
      >
      > ソースドキュメントとスコープドキュメントの両方を提供して、脅威モデルを特定の設計 (機能設計ドキュメントなど) にスコープし、エージェントに既存のシステムを理解するためのコンテキストとしてソースコードを提供します。

   c. 設定したロールからサービスロールを選択します。
   d. (オプション) CloudWatch ログのロググループを選択します。
5. 「脅威モデルの作成」を選択します。
6. 脅威モデルの詳細ページで、実行の開始を選択します。

ステップ 4: 脅威を確認する

1. 実行は分析タスクを通じて進行します。プリフライトタブで進行状況をモニタリングし、ログタブでタスクの詳細を表示できます。
2. 完了すると、実行ステータスは完了に変わります。
3. 概要タブを選択して、システムの概要と重要度の分布を確認します。
4. 脅威タブを選択して、特定された脅威を確認します。
   a. リストから脅威を選択すると、サイドパネルにその詳細が表示されます。
   b. ステートメント、重要度、STRIDE カテゴリ、推奨事項、証拠、およびその他のフィールドを確認します。
   c. 各脅威に対処またはトリアージするときに、脅威のステータスを解決済みまたは拒否済みに更新します。

詳細については、「the section called “脅威モデルを作成する”」および「the section called “脅威モデルからの脅威を確認する”」を参照してください。

管理者向け (コンソール)

管理者は、AWS マネジメントコンソールで AWS セキュリティエージェントを設定し、AWS セキュリティエージェントのウェブアプリケーションを介してユーザーがアクセスするエージェントスペースを設定します。各エージェントスペースは、特定のアクセス許可とリソースを持つ個別の環境を表します。

テストするアプリケーションごとに一意のエージェントスペースを作成することをお勧めします。たとえば、請求アプリケーションとタスク追跡アプリケーションという 2 つの内部プロジェクトがある場合は、2 つの個別のエージェントスペースを作成する必要があります。

設定例

管理者がエージェントスペースをセットアップして、内部請求アプリケーションのセキュリティを評価するとします。管理者は次の操作を行います。

- ドメインを検証する (などbeta.billing.example.com)
- GitHub に接続し、コードレビューを有効にする
- ペネトレーションテスト用の適切な VPC、サブネット、セキュリティグループを割り当ててネットワークアクセスを設定する

ユーザーが侵入テストまたは設計レビューを開始すると、これらの事前設定されたリソースから選択し、定義したガードレール内で作業しながら、特定の評価ニーズに対する柔軟性を維持できます。

エージェントスペースの設定と作成

組織の AWS セキュリティエージェントを設定し、各アプリケーションのリソースと評価をスコープするエージェントスペースを作成します。

AWS セキュリティエージェントをセットアップする

最初のエージェントスペースを作成し、ユーザーがウェブアプリケーションにアクセスする方法を確立して、組織の AWS セキュリティエージェントを設定します。

この初期設定により、管理者は AWS コンソールでセキュリティ機能を設定し、セキュリティエージェントウェブアプリケーションを通じて設計レビューとペネトレーションテストへのアクセス権

をユーザーに付与できます。この 1 回限りのセットアップの後、シンプルなプロセスで追加のエージェントスペースを作成できます。

概要

エージェントスペースについて

エージェントスペースは、特定のアプリケーションまたはプロジェクトを保護するための専用ワークスペースです。これには、すべてのセキュリティレビュー、オプションで接続された GitHub リポジトリ、侵入テスト設定、結果、そのアプリケーションの検出結果が含まれます。

エージェントスペースは、各アプリケーションのセキュリティ評価を分けておき、チームが特定のアプリケーションに集中できるようにすることで、セキュリティ作業を整理するのに役立ちます。明確な境界を維持するために、アプリケーションまたはプロジェクトごとに 1 つのエージェントスペースを作成することをお勧めします。

セキュリティ要件は組織レベルで定義され、すべてのエージェントスペースに適用されます。一方、各エージェントスペースは独自の設計ドキュメント、コードリポジトリ、侵入テスト設定、侵入テスト結果、セキュリティ検出結果を保持します。

エージェントスペースに含まれる内容

各エージェントスペースには以下が含まれます。

- アプリケーションまたはプロジェクトに関連付けられたオプションで接続された GitHub リポジトリ
- アプリケーションの以前の設計レビュー
- アプリケーション固有の侵入テストの設定と境界
- 侵入テストのテスト結果とセキュリティの検出結果

セットアップ時に設定するもの

この初回セットアップでは、すべてのエージェントスペースに適用される組織的な決定を行います。

- アクセス方法 - ユーザーがセキュリティエージェントウェブアプリケーションにアクセスする方法を選択します (IAM Identity Center SSO または AWS コンソールを介した IAM のみのアクセス)
- アクセス許可 - ウェブアプリケーションが AWS のサービスにアクセスするために使用する IAM ロールを設定します。
- 最初のエージェントスペース - 名前と説明を使用して最初のエージェントスペースを作成します。

> **Note**
>
> この設定を完了すると、追加のエージェントスペースの作成がより簡単になります。名前と説明を入力するだけです。後続のエージェントスペースの作成の詳細については、the section called “エージェントスペースを作成する”「」を参照してください。

前提条件

開始する前に、以下があることを確認してください。

- AWS Security Agent リソースを作成および管理するためのアクセス許可
- 組織の ID 管理要件を理解する
- (オプション) IAM ロールを作成し、IAM Identity Center を設定するアクセス許可を持つ AWS アカウント (SSO アクセスを使用する場合)
- (オプション) カスタムアクセス許可設定を使用する場合の既存の IAM ロール

ステップ 1: 最初のエージェントスペースを作成する

ウェブアプリケーションのユーザーに表示される最初のエージェントスペースの基本プロパティを定義します。

1. AWS セキュリティエージェントのコンソールページで、セキュリティエージェントの設定を選択します。
2. エージェントスペース名フィールドに、エージェントスペースの名前を入力します。

> **Note**
>
> エージェントスペース名はウェブアプリケーションのユーザーに表示され、このスペースが表すアプリケーションまたはプロジェクトを識別するのに役立ちます。

3. (オプション) 説明 フィールドに、エージェントスペースの目的の区別に役立つ説明を入力します。

> **Tip**
>
> この説明は、エージェントスペースの目的を区別するのに役立ちます。「カスタマーポータルウェブアプリケーション」、「支払い処理マイクロサービス」、「内部分析プラット

フォーム」など、このエージェントスペースが保護する特定のアプリケーションまたはプロジェクトを説明することをお勧めします。

ステップ 2: アクセス方法を選択する

ユーザーがセキュリティエージェントウェブアプリケーションにアクセスする方法を選択します。IAM Identity Center 経由で SSO アクセスを有効にするか、AWS コンソール経由でアクセスを提供するかを選択します。

> **Note**
>
> IAM のみのアクセスを選択し、後で IAM Identity Center を使用する場合は、AWS セキュリティエージェントのセットアップを削除し、セットアッププロセスを再度完了する必要があります。

1. アクセス方法セクションで、次のいずれかのオプションを選択します。
   - IAM アイデンティティセンター (SSO) - IAM アイデンティティセンターを通じてチームの SSO アクセスを有効にします。このオプションは、一元的なユーザー管理が必要で、ユーザーが AWS コンソールを経由せずにウェブアプリケーションに直接アクセスできるようにするチームに推奨されます。
   - IAM のみのアクセス - AWS コンソールの管理者アクセスリンクを介してアクセスを提供します。このオプションはセットアップが簡単で、コンソールベースのアクセスを好むチームや SSO 機能を必要としないチームに適しています。
2. IAM アイデンティティセンター (SSO) を選択した場合は、以下のステップ 2a に進みます。
3. IAM のみのアクセスを選択した場合は、ステップ 3 に進みます。

ステップ 2a: IAM アイデンティティセンターを設定する (SSO アクセスのみ)

アクセス方法として IAM アイデンティティセンター (SSO) を選択した場合のみ、これらのステップを実行します。

1. 「IAM アイデンティティセンターへの接続」セクションで、AWS リージョンを確認します。

> **Important**
>
> アプリケーションは、IAM Identity Center を有効にするリージョンと同じリージョンで設定する必要があります。表示されるリージョンは、エージェントスペースが作成されるリージョンです。IAM アイデンティティセンターは、エージェントスペースを作成するリージョンと同じリージョンで有効にする必要があります。

2. IAM Identity Center セクションで、次のいずれかを選択します。
   - アカウントインスタンスの作成を選択して、新しい IAM Identity Center アカウントインスタンスを作成します。
   - 組織インスタンスが同じリージョンに既に存在する場合、AWS セキュリティエージェントは自動的にそのインスタンスに接続します。

> **Note**
>
> AWS セキュリティエージェントが IAM アイデンティティセンターに接続されると、IAM アイデンティティセンターのユーザーをアプリケーションに割り当てることでアクセスを管理できます。

3. Active Directory または外部 ID プロバイダーを接続する場合は、情報アラートを確認します。

> **Important**
>
> Active Directory または別の ID ソースでユーザーをすでに管理していて、その ID ソースを IAM アイデンティティセンターに接続する予定がある場合は、セットアップをキャンセルし、まず IAM アイデンティティセンターコンソールに移動します。AWS セキュリティエージェントを設定する前に、接続する ID ソースを選択します。ID ソースを後で変更すると、既存のユーザー割り当てが削除される可能性があります。

ステップ 3: アクセス許可を設定する (オプション)

Security Agent Web Application が AWS のサービス、APIs、およびアカウントにアクセスするために使用する IAM ロールを設定します。

1. アクセス許可設定 - オプションセクションを見つけます。
2. セクションが折りたたまれている場合は、アクセス許可設定セクションを選択して展開します。

3. 以下のオプションのいずれかを選択します。
   - デフォルトロールの作成 - AWS セキュリティエージェントは、ウェブアプリケーションに必要なアクセス許可を持つ新しい IAM ロールを自動的に作成します。
   - 別のロールを使用する - 使用可能なオプションから既存の IAM ロールを選択します。
4. ロールの説明を確認します。

> **Note**
>
> ウェブアプリケーションが他の AWS のサービス、APIs、およびアカウントにアクセスするためのデフォルトの IAM ロールが作成されます。ロールには、セキュリティ評価オペレーションに必要なアクセス許可が付与されます。

ステップ 4: セットアップを完了する

必要な設定をすべて設定したら、セットアップを完了して最初のエージェントスペースを作成し、セキュリティエージェントウェブアプリケーションを確立します。

1. すべての設定セクションを確認して、正確性を確認します。
2. [設定] を選択します。
3. AWS セキュリティエージェントは、エージェントスペースを作成し、ウェブアプリケーションを確立して、必要な AWS リソースを設定します。

> **Note**
>
> 初期設定後、ペネトレーションテストの境界、セキュリティ要件、GitHub 統合などの機能を設定するオプションがあります。

次の手順

AWS セキュリティエージェントを設定した後:

- 組織のセキュリティ要件を定義する
- エージェントスペースのドメイン検証や VPC アクセスなどの侵入テスト機能を設定する
- コードレビューと侵入テストコンテキストのために GitHub リポジトリを接続する

- (IAM アイデンティティセンターを使用している場合) AWS セキュリティエージェントコンソールのエージェントスペースにユーザーを割り当てる
- (IAM 専用アクセスを使用している場合) AWS コンソールの管理者アクセスリンクからウェブアプリケーションを起動します。
- 他のアプリケーション用の追加のエージェントスペースを作成する (「」を参照the section called “エージェントスペースを作成する”)

エージェントスペースを作成する

エージェントスペースを作成して、組織内のアプリケーションまたはプロジェクトを保護します。各エージェントスペースには、そのアプリケーションまたはプロジェクトに固有のセキュリティ評価、検出結果、設定用のワークスペースが用意されており、複数のユーザーがアクセスできます。

この手順は、AWS セキュリティエージェントの初期セットアップをすでに完了していることを前提としています。AWS セキュリティエージェントをまだ設定していない場合は、「」を参照してくださいthe section called “AWS セキュリティエージェントをセットアップする”。

概要

エージェントスペースについて

エージェントスペースは、特定のアプリケーションまたはプロジェクトを保護するための専用ワークスペースです。これには、そのアプリケーションのすべてのセキュリティレビュー、オプションで接続されたコードリポジトリ、侵入テスト設定、結果、検出結果が含まれます。

エージェントスペースは、各アプリケーションのセキュリティ評価を分けておき、チームが特定のアプリケーションに集中できるようにすることで、セキュリティ作業を整理するのに役立ちます。明確な境界を維持するために、アプリケーションまたはプロジェクトごとに 1 つのエージェントスペースを作成することをお勧めします。

エージェントスペースの包括的な説明と、それらが組織のセキュリティ構造にどのように適合するかについては、「」を参照してくださいthe section called “リソース階層とライフサイクルを理解する”。

エージェントスペースに含まれる内容

各エージェントスペースには以下が含まれます。

- アプリケーションまたはプロジェクトに関連付けられたオプションで接続されたコードリポジトリ

- アプリケーションまたはプロジェクトの以前の設計レビュー
- アプリケーション固有の侵入テストの設定と境界
- 侵入テストのテスト結果とセキュリティの検出結果

エージェントスペースを作成する

保護するアプリケーションまたはプロジェクトの新しいエージェントスペースを作成します。

1. AWS セキュリティエージェントコンソールで、エージェントスペースページに移動します。
2. エージェントスペースの作成を選択します。
3. エージェントスペース名フィールドに、エージェントスペースの名前を入力します。

> **Note**
>
> エージェントスペース名はウェブアプリケーションのユーザーに表示され、このスペースが表すアプリケーションまたはプロジェクトを識別するのに役立ちます。

4. (オプション) Description フィールドに、エージェントスペースの目的の区別に役立つ説明を入力します。

> **Tip**
>
> この説明は、エージェントスペースの目的を区別するのに役立ちます。「カスタマーポータルウェブアプリケーション」、「支払い処理マイクロサービス」、「内部分析プラットフォーム」など、このエージェントスペースが保護する特定のアプリケーションまたはプロジェクトを説明することをお勧めします。

5. [作成] を選択します。

> **Note**
>
> AWS セキュリティエージェントがエージェントスペースを作成します。機能を設定し、このアプリケーションに固有のリソースを接続できるようになりました。

次の手順

エージェントスペースを作成した後:

- ソースコードリポジトリ (GitHub、GitLab、Bitbucket、または GitHub Enterprise Server) を接続して、コードレビューと侵入テストのコンテキストを確認する
- 設計レビューとペネトレーションテストでドキュメントコンテキストの Confluence を接続する
- 接続されたリポジトリのコードレビュー機能を有効にする (「」を参照the section called “GitHub リポジトリのプルリクエストコードレビューを有効にする”)
- ドメイン検証を含む侵入テスト機能を設定する
- (IAM アイデンティティセンターを使用している場合) エージェントスペースページのウェブアプリセクションで、このエージェントスペースにユーザーを割り当てます (「」を参照the section called “ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する”)。
- (IAM 専用アクセスを使用している場合) コンソールにアクセスできるユーザーは、このエージェントスペースの管理者アクセスリンクからウェブアプリケーションを起動できます。

Connect 統合

エージェントスペースがコードレビュー、侵入テスト、脅威モデリングに使用するソースコードプロバイダー (GitHub、GitLab、Bitbucket、GitHub Enterprise Server) とドキュメントプロバイダー (Confluence) をプライベートネットワーク接続とともに接続および管理します。

エージェントスペースとの統合の仕組み

AWS セキュリティエージェントは、サードパーティーのソースコードおよびドキュメントプロバイダーに接続して、エージェントがセキュリティ評価中にコードとドキュメントを分析できるようにします。特定のプロバイダーを接続する前に、統合がエージェントスペースと機能にどのように関連しているかを理解するのに役立ちます。

一度登録すると、どこでも再利用できます

プロバイダーを AWS セキュリティエージェントに接続するには、異なるレベルで発生する 2 つの異なるステップが必要です。

1. 統合 (アカウントレベル) を登録します。AWS セキュリティエージェントマネジメントコンソールの統合ページからプロバイダーを 1 回登録します。登録は、GitHub 組織、GitLab グループ、Bitbucket ワークスペース、Confluence サイトなどのプロバイダーアカウントへの承認された接続を表します。登録はアカウントレベルのリソースです。
2. リソースをエージェントスペース (エージェントスペースレベル) に接続します。エージェントスペース内から、ソースコードプロバイダーのリポジトリまたは Confluence のページなど、登録さ

れた統合からリソースを接続し、エージェントがリソースを使用する方法を設定します。このステップは、各エージェントスペースに固有です。

1 つの登録は、アカウント内のすべてのエージェントスペースで再利用されます。1 つのエージェントスペースの設定中にプロバイダーを登録した場合、リソースを別のエージェントスペースに接続するときに同じ登録を使用できます。同じプロバイダーを再度登録することはありません。

機能間で共有される 1 つの接続

リソースをエージェントスペースに接続すると、そのリソースはエージェントの機能間で共有されます。機能ごとにリポジトリを個別に接続することはありません。

- 読み取りアクセスは、 リソースを接続することで付与されます。リポジトリを接続すると、AWS セキュリティエージェントはそれをフルコードスキャン (コードレビュー) 、ペネトレーションテストコンテキスト、脅威モデリング、設計レビューのために読み取ることができます。Confluence ページを接続すると、エージェントはそれを設計レビュー、脅威モデリング、侵入テストのドキュメントコンテキストとして読み取ることができます。
- 書き込みアクションは、リポジトリごとにオプトインされます。ソースコードリポジトリをエージェントスペースに接続すると、リポジトリごとに書き込みアクションをさらに有効にできます。
  - コードレビューコメント - AWS セキュリティエージェントは、レビュー結果をプルリクエスト (または GitLab でのマージリクエスト) のコメントとして投稿します。
  - コード修復 - AWS セキュリティエージェントは、検出された脆弱性を修正してプルリクエスト (またはマージリクエスト) を開きます。

    これらの書き込みアクションは、パブリックリポジトリでは使用できません。読み取り専用分析は引き続き適用されます。

> **Note**
>
> Confluence は、ソースコードプロバイダーではなくドキュメントプロバイダーです。AWS セキュリティエージェントは、接続された Confluence ページを読み取り、設計レビュー、脅威モデリング、侵入テストのコンテキストを提供します。ページを選択すると、読み取りアクセスが許可されます。ページごとの機能切り替えはありません。

サポートされているプロバイダー

AWS セキュリティエージェントは、次のプロバイダーをサポートしています。

- ソースコード - GitHub (クラウドホスト GitHub およびクラウドホスト GitHub Enterprise)、GitHub Enterprise Server (セルフホスト)、GitLab (クラウドホスト)、GitLab Self-Managed (セルフホスト)、Bitbucket Cloud。
- ドキュメント - Confluence Cloud。

パブリックインターネット経由で到達できないセルフホストプロバイダーの場合、プライベート接続を介してエージェントのトラフィックをルーティングできます。詳細については、「the section called “プライベートにホストされたソースコントロールに接続する”」を参照してください。

次の手順

- 統合ページからプロバイダーを登録します。など、プロバイダーの接続トピックを参照してくださいthe section called “AWS セキュリティエージェントを GitHub リポジトリに接続する”。
- リソースをエージェントスペースに接続し、機能を設定します。「the section called “コードレビューを有効にする”」および「the section called “ペネトレーションテストを有効にする”」を参照してください。

AWS セキュリティエージェントを GitHub リポジトリに接続する

AWS セキュリティエージェントを GitHub リポジトリに接続して、コードレビュー、脅威モデリング、侵入テスト、自動修復機能を有効にします。AWS セキュリティエージェントは、クラウドホスト GitHub とクラウドホスト GitHub Enterprise の両方をサポートしています。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用され、機能間でどのように共有されるかを理解してください。

GitHub 統合には複数の目的があります。

> **Note**
>
> このページでは、クラウドホスト型 GitHub (github.com) とクラウドホスト型 GitHub Enterprise について説明します。セルフホスト型 GitHub Enterprise Server については、「」を参照してくださいthe section called “AWS セキュリティエージェントを GitHub Enterprise Server に接続する”。

- コードレビュー - 各プルリクエストのコード変更を組織のセキュリティ要件に照らして自動的に分析し、オンデマンドのフルリポジトリスキャンを実行します。
- 脅威モデリング - ソースコード、データフロー、アーキテクチャを分析してアプリケーションの理解を提供します
- 侵入テストコンテキスト - ソースコードを分析して、侵入テストに関するアプリケーションの理解を提供します
- 自動修復 - セキュリティ評価中に検出された脆弱性の修正を含むプルリクエストを送信する

GitHub を AWS セキュリティエージェントに接続するには、GitHub 組織またはユーザーアカウントの AWS セキュリティエージェント GitHub アプリを承認してから、AWS コンソールに接続を登録する必要があります。

GitHub 統合の仕組み

プルリクエスト分析は GitHub 内で行われます。GitHub アプリを承認し、リポジトリを接続し、AWS マネジメントコンソールでコードレビューコメントを有効にすると、AWS セキュリティエージェントは新しいプルリクエスト (変更されたコードのみの差分スキャン) ごとに変更を自動的にスキャンし、結果をプルリクエストコメントとして投稿します。

リポジトリのコードベース全体をスキャンする完全なコードレビューは、GitHub ではなく AWS Security Agent ウェブアプリケーションで作成して実行します。

侵入テストと脅威モデリングは、AWS セキュリティエージェントウェブアプリケーション内で開始されます。ユーザーは、接続されたリポジトリを選択してアプリケーションコンテキストを提供し、ペネトレーションテストのターゲットドメインを指定します。自動修復を有効にすると、ユーザーは接続されたリポジトリへのプルリクエストを開いて、AWS セキュリティエージェントに検出結果の修正をリクエストできます。

前提条件

開始する前に、以下があることを確認してください。

- GitHub 組織の管理者アクセスまたは GitHub ユーザーアカウント所有者アクセス
- AWS マネジメントコンソールでエージェントスペースの統合を設定するアクセス許可
- コードレビュー、脅威モデリング、侵入テストのために接続するリポジトリを理解する

Important

GitHub アプリは、GitHub アカウントまたは GitHub 組織に一度だけインストールできます。同じ GitHub 組織を AWS セキュリティエージェントに接続する必要がある場合は、統合が最初に登録されたのと同じ AWS アカウントを使用する必要があります。

Note

GitHub エンタープライズ組織が IP 許可リストを有効にしている場合は、GitHub アプリで許可された IP アドレスを受け入れる必要があります。IP アドレスを許可リストに自動的に追加することもできます。詳細については、GitHub ドキュメントの「GitHub Apps によるアクセスの許可」および「許可された IP アドレスの有効化GitHub 」を参照してください。GitHub リソースへのアクセスには、次の IP アドレスが使用されます。

- 米国東部 (バージニア北部) (us-east-1)
  - `34.228.181.128`
  - `44.219.176.187`
  - `54.226.244.221`
- 米国西部 (オレゴン) (us-west-2)
  - `34.212.16.133`
  - `52.89.67.212`
  - `54.187.135.61`
- アジアパシフィック (ムンバイ) (ap-south-1)
  - `13.126.209.199`
  - `13.234.6.24`
  - `35.154.102.216`
- アジアパシフィック (シンガポール) (ap-southeast-1)
  - `18.139.13.125`
  - `47.130.240.215`
  - `54.179.238.173`
- アジアパシフィック (シドニー) (ap-southeast-2)
  - `13.237.95.197`

    - 13.238.84.102
    - 52.64.174.242
  - アジアパシフィック (東京) (ap-northeast-1)
    - 13.192.12.233
    - 35.74.181.230
    - 57.183.50.158
  - ヨーロッパ (フランクフルト) (eu-central-1)
    - 18.158.110.140
    - 52.57.96.160
    - 52.59.55.56
  - 欧州 (アイルランド) (eu-west-1)
    - 34.251.85.24
    - 52.30.157.157
    - 52.51.192.222
  - 南米 (サンパウロ) (sa-east-1)
    - 54.94.247.213
    - 54.207.222.14
    - 54.232.201.242

AWS Security Agent GitHub アプリの承認と登録

AWS セキュリティエージェント GitHub アプリが GitHub 組織またはユーザーアカウントにアクセスすることを許可し、AWS コンソールに接続を登録します。

> **Important**
>
> ブラウザを閉じたり、移動したりせずに、このプロセスのすべてのステップを完了します。登録プロセスが中断された場合は、GitHub アプリをアンインストールして最初からやり直す必要がある場合があります。

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。

3. GitHub を選択します。
4. [次へ] を選択します。
5. インストールと認可を選択します。

   認可を完了するには、GitHub にリダイレクトされます。接続する組織またはユーザーアカウントへの管理者アクセス権を持つアカウントで GitHub にログインしていることを確認します。
6. GitHub で、AWS セキュリティエージェント GitHub アプリをインストールするアカウントまたは組織を選択します。
7. AWS セキュリティエージェントがアクセスできるリポジトリを選択します。
   - すべてのリポジトリ - 組織またはユーザーアカウント内のすべての現在および将来のリポジトリへのアクセスを許可します。
   - リポジトリのみを選択する - ドロップダウンから特定のリポジトリを選択します。複数のリポジトリを一度に 1 つずつ選択できます。

   > **Note**
   >
   > GitHub 組織またはユーザーアカウント設定の GitHub アプリ設定にアクセスして、リポジトリへのアクセスをいつでも変更できます。
8. インストールと認可を選択します。
9. AWS マネジメントコンソールにリダイレクトされて、登録を完了します。
10. 登録の詳細セクションで、次のフィールドを設定します。
    a. 登録名 - この GitHub 接続のわかりやすい名前を入力します。Acme-Corp-Org」や「Production-Repos」など、GitHub 組織またはユーザーアカウントを識別する名前を使用します。
    b. アカウントタイプ - ドロップダウンから次のいずれかを選択します。
       - Organization - GitHub 組織アカウントを接続した場合
       - ユーザー - 個人用 GitHub ユーザーアカウントを接続した場合
    c. Organization name (Organization を選択した場合のみ表示されます) - GitHub に表示される GitHub 組織の正確な名前を入力します。
11. 接続 を選択します。
12. 確認メッセージが表示され、統合ページに戻り、新しい GitHub 接続が登録名とともに表示されます。追加の GitHub 組織またはユーザーアカウントを接続するには、統合を再度追加を選択してこのプロセスを繰り返します。

GitHub 統合のトラブルシューティング

GitHub 統合プロセス中に問題が発生した場合は、次のガイダンスを使用して一般的な問題を解決します。

登録を完了できません

登録プロセスを完了できなかった場合 (ブラウザが閉じられた、登録ページから移動した、セッションが中断された場合など) 、AWS セキュリティエージェント GitHub アプリが GitHub 組織にインストールされているが、AWS コンソールに登録されていない可能性があります。

症状:

- GitHub アプリを再度認可しようとすると、GitHub には「インストール」ではなく「設定」が表示されます。
- AWS コンソールで登録を完了できない
- 統合が統合リストに表示されない

解決策:

1. GitHub 組織またはユーザーアカウントから AWS セキュリティエージェント GitHub アプリをアンインストールします。
2. AWS セキュリティエージェントコンソールに戻り、最初から統合プロセスを再開します。

同じ GitHub 組織を統合しようとする複数の AWS アカウント

GitHub アプリは、GitHub アカウントまたは GitHub 組織に一度だけインストールできます。別の AWS セキュリティエージェントアカウントと既に統合されている GitHub 組織のリポジトリを使用する必要がある場合は、統合が最初に登録された AWS アカウントを使用する必要があります。

解決策:

- GitHub 統合が登録されている AWS セキュリティエージェントアカウントを特定する
- その AWS アカウントを使用してエージェントスペースを作成し、リポジトリを接続する
- 統合を別の AWS アカウントに移動する必要がある場合は、まず元の AWS アカウントから GitHub アプリをアンインストールしてから、新しいアカウントと統合します。

次の手順

GitHub を AWS セキュリティエージェントに接続した後:

- これらのリポジトリを使用するエージェントスペースに移動します。
- コードレビューを有効にするまたはペネトレーションテストを設定を選択して、特定のリポジトリをエージェントスペースに接続し、その使用状況を設定します。
- 自動修復を有効にして、AWS セキュリティエージェントが脆弱性修正を含むプルリクエストを送信できるようにします。
- GitHub 組織設定で GitHub アプリのアクセス許可とリポジトリアクセスを確認する

GitHub 統合を削除する

特定の GitHub 組織またはユーザーアカウントからリポジトリにアクセスするために AWS セキュリティエージェントが不要になった場合は、GitHub 統合を削除します。AWS コンソールで統合を削除する前に、まず GitHub から AWS セキュリティエージェント GitHub アプリをアンインストールする必要があります。

削除の前提条件

GitHub 統合を削除する前に、以下を確認してください。

- この統合からリポジトリが接続されているエージェントスペースを確認した
- 影響を理解する: この統合を削除すると、この統合のリポジトリを使用して、すべてのエージェントスペースでコードレビュー、ペネトレーションテストコンテキスト、ペネトレーションテスト修復のためのリポジトリ接続が切断されます。
- GitHub アプリをアンインストールするための GitHub 組織の管理者アクセスまたはユーザーアカウント所有者アクセス

ステップ 1: GitHub から AWS セキュリティエージェント GitHub アプリをアンインストールする

まず、GitHub 組織またはユーザーアカウントから AWS セキュリティエージェント GitHub アプリをアンインストールします。

GitHub Organizations の場合:

1. github.com に移動し、組織ページを開きます。

2. 左側のサイドバーで、設定 を選択します。
3. Code、Planning、および Automation で、Installed GitHub Apps を選択します。
4. リストから AWS セキュリティエージェントアプリを見つけます。
5. AWS セキュリティエージェントアプリの横にある設定 を選択します。
6. 設定ページの下部までスクロールし、アンインストールを選択します。
7. プロンプトが表示されたら、アンインストールを確認します。

GitHub ユーザーアカウントの場合:

1. github.com に移動します。
2. プロファイルの写真を選択します。
3. [設定] を選択します。
4. 左側のサイドバーで、アプリケーションを選択します。
5. Installed GitHub Apps タブを開きます。
6. リストから AWS セキュリティエージェントアプリを見つけます。
7. AWS セキュリティエージェントアプリの横にある設定 を選択します。
8. 設定ページの下部までスクロールし、アンインストールを選択します。
9. プロンプトが表示されたら、アンインストールを確認します。

ステップ 2: AWS セキュリティエージェントから統合を削除する

GitHub アプリをアンインストールしたら、AWS コンソールから統合登録を削除します。

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. 統合リストで削除する GitHub 統合を見つけます。
3. 統合をクリックして選択します。
4. [を削除] を選択します。
5. 影響について警告する確認ダイアログを確認します。

> **Warning**
>
> この統合を削除すると、この GitHub 組織またはユーザーアカウントから接続されたリポジトリを持つすべてのエージェントスペースに影響します。これにより、以下が中断されます。

- 接続されたリポジトリのコードレビュー機能
- 接続されたリポジトリからの侵入テストコンテキスト
- 接続されたリポジトリの侵入テスト修復機能

続行する前に、GitHub から AWS セキュリティエージェント GitHub アプリがアンインストールされていることを確認してください。

6. GitHub から GitHub アプリをアンインストールしていない場合は、警告が表示されます。ステップ 1 に戻り、最初にアンインストールを完了します。
7. GitHub アプリをアンインストールして影響を理解した場合は、削除の確認を選択します。
8. 統合は統合リストから削除されます。この統合のリポジトリを持つエージェントスペースは、コードレビュー、侵入テストコンテキスト、または自動修復のためにそれらのリポジトリにアクセスできなくなりました。

AWS セキュリティエージェントを GitLab リポジトリに接続する

AWS セキュリティエージェントを GitLab クラウドリポジトリに接続して、コードレビュー、脅威モデリング、侵入テスト、自動修復機能を有効にします。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用され、機能間でどのように共有されるかを理解してください。

GitLab 統合には複数の目的があります。

- コードレビュー - 各マージリクエストのコード変更を組織のセキュリティ要件に照らして自動的に分析し、オンデマンドのフルリポジトリスキャンを実行します。
- 脅威モデリング - ソースコード、データフロー、アーキテクチャを分析してアプリケーションの理解を提供します
- 侵入テストコンテキスト - ソースコードを分析して、侵入テストに関するアプリケーションの理解を提供します。
- 自動修復 - セキュリティ評価中に検出された脆弱性の修正を含むマージリクエストを送信する

GitLab を AWS セキュリティエージェントに接続するには、適切なアクセス許可を持つ GitLab アクセストークンを提供し、接続を AWS コンソールに登録する必要があります。

GitLab 統合の仕組み

マージリクエストの分析は GitLab 内で行われます。アクセストークンを提供し、プロジェクトを接続し、AWS マネジメントコンソールでコードレビューコメントを有効にすると、AWS セキュリティエージェントは新しいマージリクエスト (変更されたコードのみの差分スキャン) ごとに変更を自動的にスキャンし、結果をマージリクエストコメントとして投稿します。

GitLab ではなく AWS Security Agent ウェブアプリケーションで、プロジェクト全体のコードベースをスキャンする完全なコードレビューを作成して実行します。

侵入テストと脅威モデリングは、AWS セキュリティエージェントウェブアプリケーション内で開始されます。ユーザーはターゲットドメインを指定し、接続されたリポジトリを選択してアプリケーションコンテキストを提供します。自動修復を有効にすると、ユーザーは接続されたリポジトリへのマージリクエストを開いて、AWS セキュリティエージェントに検出結果の修正をリクエストできます。

> **Note**
>
> 脆弱性が修正される前に開示されないように、パブリック GitLab リポジトリでは自動修復を使用できません。

前提条件

開始する前に、以下があることを確認してください。

- 接続するプロジェクトへのメインテイナーまたは所有者アクセス権を持つ GitLab.com アカウント
- 接続タイプに必要なスコープを持つ GitLab アクセストークン:
  - Personal - すべての読み取りアクセス許可と `api` アクセス許可を持つ個人用アクセストークン。
  - グループ - `read_api`および `read_repository`スコープを持つグループアクセストークン。
- AWS Security Agent Management Console で統合を設定するアクセス許可

> **Important**
>
> トークンの有効期限を現在の日付から最大 365 日に設定します。

> **Note**
>
> GitLab Personal Access Tokens は、複数の AWS アカウントで使用できます。GitHub と Atlassian の統合とは異なり、同じ GitLab アカウントを複数の AWS セキュリティエージェントインスタンスに接続することに制限はありません。

GitLab 接続を登録する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。
3. GitLab を選択し、次へを選択します。
4. アカウントタイプの選択で、次のいずれかを選択します。
   - Personal - 個々の GitLab ユーザーアカウントを接続します。
   - Group - 複数のプロジェクトを含む GitLab グループを接続します。グループ ID を入力します。
5. アクセストークンフィールドに、GitLab アクセストークンを貼り付けます。
6. 登録名フィールドに、 など、この接続のわかりやすい名前を入力しますEngineering-Team-GitLab。有効な文字は、文字、数字、ピリオド、アンダースコア、ハイフンです。
7. 接続 を選択します。

   統合ページに戻り、新しい接続が登録名とともに表示されます。

GitLab 統合のトラブルシューティング

GitLab 統合プロセス中に問題が発生した場合は、次のガイダンスを使用して一般的な問題を解決します。

無効または期限切れのトークン

症状

- 統合が接続に失敗する
- 以前は機能していた統合が機能しなくなった
- 認証の失敗を示すエラーメッセージ

解像度

1. GitLab で、ユーザー設定に移動し、アクセストークンを選択します。
2. トークンの有効期限が切れていないことを確認します。
3. トークンのタイプに必要なスコープがあることを確認します。
4. トークンの有効期限が切れている場合は、新しいトークンを作成し、AWS コンソールで統合を更新します。

レート制限

症状

- コードレビュー中の断続的な障害
- 遅延マージリクエスト分析

解像度

- GitLab は API リクエストにレート制限を適用します。多数のリポジトリや頻繁なマージリクエストがある場合、一部のリクエストはスロットリングされる可能性があります。
- レート制限ウィンドウがリセットされるまで待つか、GitLab サポートに連絡してレート制限を引き上げてください。

次の手順

GitLab を AWS セキュリティエージェントに接続した後:

- これらのリポジトリを使用するエージェントスペースに移動します。
- コードレビューを有効にするまたはペネトレーションテストを設定を選択して、特定のプロジェクトをエージェントスペースに接続し、その使用状況を設定します。
- コード修復を有効にして、AWS セキュリティエージェントが脆弱性修正を含むマージリクエストを送信できるようにします
- プライベートにホストされる GitLab インスタンスについては、「」を参照してください。 the section called “AWS セキュリティエージェントを GitLab セルフマネージドに接続する”

AWS セキュリティエージェントを GitLab セルフマネージドに接続する

AWS セキュリティエージェントを GitLab セルフマネージドインスタンスに接続して、独自のインフラストラクチャでホストされているリポジトリのコードレビュー、脅威モデリング、侵入テスト、自動修復機能を有効にします。

GitLab 自己管理型統合は GitLab クラウド (「」を参照the section called “AWS セキュリティエージェントを GitLab リポジトリに接続する”) と同じように機能し、プライベートインスタンスへのネットワーク接続のための追加設定を行います。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用され、機能間でどのように共有されるかを理解してください。

> **Note**
>
> GitLab 自己管理型は、別の統合タイプではなく、GitLab 統合を通じて登録されます。登録フローで、GitLab セルフホストエンドポイントの使用を選択し、インスタンス URL を指定します。クラウドでホストされる GitLab フローについては、「」を参照してくださいthe section called “AWS セキュリティエージェントを GitLab リポジトリに接続する”。

前提条件

開始する前に、以下があることを確認してください。

- 次のいずれかの GitLab 自己管理型インスタンス。
  - インターネット経由でパブリックにアクセス可能、または
  - プライベート接続経由でアクセス可能 (「」を参照the section called “プライベートにホストされたソースコントロールに接続する”)
- 接続タイプに必要なスコープを持つ GitLab アクセストークン:
  - Personal - すべての読み取りアクセス許可と `api` アクセス許可を持つ個人用アクセストークン。
  - グループ - `read_api`および `read_repository`スコープを持つグループアクセストークン。
- 接続するプロジェクトへの管理者または所有者のアクセス
- GitLab インスタンスは、最小 TLS バージョン 1.2 の HTTPS トラフィックを提供する必要があります

> **Note**
>
> GitLab 自己管理型インスタンスがプライベート認証機関によって発行された TLS 証明書を使用している場合は、プライベート接続の作成時に証明書の PEM エンコードされたパブリックキーを指定できます。これにより、AWS セキュリティエージェントはインスタンスへの TLS 接続を信頼できます。

GitLab 自己管理型接続を登録する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。
3. GitLab を選択し、次へを選択します。
4. アカウントタイプの選択で、個人またはグループを選択します。Group を選択した場合は、グループ ID を入力します。
5. GitLab セルフホスト型エンドポイントを使用するを選択します。
6. GitLab セルフホストエンドポイント URL フィールドに、 などのインスタンスの URL を入力しますhttps://gitlab.example.com。
7. インスタンスがパブリックにアクセスできない場合は、プライベート接続を使用してエンドポイントに接続を選択し、既存のプライベート接続を選択するか、新しいプライベート接続を作成します。「the section called “プライベートにホストされたソースコントロールに接続する”」を参照してください。
8. アクセストークンフィールドに、GitLab アクセストークンを貼り付けます。
9. 登録名 フィールドに、この接続のわかりやすい名前を入力します。有効な文字は、文字、数字、ピリオド、アンダースコア、ハイフンです。
10. 接続 を選択します。

    統合ページに戻り、新しい接続が登録名とともに表示されます。

プライベート接続

GitLab セルフマネージドインスタンスがパブリックにアクセスできない場合は、統合を登録する前にプライベート接続を作成する必要があります。詳細な手順the section called “プライベートにホストされたソースコントロールに接続する”については、「」を参照してください。

> **Important**
>
> サービスマネージドプライベート接続では、エージェントスペースが作成されたのと同じ AWS アカウントで GitLab セルフマネージドインスタンスを実行する必要があります。クロスアカウントアクセスの場合は、独自の VPC Lattice リソース設定を提供するセルフマネージドプライベート接続を使用します。

GitLab 自己管理型統合のトラブルシューティング

のトラブルシューティング手順に加えてthe section called “AWS セキュリティエージェントを GitLab リポジトリに接続する”、以下の問題はセルフマネージドインスタンスに固有のものです。

インスタンスに到達できません

症状

- 接続がタイムアウトまたはネットワークエラーで失敗する
- 統合は以前は機能していましたが、機能しなくなりました

解像度

- GitLab インスタンスが実行されていてアクセス可能であることを確認します。
- プライベート接続を使用している場合は、VPC Lattice リソースゲートウェイが正常であり、ENIs がインスタンスにネットワーク接続されていることを確認します。
- セキュリティグループが設定されたポートでトラフィックを許可していることを確認する
- TLS 証明書が有効で有効期限が切れていないことを確認する

TLS 証明書エラー

症状

- 接続が SSL/TLS エラーで失敗する

解像度

- インスタンスが TLS 1.2 以降で HTTPS を提供していることを確認する

- プライベート認証機関を使用する場合は、プライベート接続のセットアップ中に PEM エンコードされたパブリックキーが提供されていることを確認します。
- 証明書の有効期限が切れていないことを確認する

次の手順

GitLab 自己管理型を AWS セキュリティエージェントに接続した後:

- これらのリポジトリを使用するエージェントスペースに移動します。
- 特定のプロジェクトを接続するには、コードレビューを有効にするまたはペネトレーションテストを設定するを選択します。
- マージリクエストベースの修正のコード修復を有効にする

GitLab 統合を削除する

特定の GitLab アカウントまたはグループからリポジトリにアクセスするために AWS セキュリティエージェントが不要になった場合は、GitLab 統合を削除します。

削除の前提条件

GitLab 統合を削除する前に、以下を確認してください。

- この統合からリポジトリが接続されているエージェントスペースを確認した
- 影響を理解する: この統合を削除すると、コードレビュー、侵入テストコンテキスト、脅威モデリング、およびこの統合のリポジトリを使用したすべてのエージェントスペースにわたる自動修復のためのリポジトリ接続が切断されます。

AWS セキュリティエージェントから統合を削除する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. 削除する GitLab 統合を見つけます。
3. 統合をクリックして選択します。
4. [を削除] を選択します。
5. 確認ダイアログを確認し、削除の確認を選択します。

> **Note**
>
> 統合を削除した後、それ以上のアクセスを防ぐために、GitLab で個人用アクセストークンを取り消すこともできます。GitLab ユーザー設定に移動し、トークンを削除または取り消します。

GitLab 自己管理型統合にも同じプロセスが適用されます。

AWS セキュリティエージェントを GitHub Enterprise Server に接続する

AWS セキュリティエージェントを GitHub Enterprise Server (GHES) インスタンスに接続して、独自のインフラストラクチャでホストされているリポジトリのコードレビュー、脅威モデリング、侵入テスト、自動修復機能を有効にします。

GitHub Enterprise Server 統合は、クラウドホスト GitHub と同じ機能 (「AWS セキュリティエージェントを GitHub リポジトリに接続する」を参照) と、セルフホストインスタンスへのネットワーク接続のための追加設定を提供します。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用され、機能間でどのように共有されるかを理解してください。

> **Note**
>
> GitHub Enterprise Server は、別の統合タイプではなく、GitHub 統合を通じて登録されます。登録フローでは、インスタンスタイプとして GitHub Enterprise Server を選択します。クラウドでホストされる GitHub.com フローについては、「」を参照してくださいthe section called “AWS セキュリティエージェントを GitHub リポジトリに接続する”。

GitHub Enterprise Server 統合の仕組み

プルリクエスト分析は GitHub Enterprise Server 内で行われます。接続を許可し、リポジトリを接続し、AWS マネジメントコンソールでコードレビューコメントを有効にすると、AWS セキュリティエージェントは新しいプルリクエスト (変更されたコードのみの差分スキャン) ごとに変更を自動的にスキャンし、結果をプルリクエストコメントとして投稿します。

リポジトリのコードベース全体をスキャンする完全なコードレビューは、GitHub Enterprise Server ではなく AWS Security Agent ウェブアプリケーションで作成して実行します。

侵入テストと脅威モデリングは、AWS セキュリティエージェントウェブアプリケーション内で開始されます。ユーザーはターゲットドメインを指定し、接続されたリポジトリを選択してアプリケーションコンテキストを提供します。自動修復を有効にすると、ユーザーは接続されたリポジトリへのプルリクエストを開いて、AWS セキュリティエージェントに検出結果の修正をリクエストできます。

前提条件

開始する前に、以下があることを確認してください。

- 次のいずれかの GitHub Enterprise Server インスタンス。
  - インターネット経由でパブリックにアクセス可能、または
  - プライベート接続経由でアクセス可能 (「」を参照the section called “プライベートにホストされたソースコントロールに接続する”)
- GHES インスタンスでのサイト管理者または組織の管理者アクセス
- GHES インスタンスは、最小 TLS バージョン 1.2 の HTTPS トラフィックを提供する必要があります
- AWS Security Agent Management Console で統合を設定するアクセス許可

> **Note**
>
> GitHub Enterprise Server の統合は、複数の AWS アカウントで使用できます。

GitHub Enterprise Server 接続を登録する

GitHub Enterprise Server 接続を登録するには、OAuth ベースの認可フローを使用します。

> **Important**
>
> ブラウザを閉じたり、移動したりせずに、このプロセスのすべてのステップを完了します。登録プロセスが中断された場合は、最初から再起動する必要がある場合があります。

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。

3. GitHub を選択し、次へを選択します。
4. インスタンスタイプで、GitHub Enterprise Server を選択します。
5. GitHub Enterprise Server URL フィールドに、 などのインスタンスの HTTPS URL を入力しますhttps://github.example.com。
6. インスタンスがパブリックにアクセスできない場合は、プライベート接続を使用してエンドポイントに接続を選択し、既存のプライベート接続を選択するか、新しいプライベート接続を作成します。「the section called “プライベートにホストされたソースコントロールに接続する”」を参照してください。
7. 登録の詳細セクションで、次のフィールドを設定します。
   a. 登録名 - この接続のわかりやすい名前を入力します。有効な文字は、文字、数字、ピリオド、アンダースコア、ハイフンです。
   b. GitHub アカウントタイプ - Organization または User を選択します。
   c. Organization name (Organization を選択した場合のみ表示されます) - GitHub Enterprise Server 組織の正確な名前を入力します。名前では、大文字と小文字が区別されます。
8. 接続 を選択します。

> **Note**
>
> AWS セキュリティエージェントは、コンソールからリダイレクトして、GitHub Enterprise Server インスタンスで認可を完了します。認可が完了したら、コンソールに戻り、新しい接続が統合ページに表示されます。

プライベート接続

GitHub Enterprise Server インスタンスがパブリックにアクセスできない場合は、統合を登録する前にプライベート接続を作成する必要があります。詳細な手順the section called “プライベートにホストされたソースコントロールに接続する”については、「」を参照してください。

> **Important**
>
> サービスマネージドプライベート接続では、エージェントスペースが作成されたのと同じ AWS アカウントで GHES インスタンスを実行する必要があります。クロスアカウントアクセスの場合は、セルフマネージドプライベート接続を使用します。

> **Note**
>
> GHES インスタンスがプライベート認証機関によって発行された TLS 証明書を使用している場合は、プライベート接続の作成時に証明書の PEM エンコードされたパブリックキーを指定します。

GitHub Enterprise Server 統合のトラブルシューティング

AWS セキュリティエージェントの GitHub Enterprise Server への接続で問題が発生した場合は、次のガイダンスに従って一般的な問題を診断して解決します。

OAuth リダイレクトの失敗

症状

- ブラウザリダイレクトが認可フロー中に失敗する
- GHES で を承認した後に表示されるエラーページ

解像度

- ブラウザから GHES インスタンスにアクセスできることを確認します。
- OAuth コールバック URL が正しく設定されていることを確認します。
- 統合プロセスを最初から再起動する

インスタンスに到達できません

症状

- 接続がタイムアウトまたはネットワークエラーで失敗する

解像度

- GHES インスタンスが実行中でアクセス可能であることを確認する
- プライベート接続を使用している場合は、VPC Lattice 接続を確認します。
- セキュリティグループが設定されたポートでトラフィックを許可していることを確認する
- TLS 証明書が有効であることを確認する (TLS 1.2 以上)

次の手順

GitHub Enterprise Server を AWS セキュリティエージェントに接続した後:

- これらのリポジトリを使用するエージェントスペースに移動します。
- 特定のリポジトリを接続するには、コードレビューを有効にするまたはペネトレーションテストを設定するを選択します。
- コード修復を有効にして、AWS セキュリティエージェントが脆弱性修正を含むプルリクエストを送信できるようにします

GitHub Enterprise Server 統合を削除する

AWS セキュリティエージェントが特定の GHES インスタンスからリポジトリにアクセスする必要がなくなった場合は、GitHub Enterprise Server 統合を削除します。

削除の前提条件

GHES 統合を削除する前に:

- この統合から接続されたリポジトリを持つエージェントスペースを確認する
- 影響を理解する: を削除すると、接続されているすべてのリポジトリのコードレビュー、侵入テストコンテキスト、脅威モデリング、自動修復が中断されます。

統合を削除する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. 削除する GitHub Enterprise Server 統合を見つけます。
3. 統合を選択します。
4. [を削除] を選択します。
5. 確認ダイアログを確認し、削除の確認を選択します。

> **Note**
>
> 削除後、GHES インスタンスの OAuth アプリケーションを取り消すこともできます。GHES 組織設定に移動し、承認されたアプリケーションを削除します。

Atlassian インストール ID を検索する

AWS コンソールで Bitbucket または Confluence 統合を登録する前に、Atlassian サイトにインストールされている AWS Security Agent Forge アプリからインストール ID を見つける必要があります。この ID を登録フローに貼り付けます。

Bitbucket の場合

1. Bitbucket で、Workspace の設定に移動します。
2. サイドバーからアプリの偽造を選択します。
3. インストールされているアプリのリストで、AWS セキュリティエージェントとの接続を見つけます。
4. クリップボードにコピーを選択して、インストール ID をコピーします。

Confluence の場合

1. Confluence で、Confluence 管理に移動します。
2. サイドバーからアプリを選択します。
3. インストールされているアプリのリストで、AWS セキュリティエージェントとの接続を見つけます。
4. クリップボードにコピーを選択して、インストール ID をコピーします。

このインストール ID は、AWS セキュリティエージェントマネジメントコンソールで登録を完了するときに必要になります。

AWS セキュリティエージェントを Bitbucket リポジトリに接続する

AWS セキュリティエージェントを Bitbucket Cloud リポジトリに接続して、コードレビュー、脅威モデリング、侵入テスト、自動修復機能を有効にします。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用され、機能間でどのように共有されるかを理解してください。

Bitbucket 統合には複数の目的があります。

- コードレビュー - 各プルリクエストのコード変更を組織のセキュリティ要件に照らして自動的に分析し、オンデマンドのフルリポジトリスキャンを実行します。

- 脅威モデリング - ソースコード、データフロー、アーキテクチャを分析してアプリケーションの理解を提供します
- 侵入テストコンテキスト - 侵入テストに関するアプリケーションの理解を提供します
- 自動修復 - セキュリティ評価中に検出された脆弱性を修正してプルリクエストを送信する

Bitbucket を AWS セキュリティエージェントに接続するには、Atlassian サイトに AWS Security Agent Forge アプリをインストールし、OAuth 認可フローを完了する必要があります。

> **Note**
>
> AWS セキュリティエージェントは Bitbucket Cloud のみをサポートします。Bitbucket サーバーと Bitbucket データセンターはサポートされていません。

Bitbucket 統合の仕組み

プルリクエスト分析は Bitbucket 内で行われます。Forge アプリをインストールし、リポジトリを接続し、AWS マネジメントコンソールでコードレビューコメントを有効にすると、AWS セキュリティエージェントは新しいプルリクエスト (変更されたコードのみの差分スキャン) ごとに変更を自動的にスキャンし、結果をプルリクエストコメントとして投稿します。

Bitbucket ではなく AWS セキュリティエージェントウェブアプリケーションで、リポジトリのコードベース全体をスキャンする完全なコードレビューを作成して実行します。

侵入テストと脅威モデリングは、AWS セキュリティエージェントウェブアプリケーション内で開始されます。ユーザーはターゲットドメインを指定し、接続されたリポジトリを選択してアプリケーションコンテキストを提供します。

> **Note**
>
> パブリック Bitbucket リポジトリが修正される前に脆弱性を開示しないように、自動修復は利用できません。

前提条件

開始する前に、以下があることを確認してください。

- 管理者アクセス権を持つ Bitbucket Cloud ワークスペース
- サイト管理者権限を持つ Atlassian アカウント
- AWS Security Agent Management Console で統合を設定するアクセス許可

> **Important**
>
> 1 つのアトラシアンサイトは、リージョンごとに 1 つの AWS アカウントにのみ関連付けることができます。同じリージョン内の別の AWS アカウントで同じ Bitbucket サイトを AWS セキュリティエージェントに接続する必要がある場合は、まず既存の統合を削除する必要があります。

> **Note**
>
> Atlassian Forge アプリの料金がこの統合に適用されます。詳細については、Atlassian ドキュメントの「Forge platform pricing」を参照してください。

Bitbucket 接続を登録する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。
3. Bitbucket を選択し、次へを選択します。
4. 画面の指示に従って、AWS Security Agent Forge アプリを Bitbucket ワークスペースにインストールします。インストール後、インストール ID をコピーします。「the section called “Atlassian インストール ID を検索する”」を参照してください。
5. Installation ID フィールドに、Bitbucket からコピーしたインストール ID を貼り付けます。
6. Bitbucket ワークスペースフィールドに、Bitbucket ワークスペースのスラグを入力します。例: `acme-corp`。
7. [承認] を選択します。

   Atlassian にリダイレクトされ、AWS セキュリティエージェントが Bitbucket ワークスペースにアクセスすることを許可します。認可が完了したら、コンソールに戻ります。
8. 登録の詳細セクションで、この接続の登録名を入力します。有効な文字は、文字、数字、ピリオド、アンダースコア、ハイフンです。

9. 接続 を選択します。

> **Note**
>
> 接続後のプロビジョニングの遅延
>
> Connect を選択すると、Atlassian 側でのプロビジョニングに数分かかる場合があります。この間、統合は保留中のステータスを報告します。プロビジョニングが完了する前にリポジトリを選択したり、コードレビューを有効にしたりすると、インストールがまだ利用できないというメッセージが表示されます。数分後にもう一度お試しください。

Bitbucket 統合のトラブルシューティング

Bitbucket 統合プロセス中に問題が発生した場合は、次のガイダンスを使用して一般的な問題を解決します。

登録を完了できません

登録プロセスが中断された場合 (ブラウザが閉じられ、セッションタイムアウト) 、Forge アプリは Atlassian サイトにインストールされているが、AWS コンソールに登録されていない可能性があります。

解像度

- AWS Security Agent コンソールに戻り、統合プロセスを再起動します。
- Forge アプリはインストールされたままであり、再インストールする必要はありません。

サイトが既に別の AWS アカウントに接続されている

症状

- Atlassian サイトが既に別のアカウントに関連付けられていることを示すエラー

解像度

- 1 つの Atlassian サイトは、リージョンごとに 1 つの AWS アカウントにのみ接続できます。
- 既存の統合がある AWS アカウントを特定し、そのアカウントを使用するか、最初に既存の統合を削除します。

インストールは使用できません

症状

- リポジトリを選択するか、コードレビューを有効にすると、Bitbucket のインストールのステータスが ではなく PENDING_INSTALLATIONであることを示すメッセージが表示されますAVAILABLE。

解像度

- インストールはまだ Atlassian 側でプロビジョニング中です。プロビジョニングは通常数分以内に完了します。数分待ってから、もう一度試してください。
- 数分経ってもステータスが利用できない場合は、統合を削除して再度登録します。再登録する前に、Bitbucket ワークスペースから Forge アプリをアンインストールします。手順については、「Bitbucket 統合の削除」を参照してください。以前の Forge アプリをアンインストールせずに再登録すると、インストールが保留状態で停止する可能性があります。

次の手順

Bitbucket を AWS セキュリティエージェントに接続した後:

- これらのリポジトリを使用するエージェントスペースに移動します。
- コードレビューを有効にするまたはペネトレーションテストを設定して、特定のリポジトリをエージェントスペースに接続するを選択します。
- コード修復を有効にして、AWS セキュリティエージェントが脆弱性修正を含むプルリクエストを送信できるようにします

Bitbucket 統合を削除する

特定の Bitbucket ワークスペースからリポジトリにアクセスするために AWS セキュリティエージェントが不要になった場合は、Bitbucket 統合を削除します。

削除の前提条件

Bitbucket 統合を削除する前に:

- この統合から接続されたリポジトリを持つエージェントスペースを確認する

- 影響を理解する: を削除すると、接続されているすべてのリポジトリのコードレビュー、侵入テストコンテキスト、脅威モデリング、自動修復が中断されます。

ステップ 1: AWS セキュリティエージェントから統合を削除する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. 削除する Bitbucket 統合を見つけます。
3. 統合を選択します。
4. [を削除] を選択します。
5. 確認ダイアログを確認し、削除の確認を選択します。

ステップ 2: Forge アプリをアンインストールする

AWS セキュリティエージェントから統合を削除したら、Atlassian サイトから Forge アプリをアンインストールします。

1. Bitbucket で、Workspace 設定に移動します。
2. Forge Apps を選択します。
3. AWS セキュリティエージェントで接続を見つけます。
4. アンインストール を選択します。

> **Important**
>
> Forge アプリは自動的にアンインストールされません
>
> AWS セキュリティエージェントコンソールで統合を削除しても、Forge アプリは Atlassian サイトからアンインストールされません。同じ Bitbucket ワークスペースを再度登録する場合は、まず Forge アプリをアンインストールする必要があります。そうしないと、新しいインストールが保留状態で停止する可能性があります。

AWS セキュリティエージェントを Confluence に接続する

AWS セキュリティエージェントを Confluence Cloud に接続して、セキュリティ評価のドキュメントコンテキストを提供します。コードプロバイダーとは異なり、Confluence は、脅威モデル、アー

キテクチャドキュメント、API 仕様、およびセキュリティレビューの品質を向上させるその他のマテリアルを提供するドキュメントソースとして機能します。開始する前に、the section called “エージェントスペースとの統合の仕組み”「」を参照して、登録がエージェントスペース間でどのように再利用されるかを理解してください。

Confluence 統合には複数の目的があります。

- 設計レビューコンテキスト - セキュリティ設計レビュー用のアーキテクチャドキュメントと設計仕様を提供します。
- 脅威モデリング - 脅威分析用の既存の脅威モデルとシステムドキュメントを提供します。
- 侵入テストコンテキスト - 侵入テスト中により深く理解するためのアプリケーションドキュメントを提供します

Confluence を AWS セキュリティエージェントに接続するには、Atlassian サイトに AWS Security Agent Forge アプリをインストールし、OAuth 認可フローを完了する必要があります。

> **Note**
>
> AWS セキュリティエージェントは Confluence Cloud のみをサポートします。Confluence データセンターと Confluence Server はサポートされていません。

Confluence 統合の仕組み

Confluence は、ソースコードプロバイダーではなくドキュメントプロバイダーです。Forge アプリをインストールし、AWS マネジメントコンソールでスペースまたはページを接続すると、AWS セキュリティエージェントは Confluence コンテンツにアクセスして、セキュリティ評価中にコンテキストを提供できます。

AWS Security Agent は、ページコンテンツを読み取り、アプリケーションアーキテクチャ、セキュリティ要件、設計上の決定を理解します。このコンテキストにより、設計レビュー、コードレビュー、侵入テスト中のセキュリティ検出結果の品質と関連性が向上します。

前提条件

開始する前に、以下があることを確認してください。

- 管理者アクセス権を持つ Confluence Cloud サイト

- サイト管理者権限を持つ Atlassian アカウント
- AWS Security Agent Management Console で統合を設定するアクセス許可

> **Important**
>
> 1 つのアトラシアンサイトは、リージョンごとに 1 つの AWS アカウントにのみ関連付けることができます。同じ Confluence サイトを同じリージョン内の別の AWS アカウントの AWS セキュリティエージェントに接続する必要がある場合は、まず既存の統合を削除する必要があります。

> **Note**
>
> Atlassian Forge アプリの料金がこの統合に適用されます。詳細については、Atlassian ドキュメントの「Forge platform pricing」を参照してください。

Confluence 接続を登録する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. [Add integration] (統合の追加) を選択します。
3. Confluence を選択し、次へを選択します。
4. 画面の指示に従って、AWS Security Agent Forge アプリを Atlassian サイトにインストールします。インストール後、インストール ID をコピーします。「the section called “Atlassian インストール ID を検索する”」を参照してください。
5. Confluence サイト URL フィールドに、 などのサイト URL を入力しますhttps://acme.atlassian.net。
6. Installation ID フィールドに、Confluence からコピーしたインストール ID を貼り付けます。
7. [承認] を選択します。

   Atlassian にリダイレクトされ、AWS セキュリティエージェントが Confluence サイトにアクセスすることを許可します。認可が完了したら、コンソールに戻ります。
8. 登録の詳細 セクションに、この接続の登録名を入力します。有効な文字は、文字、数字、ピリオド、アンダースコア、ハイフンです。
9. 接続 を選択します。

エージェントスペースのページを選択する

Confluence 統合を登録したら、特定のページをエージェントスペースに接続します。ページを選択すると、そのページのコンテンツへの読み取り (フェッチ) アクセスが AWS セキュリティエージェントに付与されます。ページごとの機能オプションはありません。エージェントは接続されたすべてのページを読み込みます。接続ウィザードのレビューステップでは、エージェントにアクセスさせたくないページを削除できます。

Confluence 統合のトラブルシューティング

Confluence 統合プロセス中に問題が発生した場合は、次のガイダンスを使用して一般的な問題を解決します。

登録を完了できません

登録プロセスが中断された場合、Forge アプリは Atlassian サイトにインストールされているが、AWS コンソールに登録されていない可能性があります。

解像度

- AWS セキュリティエージェントコンソールに戻り、統合プロセスを再起動します。
- Forge アプリはインストールされたままであり、再インストールする必要はありません。

Atlassian からアンインストールされた Forge アプリ

統合が AWS セキュリティエージェントにまだ存在する間に、AWS Security Agent Forge アプリが Atlassian サイトからアンインストールされた場合:

症状

- 統合は AWS コンソールに表示されますが、Confluence コンテンツにアクセスできません
- ページを一覧表示または読み込もうとしたときのエラー

解像度

- Atlassian Marketplace から Forge アプリを再インストールする
- 問題が解決しない場合は、AWS コンソールで統合を削除し、再登録します。

サイトが既に別の AWS アカウントに接続されている

解像度

- 1 つの Atlassian サイトは、リージョンごとに 1 つの AWS アカウントにのみ接続できます。
- 既存の統合がある AWS アカウントを特定し、そのアカウントを使用するか、最初に既存の統合を削除します。

次の手順

Confluence を AWS セキュリティエージェントに接続した後:

- このドキュメントを使用するエージェントスペースに移動します。
- 設計レビューと侵入テストのコンテキストとして含める特定のページを選択する
- 必要に応じて S3 経由で追加のドキュメントをアップロードする (「」を参照the section called “S3 バケットからエージェントリソースを提供する”)

Confluence 統合を削除する

AWS セキュリティエージェントが特定の Confluence サイトのドキュメントにアクセスする必要がなくなった場合は、Confluence 統合を削除します。

削除の前提条件

Confluence 統合を削除する前に:

- この統合からページが接続されているエージェントスペースを確認する
- 影響を理解する: を削除すると、この統合のコンテンツを使用して、すべてのエージェントスペースで設計レビュー、侵入テスト、脅威モデリングのドキュメントコンテキストが壊れます。

ステップ 1: AWS セキュリティエージェントから統合を削除する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. 削除する Confluence 統合を見つけます。
3. 統合を選択します。
4. [を削除] を選択します。

5. 確認ダイアログを確認し、削除の確認を選択します。

ステップ 2: Forge アプリをアンインストールする (オプション)

AWS セキュリティエージェントから統合を削除した後、オプションで Atlassian サイトから Forge アプリをアンインストールできます。

1. Confluence で、Confluence 管理に移動します。
2. アプリを選択します。
3. AWS セキュリティエージェントで接続を見つけます。
4. アンインストール を選択します。

プライベートにホストされたソースコントロールに接続する

> **Important**
>
> プライベート接続は AWS セキュリティエージェントのプレビューリリースであり、変更される可能性があります。

AWS セキュリティエージェントは、GitLab セルフマネージドインスタンスや GitHub Enterprise Server など、プライベートネットワークで実行されているソース管理システムに接続できます。プライベート接続は Amazon VPC Lattice を使用して、システムをパブリックインターネットに公開することなく、AWS セキュリティエージェントとプライベートインフラストラクチャ間の安全な接続を確立します。

プライベート接続の仕組み

プライベート接続は、AWS セキュリティエージェントと VPC 内のターゲットリソースとの間に安全なネットワークパスを作成します。AWS セキュリティエージェントは、Amazon VPC Lattice を使用してこの接続を確立します。VPC Lattice は、基盤となるネットワークインフラストラクチャを管理することなく、VPCs とアカウント間のサービス間のトラフィックを接続、保護、モニタリングするための AWS アプリケーションネットワーキングサービスです。

プライベート接続を作成する場合:

1. ターゲットサービスへのネットワーク接続を持つ VPC、サブネット、および (オプションで) セキュリティグループを指定します。

2. AWS セキュリティエージェントは、サービスマネージドリソースゲートウェイを作成し、指定したサブネットに Elastic Network Interface (ENIs) をプロビジョニングします。
3. エージェントはリソースゲートウェイを使用して、プライベートネットワークパス経由でターゲットサービスの IP アドレスまたは DNS 名にトラフィックをルーティングします。

サービスマネージドプライベート接続とセルフマネージドプライベート接続

AWS セキュリティエージェントは、プライベート接続に 2 つのモードをサポートしています。

サービスマネージド型 (シンプルなセットアップに推奨):

- AWS セキュリティエージェントは、VPC Lattice リソースゲートウェイを作成および管理します。
- ターゲットサービスは、エージェントスペースが作成されるのと同じ AWS アカウントで実行されている必要があります。
- 複数の AWS アカウントにまたがることはできません。

セルフマネージド型 (アドバンストまたはクロスアカウント設定の場合):

- 独自の VPC Lattice リソース設定を作成および管理します。
- 既存のリソース設定の Amazon リソースネーム (ARN) を指定します。
- クロスアカウントアクセスをサポートします (お客様はクロスアカウントネットワークを管理します)。

前提条件

プライベート接続を作成する前に、以下があることを確認します。

- アクティブなエージェントスペース
- 既知のプライベート IP アドレスまたは DNS 名でプライベートにアクセス可能なターゲットサービス (GitLab 自己管理型または GitHub Enterprise Server)
- ターゲットサービスは、最小 TLS バージョン 1.2 の HTTPS トラフィックを提供する必要があります
- VPC 内のサブネット (1～20 サブネット)。高可用性を実現するには、複数のアベイラビリティーゾーンのサブネットを選択することをお勧めします。

- (オプション) ENIs へのトラフィックを制御するセキュリティグループ (最大 5)

> **Note**
>
> VPC Lattice では、次のアベイラビリティーゾーンはサポートされていません: use1-az3、usw1-az2、apne1-az3、、apne2-az2、、euc1-az2euw1-az4cac1-az3、ilc1-az2。

> **Note**
>
> プライベート接続はアカウントレベルのリソースです。プライベート接続を作成したら、同じホストに到達する必要がある複数の統合とエージェントスペースで再利用できます。

> **Note**
>
> OAuth 認証を使用するプロバイダーのプライベート接続を選択すると、プライベート接続はプロバイダーエンドポイントとトークン交換エンドポイントの両方に適用されます。プライベート接続に、両方のエンドポイントにトラフィックをルーティングできるホストアドレスが設定されていることを確認します。

プライベート接続を作成する

1. AWS セキュリティエージェントマネジメントコンソールで、統合に移動します。
2. プライベート接続タブを選択します。
3. プライベート接続の作成 を選択します。
4. 接続の詳細に名前を入力します。名前には小文字、数字、ハイフンを含めることができ、先頭と末尾は文字または数字にする必要があります。
5. 接続の詳細セクションで、AWS セキュリティエージェントがターゲットサービスに接続する方法を選択します。
   - AWS セキュリティエージェントが接続を作成および管理できるようにするには、既存のリソース設定の使用を消去してから、リソースの場所、アクセスコントロール、およびサービスターゲットの詳細セクションを完了します。

- 既に作成した VPC Lattice リソース設定をルーティングするには、「既存のリソース設定を使用する」を選択し、「既存のリソース設定」セクションを完了します。サービスマネージドセクションは適用されません。

6. (サービスマネージド) リソースの場所:
   a. リソースがある VPC を選択します。
   b. 1 つ以上のサブネットを選択します。少なくとも 2 つのアベイラビリティーゾーンでサブネットを選択することをお勧めします。
   c. (オプション) IP アドレスタイプ (IPv4、IPv6、またはデュアルスタック) を選択します。
7. (サービスマネージド) アクセスコントロール:
   a. 接続されたリソースに関連付けられているセキュリティグループを選択します。
   b. (オプション) 詳細設定で、ポート範囲を入力します。たとえば、最大 11 個のカンマ区切り TCP ポートまたは範囲です`443, 8080-8090`。
8. (サービスマネージド) サービスターゲットの詳細:
   a. Host address フィールドに、ターゲットサービスの DNS ホスト名または IP アドレスを入力します。
   b. DNS 解決の場合は、パブリックに解決可能なホストアドレスには Public、VPC 内でのみ解決可能なアドレスには VPC (プライベート DNS) を選択します。
   c. (オプション) 証明書パブリックキーフィールドに、ターゲットが自己署名証明書またはプライベート認証機関を使用している場合は、PEM エンコードされたパブリックキーを入力します。これにより、AWS セキュリティエージェントは TLS 接続を信頼できます。
9. (セルフマネージド) 既存のリソース設定のリソース設定で、リストから VPC Lattice リソース設定を選択するか、リソース設定 ID (で始まる`rcfg-`) またはその ARN を入力します。このリストには、現在のリージョンのリソース設定が表示されます。
10. [接続を作成] を選択します。

    接続ステータスが「進行中の作成」に変わります。これには最大 10 分かかることがあります。完了すると、ステータスは Available に変わります。

統合でプライベート接続を使用する

GitLab 自己管理型または GitHub Enterprise Server 統合を登録するときは、プライベート接続を使用してエンドポイントに接続を選択し、プライベート接続リストから接続を選択します。AWS セキュリティエージェントは、その接続を介してトラフィックをプロバイダーにルーティングします。使用可能なステータスの接続のみがリストに表示されます。

登録時にプライベート接続の作成を選択することもできます。AWS セキュリティエージェントは、接続の作成中に登録ドラフトを保持し、登録フローに戻ります。

セキュリティ

- パブリックインターネットへの露出なし - すべてのトラフィックは AWS ネットワークに残ります。
- カスタマー管理のセキュリティグループ - インバウンドトラフィックルールとアウトバウンドトラフィックルールを管理します。
- 最小特権のサービスにリンクされたロール - AWS セキュリティエージェントは、 でタグ付けされたリソースにスコープされたサービスにリンクされたロールを使用しますAWSSecurityAgentManaged。

> **Note**
>
> 組織に VPC Lattice API アクションを制限するサービスコントロールポリシー (SCPs) がある場合、サービスマネージドリソースゲートウェイはサービスにリンクされたロールを介して作成されます。SCPsサービスにリンクされたロールに必要なアクションを許可していることを確認します。

プライベート接続を検証する

プライベート接続が使用可能ステータスになったら、接続を確認します。

- ターゲットサービスが実行されていて、予想されるポートで接続を受け入れていることを確認します。
- ENIs にアタッチされたセキュリティグループがターゲットポートでアウトバウンドトラフィックを許可していることを確認します。
- サービスのセキュリティグループが VPC CIDR 範囲内の VPC Lattice データプレーン IPs からのインバウンドトラフィックを許可していることを確認します。
- サブネットルートテーブルが ENI サブネットとサービス間のトラフィックを許可していることを確認します。

プライベート接続を削除する

1. AWS セキュリティエージェントコンソールで、統合に移動します。
2. プライベート接続タブを選択します。
3. 削除するプライベート接続を選択します。
4. [削除] を選択します。

> **Important**
>
> 統合で現在使用されているプライベート接続は削除できません。最初に統合を削除してから、プライベート接続を削除します。

次の手順

- the section called “AWS セキュリティエージェントを GitLab セルフマネージドに接続する” - プライベート GitLab インスタンスを接続する
- the section called “AWS セキュリティエージェントを GitHub Enterprise Server に接続する” - プライベート GitHub Enterprise Server を接続する

エージェントをプライベート VPC リソースに接続する

ペネトレーションテストを実行するアプリケーションがパブリックインターネットで利用できない場合は、AWS セキュリティエージェントに VPC 設定を提供する必要があります。AWS セキュリティエージェントは、VPC、サブネット、セキュリティグループを含むこの VPC 設定を使用してアプリケーションにアクセスします。

> **Note**
>
> プライベート VPC でエンドポイントをテストする場合、既知のプライベート IPs「VPC CIDR ブロック」を参照）。次の IPv4 および IPv6 範囲を使用できます。
>
> ```
> 10.0.0.0/8
> 172.16.0.0/12
> 192.168.0.0/16
> fd00::/8
> ```

> **Note**
>
> サブネットに接続すると、AWS セキュリティエージェントは侵入テスト用に設定されたサブネットに ENI (Elastic Network Interface) を作成します。この ENI には、関連付けられたパブリック IP アドレスがありません。つまり、パブリックサブネット内の VPC インターネットゲートウェイと通信することはできません。ペネトレーションテストでオープンインターネットアクセスが必要な場合は、代わりに VPC NAT Gateway に関連付けられたプライベートサブネットを使用してください。

AWS マネジメントコンソールから VPC への一般的なアクセス権を AWS セキュリティエージェントに付与します。Security Agent ウェブアプリで、ユーザーは侵入テストの特定の設定を選択します。

エージェントスペースに VPC を追加するには

1. エージェントスペースの概要ページに移動する
2. アクションを選択し、ペネトレーションテスト設定を編集する
3. VPC 見出しで、VPC、サブネット、セキュリティグループを指定します。

最大 5 つの VPCs を追加できます。

必要なエージェントスペースサービスロールのアクセス許可

VPC で侵入テストを実行するには、エージェントスペースサービスロールに次のアクセス許可が含まれている必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeNetworkInterfaceAttribute",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
```

```
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateNetworkInterfacePermission",
      "Resource": "arn:aws:ec2:{{region}}:{{accountId}}:network-interface/*",
      "Condition": {
        "ArnEquals": {
          "ec2:Subnet": [
            "arn:aws:ec2:{{region}}:{{accountId}}:subnet/[[subnetIds]]"
          ]
        }
      }
    }
  ]
}
```

Security Agent ウェブアプリで侵入テストの特定の VPC 設定を選択するには

1. 侵入テストの概要ページに移動する
2. VPC 設定を追加する必要があるペネトレーションテストを選択し、ペンテストの詳細の変更を選択します。
3. 次へ を選択して VPC リソースセクションに移動します。
4. VPC、サブネット、セキュリティグループを選択する
5. 次へを選択して最後のセクションに到達し、ペネトレーションテストを保存します

> **Note**
>
> クロスアカウント侵入テストは現在、AWS Resource Access Manager を使用して共有される VPC リソース (サブネットとセキュリティグループ) でサポートされています。認証情報に使用される Secrets Manager シークレットと Lambda 関数は、AWS セキュリティエージェントのセットアップと同じ AWS アカウントで設定する必要があります。

別の AWS アカウントの VPC リソースに対するペネトレーションテストの実行

AWS Resource Access Manager を使用して、アカウントと共有されている VPC リソースに対してペネトレーションテストを実行できます。両方のアカウントは同じ AWS Organization の一部である必要があります。

1. (オプション) AWS 組織の自動リソース共有を有効にする

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

2. VPC リソースを所有する AWS アカウントの認証情報を使用して、サブネットとセキュリティグループのリソースを侵入テスト所有者アカウントと共有します。

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

3. エージェントスペースの概要ページに移動する
4. 侵入テストを選択し、サービスロール名を見つける
5. IAM ロールが共有 VPC リソースへのアクセスを許可していることを確認します。
6. アクションを選択し、ペネトレーションテスト設定を編集する
7. VPC 見出しで、共有 VPC、サブネット、セキュリティグループを指定し、更新された設定を保存します。
8. AWS セキュリティエージェントウェブアプリの侵入テストの概要ページに移動します。
9. VPC 設定を追加する必要があるペネトレーションテストを選択し、ペネトテストの詳細の変更を選択します。
10. 共有 VPC リソースを使用するようにペネトレーションテストを更新する

機能を設定する

修復、S3 ソース、ターゲットドメインなど、エージェントスペースの侵入テスト、コードレビュー、脅威モデリングを有効にして設定します。

ペネトレーションテストを有効にする

アプリケーションで自律的なペネトレーションテストを実行するように AWS セキュリティエージェントを設定します。この設定により、AWS セキュリティエージェントは AWS リソースにアクセスし、ドメインの所有権を検証し、ウェブアプリケーションと APIs の悪用可能な脆弱性を特定する包括的なセキュリティテストを実行できます。

ペネトレーションテストを有効にすると、AWS セキュリティエージェントは手動テストの遅延なしにアプリケーションセキュリティを継続的に検証できます。必要な AWS 統合を設定することで、AWS セキュリティエージェントがパブリックアプリケーションとプライベートアプリケーションの両方をテストし、検出結果を CloudWatch に記録し、認証されたテスト用の認証情報にアクセスできることを確認します。

この手順では、ターゲットドメインを設定し、オプションでテスト用の VPC、CloudWatch ログ記録、認証情報ストレージ、Lambda 関数を設定し、追加のコンテキストを提供するために S3 統合を設定し、IAM ロールを介してサービスアクセスを設定します。

前提条件

開始する前に、以下があることを確認してください。

- IAM ロールを作成するための管理権限を持つ AWS アカウント
- ドメイン所有権の検証機能 (DNS または HTTP レコードの変更)
- テストするターゲットドメインまたはアプリケーション
- (オプション) プライベートアプリケーションをテストする場合の VPC 設定の詳細
- (オプション) AWS セキュリティエージェントに追加アーティファクトを提供する場合の S3 バケット

ステップ 1: ドメインを設定する

ウィザードの最初のステップで、テストするターゲットドメインと、AWS セキュリティエージェントが所有権を検証する方法を指定します。

1. 「ターゲットドメイン」セクションの「ドメイン」フィールドにドメインを入力します。
2. 検証方法を選択します。
   - DNS_TXT – ドメインの DNS 設定に TXT レコードを追加して、ドメインの所有権を検証します。

- HTTP_ROUTE – ドメイン上の特定の URL で検証ファイルをホストすることで、ドメインの所有権を検証します。
- PRIVATE_VPC - プライベート VPC 侵入テストにのみ使用できます。ドメインがプライベート CIDR 範囲内の IP に解決されることを確認します
- 詳細については、「the section called “ペネトレーションテスト用のアプリケーションドメインを有効にする”」を参照してください。

3. 別のドメインを追加を選択し、ドメインを追加します (合計 5 つまで) 。
4. 次へ を選択して、ドメインの検証に進みます。

ステップ 2: ドメインを検証する

ウィザードの 2 番目のステップで、設定した各ドメインの所有権を確認します。AWS セキュリティエージェントは、侵入テストを実行する前に、検証済みの所有権を必要とします。

1. ターゲットドメインテーブルにリストされているドメインを確認します。
2. ドメインごとにそれを選択し、選択した方法に基づいて検証をトリガーします。
   - Route 53 ドメイン (同じ AWS アカウント): ワンクリック検証を選択します。AWS セキュリティエージェントは DNS レコードを自動的に作成し、検証を完了します。
   - DNS TXT (他の DNS プロバイダー): 検証トークンをコピーし、DNS レジストラに TXT レコードを追加し、ドメインを選択して検証を選択します。
   - HTTP ルート: 検証トークンをウェブサーバーの必要なルートパスに配置し、ドメインを選択して検証を選択します。詳細については、「the section called “ペネトレーションテスト用のアプリケーションドメインを有効にする”」を参照してください。
3. 次へ を選択して、オプションの設定に進みます。

> **Note**
>
> 検証済みドメインのサブドメインは、DNS TXT または HTTP ルート検証を使用する場合、個別の検証を必要としません。プライベート VPC 検証の場合、ターゲットエンドポイントドメイン名は検証用に設定された完全なドメイン名と一致する必要があります。

> **Note**
>
> VPC 内のプライベートドメインの場合、ドメインの検証ステータスが UNREACHABLE であっても続行できます。AWS セキュリティエージェントは、各ペンテスト実行の開始時にプライベートエンドポイントのドメイン検証を試みます。

ステップ 3: (オプション) 追加機能を設定する

ウィザードの 3 番目のステップでは、オプションの AWS リソースを設定して、ペネトレーションテストの範囲を拡張できます。このステップのすべてのセクションはオプションであり、デフォルトで折りたたまれています。

(オプション) VPC 設定を構成する

VPC 内でホストされているプライベートターゲットドメインをテストする場合は、AWS セキュリティエージェントの VPC 設定を構成します。このセクションはオプションであり、デフォルトで折りたたまれています。

1. VPCsセクションを展開します。
2. VPC ドロップダウンで、プライベートターゲットドメインをホストする VPC を選択します。
3. サブネットドロップダウンで、AWS セキュリティエージェントが使用する VPC サブネットを選択します。

> **Tip**
>
> 高可用性を実現するには、複数のアベイラビリティーゾーンから複数のサブネットを選択します。サブネットにアウトバウンド接続用の NAT ゲートウェイが含まれていることを確認します。

4. セキュリティグループドロップダウンで、AWS セキュリティエージェントが使用する VPC セキュリティグループを選択します。

> **Important**
>
> セキュリティグループが AWS セキュリティエージェントがペネトレーションテストを実行できるようにアウトバウンド接続を許可していることを確認します。

(オプション) CloudWatch ログ記録を設定する

ペネトレーションテスト実行のログを保存するように CloudWatch を設定します。このセクションはオプションであり、デフォルトで折りたたまれています。

1. CloudWatch Logs セクションを展開します。
2. Log Groups ドロップダウンで、既存の CloudWatch ロググループを選択します。
3. ロググループを選択しない場合、AWS セキュリティエージェントは適切なアクセス許可`/aws/securityagent/<agent name>/<pentest id>`を持つ という名前のロググループを作成します。

> **Note**
>
> IAM ロールに、選択した CloudWatch ロググループに書き込むアクセス許可があることを確認します。

(オプション) テスト認証情報のシークレットを設定する

アプリケーションがテストに認証情報を必要とする場合、AWS セキュリティエージェントは AWS Secrets Manager からそれらを安全に取得できます。このセクションはオプションであり、デフォルトで折りたたまれています。

1. シークレットセクションを展開します。
2. アプリケーションの認証情報を含む AWS Secrets Manager シークレットを選択します。
3. ウェブアプリケーションで侵入テストを設定するときは、認証されたテストのためにこれらのシークレットを参照できます。

> **Important**
>
> 認証情報は暗号化され、AWS Secrets Manager に保存されます。IAM ロールに、テスト中にこれらの認証情報を使用するための Secrets Manager for AWS Security Agent へのアクセス許可があることを確認します。

(オプション) テスト認証情報用に Lambda 関数を設定する

テスト中にアプリケーションの認証情報を提供できる Lambda 関数を設定します。このセクションはオプションであり、デフォルトで折りたたまれています。

1. Lambda 関数セクションを展開します。
2. アプリケーションの認証情報を提供できる Lambda 関数を選択します。
3. AWS セキュリティエージェントは、ペネトレーションテスト中にこれらの関数を呼び出して、認証情報を動的に取得します。

> **Note**
>
> IAM ロールに、指定された Lambda 関数を呼び出すアクセス許可があることを確認します。Lambda 関数は、テスト中に AWS セキュリティエージェントが使用するのに必要な形式で認証情報を返す必要があります。

(オプション) S3 バケットを設定する

AWS セキュリティエージェントへの入力として提供するドキュメントまたはアーティファクトをアップロードする場合は、S3 バケットの詳細を指定します。このセクションはオプションであり、デフォルトで折りたたまれています。

1. S3 バケットセクションを展開します。
2. Bucket フィールドに、S3 バケット名を入力または検索します。

> **Note**
>
> GitHub リポジトリを後で接続したり、ウェブアプリケーションに直接ファイルをアップロードしたりすることもできます。提供される情報は、徹底的なカバレッジを確保し、誤検出を減らし、実用的な結果をもたらすことができます。

(オプション) サービスアクセスを設定する

AWS セキュリティエージェントでは、侵入テストのために AWS リソース (VPC、CloudWatch ロググループ、Secrets Manager、Lambda 関数など) にアクセスするための IAM ロールが必要です。このセクションはオプションであり、デフォルトで折りたたまれています。

1. サービスアクセスセクションを展開します。
2. デフォルトでは、AWS セキュリティエージェントは、ペネトレーションテストに必要なアクセス許可を持つデフォルトの IAM ロールを使用します。
3. IAM ロールをカスタマイズするには、次のいずれかのオプションを選択します。
   a. デフォルトロールの作成 – AWS セキュリティエージェントは、必要なアクセス許可を持つ新しい IAM ロールを自動的に作成します。
   b. 既存のサービスロールを使用する – ドロップダウンメニューから既存の IAM ロールを選択します。
4. 既存のロールを使用する場合:
   a. 既存のロールを選択するの下にあるドロップダウンメニューを選択します。
   b. リストから IAM ロールを選択する
   c. 更新アイコンを選択して、必要に応じてリストを更新します。

> **Note**
>
> デフォルトの IAM ロールには、VPC リソース、CloudWatch ロググループ、Secrets Manager、Lambda 関数、および侵入テストに必要なその他のサービスにアクセスするためのアクセス許可が含まれています。特定のセキュリティ要件がない限り、デフォルトの IAM ロールを使用することをお勧めします。

ステップ 4: ペネトレーションテストを保存して有効にする

必要な設定をすべて設定したら、AWS セキュリティエージェントエージェントのペネトレーションテストを有効にします。

1. すべての設定セクションを確認して、正確性を確認します。
2. [保存] を選択します。
3. AWS セキュリティエージェントは設定を検証し、必要な AWS リソースを作成します。

次の手順

ペネトレーションテストを有効にした後:

- AWS セキュリティエージェントのウェブアプリケーションで侵入テストスコープを設定する

- テスト結果の通知設定をセットアップする
- 発見されたペネトレーションテストの結果を確認して対応します。
- (オプション) 検出結果修復用の追加のリポジトリを設定する

ペネトレーションテストの実行と管理の詳細については、AWS Security Agent ウェブアプリケーションのドキュメントを参照してください。

GitHub リポジトリのプルリクエストコードレビューを有効にする

プルリクエストコードレビューがコードレビューセットアップウィザードの一部として設定されました。GitHub リポジトリをエージェントスペースに接続すると、個々のリポジトリのプルリクエストコメントを有効にして、AWS セキュリティエージェントがプルリクエストを自動的に分析し、セキュリティ検出結果を投稿するようにできます。

セットアップ手順の詳細については、「」を参照してくださいthe section called “コードレビューを有効にする”。

プルリクエストの結果が GitHub にどのように表示されるか、およびそれらに対応する方法については、「」を参照してくださいthe section called “プルリクエストでコードセキュリティの検出結果を確認する”。

コードレビューを有効にする

GitHub リポジトリまたは S3 バケットからソースコードを接続して、コードレビューを有効にするようにエージェントスペースを設定します。コードレビューは、ソースコードのセキュリティの脆弱性と組織のカスタムセキュリティ要件への準拠を分析します。

コードレビュー設定の設定は、エージェントスペース全体のオペレーションです。接続する統合と S3 バケットは、コードレビューやペネトレーションテストなど、機能間で共有されます。

セットアップが完了すると、ユーザーは AWS Security Agent ウェブアプリケーションでコードレビューを作成して実行し、リポジトリと S3 ソースをスキャンしてセキュリティの問題を確認できます。

> **Note**
>
> エージェントスペースにすでに GitHub リポジトリが接続されている場合 (ペネトレーションテストのセットアップなど)、コードレビューは既に有効になっています。この設定をス

キップし、ウェブアプリケーションに直接移動してコードレビューを作成できます。「the section called “コードレビューを作成する”」を参照してください。

前提条件

開始する前に、以下があることを確認してください。

- AWS マネジメントコンソールで作成されたエージェントスペース (「」を参照the section called “エージェントスペースを作成する”)
- 次のソースコード入力の少なくとも 1 つ。
  - AWS セキュリティエージェント GitHub アプリがインストールされている GitHub 組織またはユーザーアカウント (「」を参照the section called “AWS セキュリティエージェントを GitHub リポジトリに接続する”)
  - レビューするソースコードを含む S3 バケット
- エージェントスペースの統合を設定するアクセス許可
- (オプション) セキュリティ要件検証を使用する場合、セキュリティ要件パックで少なくとも 1 つのセキュリティ要件が有効になっている (「」を参照the section called “セキュリティ要件を管理する”)

コードレビュー設定ウィザードにアクセスする

エージェントスペースのコードレビュー設定に移動します。

1. AWS セキュリティエージェントコンソールで、エージェントスペースを選択します。
2. 次のいずれかの場所からコードレビューを有効にするを選択します。
   - コードレビューカード
   - コードレビュータブで、コードレビューを有効にするを選択します。

> **Tip**
>
> AWS セキュリティエージェントでコードレビューのフィードバックを自動的に処理する場合は、コード修復も有効にします。詳細については、「the section called “ユーザーが侵入テストとコードレビューの検出結果の修正を開始できるようにする”」を参照してください。

コードレビュー設定のセットアップウィザードが表示されます。

ステップ 1: 統合、リポジトリ、バケットを接続する

ウィザードの最初のステップで、ソースコード入力を接続し、コードレビュー設定を構成します。続行するには、少なくとも 1 つの GitHub リポジトリまたは S3 バケットを追加する必要があります。

> **Important**
>
> ここで設定された統合と S3 バケットは、エージェントスペース間で共有されます。変更は、コードレビュー機能とペネトレーションテスト機能の両方に適用されます。

GitHub リポジトリを接続する

承認された GitHub 組織またはユーザーアカウントから GitHub リポジトリを追加します。追加を選択すると、2 ステップの Connect GitHub ウィザードが開きます。まずリポジトリを選択し、次に AWS セキュリティエージェントがそれぞれに対して実行できるアクションを設定します。

1. 「接続された統合」セクションで、「追加」を選択します。
2. 確認するリポジトリを含む GitHub 登録を選択します。
3. GitHub リポジトリの接続ステップで、接続する各リポジトリのチェックボックスをオンにします。
4. 次へ を選択して、機能の管理に移動します。

> **Note**
>
> 接続されたリポジトリには、コードレビューとペネトレーションテスト中にコード分析のために読み取り専用でアクセスされます。次のステップで、レビューコメントの投稿や修復プルリクエストの開始などの書き込みアクションを設定します。

> **Note**
>
> GitHub 統合をまだ登録していない場合は、設定 を選択して統合ページに移動し、AWS セキュリティエージェント GitHub アプリを承認できます。詳細については、「the section

called “AWS セキュリティエージェントを GitHub リポジトリに接続する”」を参照してください。

GitHub リポジトリ機能を設定する

Connect GitHub ウィザードの機能の管理ステップで、各リポジトリで AWS セキュリティエージェントが実行できる操作を選択します。コードレビューコメントとコード修復は、リポジトリごとに個別に設定されます。

1. リポジトリごとに、 のコードレビューコメントを切り替えて、AWS セキュリティエージェントがプルリクエストのコメントとしてセキュリティ検出結果を投稿するようにします。
2. リポジトリごとに、コード修復をオンに切り替えて、AWS セキュリティエージェントが検出結果を修正できるようにします。有効にすると、ウェブアプリユーザーは結果を修正するプルリクエストをリクエストでき、プルリクエストの作成者はエージェントがレビューコメントに対応する`@AWS-Security-Agent fix all findings.`ようにコメントできます。
3. 保存を選択して選択を適用し、コードレビュー設定ウィザードに戻ります。

リポジトリに対してコードレビューコメントが有効になっている場合:

- AWS セキュリティエージェントは、プルリクエストが「レビュー準備完了」とマークされると、プルリクエストを自動的に分析します。ドラフトプルリクエストは分析されません。
- セキュリティ検出結果は、特定の修復ガイダンスを使用してプルリクエストに直接レビューコメントとして投稿されます。
- 分析では、設定したコードレビュー設定 (セキュリティの脆弱性、カスタム要件、またはその両方) を使用します。

> **Note**
>
> プルリクエストコメントは、プライベート GitHub リポジトリでのみ使用できます。

リポジトリでコード修復を有効にすると、ウェブアプリケーションユーザーはそのリポジトリでコードレビューとペネトレーションテストの結果の両方の修復を開始でき、AWS セキュリティエージェントは各修正をプルリクエストとして配信します。詳細については、「the section called “ユーザーが侵入テストとコードレビューの検出結果の修正を開始できるようにする”」を参照してください。

プルリクエストの検出結果が GitHub にどのように表示されるか、およびそれらに応答する方法の詳細については、「」を参照してくださいthe section called “プルリクエストでコードセキュリティの検出結果を確認する”。

S3 バケットを追加する

コードレビュー用のソースコードまたはコンテキストリソースを含む S3 バケットを追加します。

1. S3 バケットセクションで、S3 リソースの追加を選択します。
2. ソースコードを含むバケットまたはプレフィックスの S3 URI を入力します。
3. [Add] (追加) を選択します。

> **Note**
>
> 最大 10 個の S3 リソースを追加できます。S3 バケットは、コードレビューやペネトレーションテストなどの機能間で共有されます。

> **Tip**
>
> ソースコード、設定ファイル、infrastructure-as-code テンプレート、または AWS Security Agent でセキュリティの問題を分析するその他のアーティファクトを含む S3 バケットを追加できます。

コードレビュー設定を構成する

コードレビュー中に AWS Security Agent が分析するセキュリティ問題のタイプを設定します。この設定は、このエージェントスペースでコードレビューが有効になっているすべてのリポジトリとソースに適用されます。

1. コードレビュー設定セクションで、次のいずれかのオプションを選択します。
   - セキュリティ要件の検証 – コードがセキュリティ要件パックで有効にしたセキュリティ要件に準拠しているかどうかを検証します。
   - セキュリティ脆弱性の検出結果 – コード内の一般的なセキュリティ脆弱性を特定します。
   - セキュリティ要件と脆弱性の検出結果 – 有効にしたセキュリティ要件と一般的なセキュリティ脆弱性の両方への準拠についてコードを分析します。これはデフォルトの設定です。

> **Note**
>
> セキュリティ要件の検証が有効になっている場合、AWS セキュリティエージェントは、セキュリティ要件パックで有効にしたセキュリティ要件に対してコードをチェックします。セキュリティ要件の検証を選択しても、少なくとも 1 つのセキュリティ要件が有効になっていない場合、AWS セキュリティエージェントは要件ベースの検出結果を識別しません。セキュリティ要件の詳細については、「the section called “セキュリティ要件を管理する”」を参照してください。

1. 次へ を選択して、オプションの設定に進みます。

ステップ 2: オプションの設定

ウィザードの 2 番目のステップで、コードレビュー環境のオプションの CloudWatch ログ記録とサービスアクセスを設定します。

(オプション) CloudWatch ログを設定する

コードレビュー中にアプリケーションの動作をキャプチャして分析するように CloudWatch ロググループを設定します。

1. CloudWatch Logs セクションを展開します。
2. Log Groups ドロップダウンで、AWS アカウントから 1 つ以上の既存の CloudWatch ロググループを選択します。

> **Note**
>
> ロググループを選択しない場合、AWS セキュリティエージェントはコードレビュー実行ログを保存するデフォルトのロググループを自動的に作成します。

(オプション) サービスアクセスを設定する

コードレビューのために S3 バケットや CloudWatch ログなどの AWS リソースにアクセスするために AWS セキュリティエージェントが使用する IAM サービスロールを設定します。

1. サービスアクセスセクションを展開します。

2. 以下のオプションのいずれかを選択します。
   - デフォルトロールの作成 – AWS セキュリティエージェントは、コードレビューに必要なアクセス許可を持つ新しい IAM ロールを自動的に作成します。
   - 既存のサービスロールを使用する – ドロップダウンメニューから既存の IAM ロールを選択します。
3. デフォルトのロールを使用する場合は、サービスロール名を入力します。名前は、アカウント内のすべてのロールで一意である必要があり、英数字と+=,.@-_文字を使用し、スペースを含めることはできません。

> **Note**
>
> サービスロール名の最大長は 64 文字です。既存のロールを選択しない場合、サービスロールが自動的に作成されます。

1. 保存を選択してセットアップを完了します。

セットアップ後

コードレビュー設定ウィザードを完了したら、次の操作を行います。

- エージェントスペースページのコードレビューカードに準備完了ステータスが表示されます。
- ユーザーはウェブアプリケーションを起動し、コードレビューを作成して、接続されたリポジトリと S3 ソースをスキャンできます。
- コードレビュータブで設定の編集を選択すると、いつでも設定を変更できます。

コードレビュー設定の編集

初期設定後にコードレビュー設定を変更するには:

1. AWS セキュリティエージェントコンソールでエージェントスペースに移動します。
2. コードレビュータブを選択します。
3. [設定の編集] を選択します。
4. 必要に応じて、統合、S3 バケット、コードレビュー設定、CloudWatch ログ、またはサービスアクセスを更新します。

5. [保存] を選択します。

次の手順

コードレビュー設定をセットアップした後:

- ウェブアプリケーションを起動してコードレビューを作成および実行する (「」を参照the section called “コードレビューを作成する”)
- コードベースの増加に応じて追加の GitHub リポジトリまたは S3 バケットを接続する
- 組織固有の検証用にセキュリティ要件パックを設定する (「」を参照the section called “セキュリティ要件を管理する”)
- GitHub でのプルリクエストの検出結果の表示方法を確認する (「」を参照the section called “プルリクエストでコードセキュリティの検出結果を確認する”)

脅威モデリングを有効にする

ソースコードリポジトリを接続し、AWS リソースを設定して、脅威モデリングを有効にするようにエージェントスペースを設定します。脅威モデリングは、アプリケーションのアーキテクチャを分析し、ソースコード、設計ドキュメント、またはその両方からセキュリティの脅威を特定します。

脅威モデリング設定の設定は、エージェントスペース全体のオペレーションです。接続する統合と S3 バケットは、脅威モデリング、コードレビュー、ペネトレーションテストなどの機能間で共有されます。

セットアップが完了すると、ユーザーは AWS Security Agent ウェブアプリケーションで脅威モデルを作成して実行できます。

> **Note**
>
> エージェントスペースに接続されているリポジトリまたは S3 バケットが既にある場合 (コードレビューや侵入テストのセットアップなど)、脅威モデリングが既に有効になっている可能性があります。ウェブアプリケーションに直接移動して、脅威モデルを作成できます。「the section called “脅威モデルを作成する”」を参照してください。

前提条件

開始する前に、以下があることを確認してください。

- AWS マネジメントコンソールで作成されたエージェントスペース (「」を参照the section called "エージェントスペースを作成する")
- エージェントスペースの統合を設定するアクセス許可
- (オプション) AWS セキュリティエージェントアプリがインストールされている GitHub、GitLab、または Bitbucket 組織 (「AWS セキュリティエージェントを GitHub リポジトリに接続する」、「AWS セキュリティエージェントを GitLab リポジトリに接続する」、または「AWS セキュリティエージェントを Bitbucket リポジトリに接続する」を参照)

脅威モデリング設定ウィザードにアクセスする

エージェントスペースの脅威モデリング設定に移動します。

1. AWS セキュリティエージェントコンソールで、エージェントスペースを選択します。
2. 脅威モデルカードまたは脅威モデルタブから脅威モデルの設定を選択します。

脅威モデルの設定ウィザードに移動します。このウィザードには 2 つのオプションステップがあります。

ステップ 1: ソースコードリポジトリを接続する (オプション)

脅威モデリングを有効にする GitHub、GitLab、または Bitbucket リポジトリを接続します。脅威モデル自体がウェブアプリケーションで作成および表示されます。

> **Important**
>
> ここで設定された統合は、エージェントスペース間で共有されます。変更は、脅威モデリング、コードレビュー、および侵入テスト機能に適用されます。

リポジトリを接続する

1. 「接続された統合」セクションで、「追加」を選択します。
2. 使用するリポジトリを含む登録を選択します。
3. 接続する各リポジトリのチェックボックスをオンにします。
4. 保存を選択して選択を適用します。

> **Note**
>
> 統合をまだ登録していない場合は、設定 を選択して統合ページに移動し、AWS セキュリティエージェントアプリを承認できます。詳細については、「AWS セキュリティエージェントを GitHub リポジトリに接続する」、「AWS セキュリティエージェントを GitLab リポジトリに接続する」、または「AWS セキュリティエージェントを Bitbucket リポジトリに接続する」を参照してください。

1. 次へ を選択して、オプションの設定に進みます。

ステップ 2: オプションの設定

脅威モデリング環境の S3 バケット、CloudWatch ログ記録、サービスアクセス設定を構成します。これらの設定は、エージェントスペースの他の機能と共有されます。

S3 バケット (オプション)

脅威モデリング中にエージェントがコンテキストとして使用するソースコードを含む S3 バケットを追加します。

1. S3 バケットセクションで、S3 リソースの追加を選択します。
2. バケットまたはプレフィックスの S3 URI を入力します。
3. [Add] (追加) を選択します。

> **Note**
>
> 最大 10 個の S3 リソースを追加できます。

CloudWatch ログ (オプション)

脅威モデルの実行中にアプリケーションの動作をキャプチャして分析するように CloudWatch ロググループを設定します。

1. CloudWatch Logs セクションで、AWS アカウントから 1 つ以上の既存の CloudWatch ロググループを選択します。

サービスアクセス

脅威モデリングのために S3 バケットや CloudWatch ログなどの AWS リソースにアクセスするために AWS セキュリティエージェントが使用する IAM サービスロールを設定します。脅威モデリングを有効にするには、サービスロールが必要です。

1. サービスアクセスセクションで、ドロップダウンから既存の IAM サービスロールを選択するか、空のままにしてサービスロールを自動的に作成します。

> **Note**
>
> 既存のロールを選択しない場合、サービスロールが自動的に作成されます。

1. 保存を選択して設定を完了します。

セットアップ後

脅威モデリング設定を完了した後:

- エージェントスペースページの脅威モデルカードに準備完了ステータスが表示されます。
- ユーザーはウェブアプリケーションを起動し、接続されたソースコード、アップロードされたスコープドキュメント、Confluence ページ、またはこれらの入力の組み合わせから脅威モデルを作成できます。

次の手順

脅威モデリングを有効にした後:

- ウェブアプリケーションを起動して脅威モデルを作成して実行する (「脅威モデルの作成」を参照)
- コードベースの増加に応じて追加のリポジトリまたは S3 バケットを接続する
- Confluence を接続して、スコープドキュメントとしてページを選択できるようにする (「Connect AWS Security Agent to Confluence」を参照)

ユーザーが侵入テストとコードレビューの検出結果の修正を開始できるようにする

AWS マネジメントコンソールでは、自動修復を有効にして、AWS セキュリティエージェントウェブアプリケーションのユーザーが特定の検出結果の修正をリクエストできるようにします。AWS セキュリティエージェントは、各修正を、影響を受けるリポジトリに対する GitHub プルリクエストとして配信します。

エージェントスペース内からリポジトリごとに修復を有効にします。侵入テストタブとコードレビュータブは、同じリポジトリ設定ウィザードを開くため、いずれかのタブを使用して自動修復が有効な設定を管理できます。修復は両方の機能の結果に適用されるため、有効にする必要があるのはリポジトリごとに 1 回だけです。

前提条件

開始する前に、以下があることを確認してください。

1. 有効なペネトレーションテスト (「」を参照the section called “ペネトレーションテストを有効にする”) またはコードレビュー (「」を参照the section called “コードレビューを有効にする”)
2. GitHub 組織の AWS セキュリティエージェント GitHub アプリをインストールして承認しました (「」を参照the section called “AWS セキュリティエージェントを GitHub リポジトリに接続する”)

リポジトリ設定ウィザードを開く

リポジトリの設定方法に一致するエントリポイントを選択します。

侵入テストタブから

このパスを使用して、侵入テスト用の GitHub リポジトリを追加し、同じパスで修復を設定します。

1. エージェントスペースの概要ページに移動します。
2. 浸透テストタブを選択します。
3. リポジトリを所有する GitHub 登録を選択します。
   a. エージェントスペースに GitHub 登録を関連付けていない場合は、GitHub を AWS セキュリティエージェントに接続する」情報ボックスの「追加」を選択して登録を選択します。
   b. 1 つ以上の GitHub 登録がすでに関連付けられている場合は、「接続された統合」セクションで「追加」を選択して別の GitHub 登録を選択します。

4. 次へ を選択して GitHub リポジトリを選択します。
5. 次へ を選択してリポジトリ機能を設定します。

コードレビュータブから

リポジトリがコードレビュー用に既に接続されている場合は、このパスを使用します。

1. エージェントスペースの概要ページに移動します。
2. コードレビュータブを選択します。
3. GitHub リポジトリテーブルで、編集 を選択してリポジトリ設定ウィザードを開きます。

自動修復と保存を有効にする

リポジトリ機能に到達したら、上記のいずれかのパスを実行します。

1. 自動修復有効列で、修復する各リポジトリを有効としてマークします。
2. Connect を選択して設定を保存します。

ウェブアプリケーションのユーザーは、これらのリポジトリでの検出結果の修正を開始できるようになりました。AWS セキュリティエージェントは、提案された修正でプルリクエストを開きます。

S3 バケットからエージェントリソースを提供する

侵入テストをサポートする API ドキュメント、脅威モデルドキュメント、ソースコードなど、S3 バケット内のリソースへのアクセス権を AWS セキュリティエージェントに付与できます。

AWS セキュリティエージェントに AWS マネジメントコンソールの S3 バケットへの読み取りアクセス権を付与します。Security Agent ウェブアプリでは、ユーザーは必要に応じて S3 から個々のリソースを選択します。

1. エージェントスペースの概要ページに移動する
2. アクションを選択し、ペネトレーションテスト設定を編集する
3. S3 バケットセクションで、S3 バケットを含む S3 URI を定義します。 S3 複数の S3 バケットを追加できます。

ペネトレーションテスト用のアプリケーションドメインを有効にする

アプリケーションで侵入テストを実行する前に、ターゲットドメインを追加して所有権を検証する必要があります。AWS セキュリティエージェントは、検証済みドメインに対してのみ侵入テストを実行します。

> **Note**
>
> アプリケーションが使用する補助ドメインを検証する必要はありません。ペネトレーションテストをアクティブに実行するドメインを検証するだけで済みます。

ステップ 1: ターゲットドメインを追加する

ターゲットドメインを追加するには、エージェントスペースの概要ページの侵入テストタブに移動します。既にペネトレーションテストを設定しているかどうかに応じて、次のいずれかの方法を使用します。

- 初回セットアップ: ペネトレーションテストの設定を選択して、ペネトレーションテストウィザードを開きます。最初のステップで、ドメインを入力し、検証方法を選択します。
- 既存の設定にドメインを追加する: Target Domains テーブルで、Add domain を選択します。ドメインを入力し、モーダルで検証方法を選択します。

ベースドメインまたは `example.com`や などのサブドメインを追加できますbilling.example.com。AWS では、TXTレコードを作成するアクセス許可があるサブドメインを使用することをお勧めします。

> **Note**
>
> DNS TXT または HTTP ルート検証を使用してドメインを検証すると、そのドメインのサブドメインが自動的にカバーされます。たとえば、 を検証するexample.comと、追加の検証billing.example.comなしで api.example.comまたは をテストできます。プライベート VPC 検証の場合、ターゲットエンドポイントドメイン名は検証用に設定された完全なドメイン名と一致する必要があります。

次のいずれかの検証方法を選択します。

- DNS TXT レコード: DNS プロバイダーで DNS TXT レコードを作成して、ドメインの所有権を検証します。
- HTTP ルート: AWS セキュリティエージェントが提供する一意のトークンを含むルートをウェブサーバー上に作成して、ドメインの所有権を検証します。
- プライベート VPC: プライベート VPC 侵入テストにのみ使用できます。ドメインがプライベート CIDR 範囲内の IP に解決されることを確認します。設定されるドメイン名は、関連付けられたターゲットエンドポイントの完全なドメイン名と一致する必要があります

ステップ 2: ドメインの所有権を検証する

ドメインを追加したら、選択した方法を使用して所有権を確認します。ドメインを選択し、検証を選択すると、ターゲットドメインテーブルからいつでも検証をトリガーできます。

DNS TXT レコードを使用した検証

AWS セキュリティエージェントは TXT DNS レコード値を生成します。所有権を証明するには、DNS プロバイダーにこのレコードを追加する必要があります。

ドメインが Route 53 (同じ AWS アカウント) に登録されている場合:

AWS セキュリティエージェントは DNS レコードを自動的に作成できます。ターゲットドメインテーブルからドメインを選択し、ワンクリック検証を選択します。AWS セキュリティエージェントは DNS 検証レコードを作成し、検証を自動的に完了します。

ドメインが別の DNS プロバイダーに登録されている場合:

1. AWS セキュリティエージェントから提供された TXT レコード値をコピーします。
2. DNS レジストラに TXT レコードを追加します。
3. Target Domains テーブルに戻り、ドメインを選択し、Verify を選択します。

HTTP ルート検証を使用した検証

この方法は、ウェブサーバーに一意のトークンを配置することでドメインの所有権を証明します。所有権を証明するウェブサーバーでルートを作成できるのは、ドメイン所有者または承認されたウェブ管理者のみです。

1. ウェブサーバーの次のパスにファイルを作成します。

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

2. 次の形式を使用して、AWS セキュリティエージェントから提供されたトークンを ファイルに配置します。

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

3. Target Domains テーブルに戻り、ドメインを選択し、Verify を選択します。AWS セキュリティエージェントは、HTTPS GET リクエストを検証 URL に送信し、トークンを検証します。
4. パブリックインターネットでドメインにアクセスできる場合は、検証を実行する前にドメインに有効な SSL 証明書があることを確認してください。

> **Note**
>
> ドメインが複数のエージェントスペースに登録されており、HTTP ルート検証を使用している場合は、両方のエージェントスペースのトークンを同じtokens配列に配置することができます。

ドメインの所有権の検証をバイパスする

エンドポイントで侵入テストを実行する権限を持つが、所有権の検証を完了できないお客様は、手動検証をリクエストできます。このリクエストを行うには、カスタマーサポートケースを開いてください。リクエストには、ビジネスユースケースと、手動の所有権検証の根拠 (エンドポイントで侵入テストを実行する権限がある理由) を含める必要があります。

ペネトレーションテストに使用されるマネージドターゲットドメイン

AWS マネジメントコンソールでは、エージェントスペースが消費するターゲットドメインを追加および管理して、侵入テストを行うことができます。これらのターゲットドメインは、ペネトレーションテストで使用する前に検証する必要があります。ターゲットドメインの検証の詳細については、「」を参照してください。 the section called “ペネトレーションテストを有効にする”

前提条件

開始する前に、以下があることを確認してください。

1. 有効なペネトレーションテスト (「」を参照the section called “ペネトレーションテストを有効にする”)

ターゲットドメインリソースを管理する

- ターゲットドメインの概要ページに移動します。
- アカウントに関連付けられているすべてのターゲットドメインリソースが表示されます
  - アカウントに関連付けられたターゲットドメインが表示されない場合は、「」の手順に従ってターゲットドメインthe section called “ペネトレーションテストを有効にする”を作成して検証します。
- ターゲットドメインはエージェントスペース間で再利用でき、検証ステータスを共有できます
  - 既存のターゲットドメインをエージェントスペースに追加するには、エージェントスペースの侵入テストタブに移動します。ドメインを追加を選択し、ドメイン名フィールドで以前に登録された利用可能なドメインから選択で目的のドメインを選択します。
  - ターゲットドメインは、侵入テストで使用する前にエージェントスペースに関連付ける必要があります
- エージェントスペースからドメインを削除しても、ドメインは削除されません。関連付けられたターゲットドメインは、ターゲットドメインの概要ページから完全に削除できます。

ターゲットドメインを検証する

ターゲットドメインをペネトレーションテストで使用するには、まず次のいずれかの方法を使用して検証する必要があります。

- Route 53 ドメイン (同じ AWS アカウント): ワンクリック検証を選択します。AWS セキュリティエージェントは DNS レコードを自動的に作成し、検証を完了します。
- DNS TXT (他の DNS プロバイダー): 検証トークンをコピーし、DNS レジストラに TXT レコードを追加し、ドメインを選択して検証を選択します。
- HTTP ルート: 検証トークンをウェブサーバーの必要なルートパスに配置し、ドメインを選択して検証を選択します。詳細については、「the section called “ペネトレーションテスト用のアプリケーションドメインを有効にする”」を参照してください。
- プライベート VPC: ターゲットドメインの IP がプライベート CIDR 範囲内にあることを確認します (プライベート CIDR 範囲のリストthe section called “エージェントをプライベート VPC リソースに接続する”については、「」を参照してください)。ペネトレーションテストのターゲット工

ンドポイントドメイン名は、検証用に設定された完全なドメイン名と一致する必要があります。プライベート VPC 侵入テストにのみ使用可能

> **Note**
>
> DNS TXT、HTTP ルート、Route 53 検証の場合、検証済みドメインのサブドメインは自動的にカバーされ、個別の検証は必要ありません。プライベート VPC の検証では、各ドメインを個別に検証する必要があります。

セキュリティ要件を管理する

AWS セキュリティエージェントが設計およびコードレビュー中に評価するセキュリティ要件を定義します。AWS マネージドパック、カスタムパック、または独自のドキュメントからインポートされた要件を使用します。

セキュリティ要件を管理する

AWS Security Agent がアプリケーションの分析に使用するセキュリティ要件をセキュリティ要件パックに整理します。パックは、セキュリティ要件のコレクションです。セキュリティドキュメントをアップロードすることで、 AWSマネージドパックを有効にしたり、独自のカスタムパックを作成したり、カスタムパックの要件を生成したりできます。

概要

セキュリティ要件は、設計レビューとコードレビュー中に AWS Security Agent がアプリケーションを評価するセキュリティ標準またはポリシーを定義します。設計またはコードが要件を満たさない場合、 AWS Security Agent は検出結果を報告します。すべてのセキュリティ要件は、セキュリティ要件パックに属します。セキュリティ要件はすべてのエージェントスペースで共有され、設計およびコードレビュー中に評価されます。

AWS Security Agent には、別々のタブに表示される 2 種類のパックがあります。

- マネージドセキュリティ要件パック – AWS業界標準とベストプラクティスに基づいて提供されるセキュリティ要件のパック。これらのパックをすぐに有効にして、カスタム設定なしで一般的なセキュリティポリシーと照らし合わせて評価できます。マネージドパックの要件は読み取り専用です。 AWSマネージド要件は、カスタム要件のテンプレートとして使用できます。使用可能なマ

ネージドパックのリストについては、AWS 「 マネージドセキュリティ要件パック」を参照してください。

- カスタムセキュリティ要件パック – 組織の特定のポリシーと標準を適用するために作成するパック。要件をゼロからカスタムパックに構築したり、開始点として AWS管理要件をカスタマイズしたり、セキュリティドキュメントをアップロードして生成したりできます。

> **Note**
>
> コンプライアンスフレームワークパックは、特定のフレームワークへのコンプライアンスに関連する可能性のあるセキュリティ要件を特定するのに役立つように設計されています。これらは、コンプライアンスを評価したり、監査に合格することを保証するものではありません。

パックを単位として有効または無効にします。パックを有効にすると、 AWS Security Agent は設計およびコードレビュー中に、そのパックの要件に照らしてアプリケーションを評価します。

> **Note**
>
> パックの有効化または無効化は、新しい設計レビューとコードレビューに適用されます。既存のレビューは影響を受けません。

マネージドパックを有効または無効にする

AWSマネージドパックを有効にして、 AWS一連のメンテナンスされたセキュリティ要件に照らしてアプリケーションを評価します。不要になったら無効にします。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. マネージドセキュリティ要件パックタブを選択します。
4. 有効化または無効化するパックを選択します。
5. 次のいずれかを行います。
   a. パックを有効にするには、パックを有効にするを選択します。
   b. パックを無効にするには、パックを無効にするを選択します。

> **Tip**
>
> パックを選択すると、パックに含まれるセキュリティ要件が表示されます。適用性、コンプライアンス基準、修復ガイダンスなど、完全な定義を表示する要件を選択します。

カスタムパックを作成する

組織に固有のセキュリティ要件をグループ化するカスタムパックを作成します。パックを作成したら、手動で要件を追加したり、 AWS管理要件をカスタマイズしたり、ドキュメントをアップロードして要件を生成したりできます。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択します。
4. セキュリティ要件パックの作成を選択します。
5. セキュリティ要件パック名とオプションの説明を入力します。
6. セキュリティ要件パックの作成を選択します。

> **Note**
>
> パックを作成したら、ソースドキュメントを追加またはアップロードして、パックのセキュリティ要件を生成できます。

セキュリティ要件を手動で追加する

AWS セキュリティエージェントが適用するセキュリティ標準を定義するために、カスタムパックにセキュリティ要件を追加します。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択し、要件を追加するパックを選択します。
4. 要件を手動で追加を選択します。
5. 以下のフィールドを設定します。

a. セキュリティ要件名 – セキュリティコントロールを識別するわかりやすい名前を入力します `enforce-encryption-at-rest` (最大 80 文字) 。

b. 説明 – このセキュリティ要件がチェックする内容の簡単な説明を入力します (最大 500 文字) 。

c. 適用性 – このセキュリティ要件を評価するシナリオ、システムタイプ、または条件について説明します。NOT_APPLICABLE (最大 10,000 文字) とマークする必要がある要件のシナリオを含めます。

d. コンプライアンス基準 – このセキュリティ要件のコンプライアンスと非コンプライアンスを構成するものを定義します。 AWS セキュリティエージェントがコンプライアンスを評価するときに探す指標、例、技術的詳細 (最大 10,000 文字) を提供します。

e. 修復ガイダンス (オプション) – 組織の内部ドキュメントまたは標準へのリンク (最大 10,000 文字) など、違反を修正する方法に関するガイダンスを提供します。

6. 次のいずれかを行います。

a. 有効にせずに要件を作成するには、「セキュリティ要件の作成」を選択します。

b. 要件を作成して有効にするには、「セキュリティ要件を作成して有効にする」を選択します。

> **Note**
>
> 要件は、それを含むパックが有効になっている場合にのみ、セキュリティレビュー中に評価されます。パックが評価されるかどうかを制御するには、パックを有効または無効にします。

AWSマネージドセキュリティ要件をカスタマイズする

AWS管理要件を出発点として使用するカスタムセキュリティ要件を作成します。 AWS Security Agent は、管理要件の要件の詳細を事前に入力し、組織に合わせて編集します。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択し、要件を追加するカスタムパックを選択します。
4. カスタムセキュリティ要件の作成 を選択します。
5. フォームに事前入力するには、 AWS「 マネージドセキュリティ要件のカスタマイズ」セクションで、テンプレートとして使用する AWSマネージド要件を検索して選択します。

6. 組織のニーズに合わせてフィールドを編集します。
7. 次のいずれかを行います。
   a. 有効にせずに要件を作成するには、「セキュリティ要件の作成」を選択します。
   b. 要件を作成してすぐに有効にするには、「セキュリティ要件を作成して有効にする」を選択します。

> **Note**
>
> AWSマネージド要件をカスタマイズすると、独立したカスタムセキュリティ要件が作成されます。 AWSマネージド型要件を後で変更しても、カスタムバージョンには影響しません。

ドキュメントをアップロードして要件を生成する

各要件を手書きではなく、既存のセキュリティドキュメントからカスタムパックのセキュリティ要件を生成します。 AWS Security Agent は、アップロードしたドキュメントを読み取り、セキュリティ関連のコンテンツを識別し、確認および編集できる構造化された要件を生成します。詳細な手順については、「ドキュメントからセキュリティ要件を生成する」を参照してください。アップロードするドキュメントに含める内容のガイダンスについては、「要件生成のためのドキュメントの準備」を参照してください。

> **Important**
>
> 新しいソースドキュメントをアップロードすると、パックのすべての要件が再生成されます。手動で追加したものを含め、既存の要件はすべて置き換えられます。

カスタムセキュリティ要件を編集する

カスタムセキュリティ要件を変更して、その定義、基準、または修復ガイダンスを更新します。AWSマネージドパックの要件は編集できません。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択し、要件を含むパックを選択します。
4. 編集する要件を選択し、カスタムセキュリティ要件の編集を選択します。

5. 必要に応じて要件フィールドを更新します。セキュリティ要件名は、作成後に変更することはできません。
6. セキュリティ要件の更新を選択します。

> **Note**
>
> セキュリティ要件の変更は、新しい設計レビューとコードレビューに適用されます。既存のレビューは影響を受けません。

パックを有効または無効にする

セキュリティ要件パックを有効または無効にして、 AWS セキュリティエージェントがそれに含まれる要件を評価するかどうかを制御します。パックを単位として有効または無効にします。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. パックタイプのタブを選択し、パックを選択します。
4. 次のいずれかを行います。
   a. パックを有効にするには、パックを有効にするを選択します。
   b. パックを無効にするには、パックを無効にするを選択します。

> **Note**
>
> パックを有効にすると、セキュリティレビュー中にパックの要件が評価されます。パックが無効になっている場合、その要件は評価されません。

カスタムセキュリティ要件を削除する

AWS Security Agent による評価が不要になった場合は、カスタムセキュリティ要件を削除します。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択し、要件を含むパックを選択します。
4. 1 つ以上のセキュリティ要件を選択し、セキュリティ要件の削除を選択します。

5. 削除を確定します。

> **Note**
>
> セキュリティ要件を削除すると、今後のレビューでは評価されなくなります。要件は、過去のレビューでも読み取り専用の結果として表示されます。

セキュリティ要件を定義するためのベストプラクティス

セキュリティ要件を作成するときは、以下のガイドラインに従って、 AWS Security Agent がアプリケーションを正確に評価し、実用的な検出結果を提供するのに役立ちます。

セキュリティ要件名 – セキュリティコントロールを識別する明確で具体的な名前を使用します。要件の目的を伝えない一般的な用語は避けてください。

説明 – コントロールがチェックする内容と軽減するリスクについて説明します。これにより、ユーザーは要件が重要である理由を理解できます。

適用性 – 要件を評価するワークロード、システムタイプ、または条件を記述します。誤検出を避けるために、特定のシナリオで要件を NOT_APPLICABLE とマークする必要があるタイミングを記述します。「このコントロールは、」と「NOT_APPLICABLE if...」が明確なスコープ境界を設定するすべてのワークロードに適用されます。

コンプライアンス基準 – これを 2 つの部分に分けて構成します。コンプライアンスを示すものと、コンプライアンス違反を示すものです。テクニカルインジケータに特化し、エッジケースを含めます。「設計が準拠している場合は」、「技術詳細」、「設計が非準拠の場合は」、「違反を示すパターン」から始めます。

修復ガイダンス – 技術的な詳細と設定例を含むガイダンスを提供します。開発者が違反を修正するために必要なリソースを確保できるように、組織の社内ドキュメントへのリンクを含めます。

次の手順

セキュリティ要件パックを設定した後:

- 設計レビューまたはコードレビューを作成して、有効にしたパックに対してアプリケーションを評価します。
- ペネトレーションテストを有効にして、設計とコードレビューを補完します。
- AWS セキュリティエージェントのウェブアプリへのアクセス権をユーザーに付与します。

AWS マネージドセキュリティ要件パック

AWS マネージドセキュリティ要件パックは、業界標準とベストプラクティスに基づいて が AWS 作成および維持するセキュリティ要件のコレクションです。マネージドパックを有効にして、設計レビューやコードレビュー中にアプリケーションを要件と照らし合わせて評価できます。要件を自分で記述する必要はありません。

マネージドパックのセキュリティ要件は読み取り専用です。これらは変更できません。 AWSマネージド要件をテンプレートとして使用してカスタム要件を作成し、組織に合わせてコピーを編集できます。詳細については、「セキュリティ要件の管理」を参照してください。

AWS は、マネージドパックを経時的に更新します。がマネージドパックの要件 AWS を追加または修正すると、その変更はパックが有効になっているすべてのアカウントに適用されます。

> **Note**
>
> コンプライアンスフレームワークパックは、特定のフレームワークへのコンプライアンスに関連する可能性のあるセキュリティ要件を特定するのに役立つように設計されています。これらは、コンプライアンスを評価したり、監査に合格することを保証するものではありません。

AWS マネージドパック: ASA ベースパック

ほとんどのワークロードに適用されるセキュリティ要件のベースラインセット。このパックでは、認証と認可、データ保護、ログ記録、入力検証などの一般的なアプリケーションセキュリティの懸念について説明します。このパックを有効にして、設計とコードレビューのセキュリティベースラインを確立します。

AWS マネージドパック: AWS Well-Architected Pack

AWS Well-Architected フレームワークのセキュリティの柱から派生したセキュリティ要件。このパックは、データ保護、ID とアクセス許可の管理、セキュリティイベントの検出と対応に関する AWS ガイダンスに照らしてアプリケーションを評価します。フレームワークの詳細については、「セキュリティの柱 - AWS Well-Architected フレームワーク」を参照してください。

AWS マネージドパック: NIST CSF Pack

NIST Cybersecurity Framework (CSF) から派生したセキュリティ要件。このパックは、アイデンティティとアクセスの管理、データセキュリティ、保護テクノロジーなど、フレームワークから引き

出されたコントロール領域に対してアプリケーションを評価します。このパックを有効にして、設計とコードレビューを NIST CSF に合わせます。

AWS マネージドパック: PCI DSS Pack

Payment Card Industry Data Security Standard (PCI DSS) から派生したセキュリティ要件。このパックは、アクセスコントロール、転送中および保管中のデータの暗号化、ログ記録など、カード所有者データの処理に関連するコントロール領域に対してアプリケーションを評価します。このパックを有効にして、設計とコードレビューを PCI DSS に合わせます。

ドキュメントからセキュリティ要件を生成する

既存のセキュリティドキュメントをアップロードして、カスタムパックのセキュリティ要件を生成します。 AWS Security Agent は、アップロードしたドキュメントを読み取り、セキュリティ関連のコンテンツを識別し、適用性、コンプライアンス基準、修復ガイダンスを含む構造化されたセキュリティ要件を生成します。生成された要件は、レビューで使用される前にレビューおよび編集できます。

セキュリティポリシー、標準、またはガイドラインが既に文書化されている場合は、各要件を手動で記述する代わりに、これを使用してください。アップロードするドキュメントに含める内容のガイダンスについては、「要件生成のためのドキュメントの準備」を参照してください。

サポートされるファイル形式

次のファイル形式をアップロードできます。

- DOC
- DOCX
- MD
- PDF
- TXT

要件の生成

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択します。

4. パックを作成するか、要件を生成する既存のカスタムパックを選択します。
5. ファイルの選択を選択するか、ドキュメントをアップロード領域にドラッグアンドドロップします。
6. 「要件の生成」を選択します。
7. AWS セキュリティエージェントがドキュメントを処理するまで待ちます。生成はバックグラウンドで実行され、大きなファイルの場合は数分かかることがあります。生成の進行中にパックの別のアップロードを開始することはできません。
8. 生成されたセキュリティ要件を確認します。必要に応じて要件を編集、削除、または追加します。 AWS セキュリティエージェントがセキュリティレビュー中に要件を評価する場合は、パックを有効にします。

> **Note**
>
> AWS Security Agent はワークロードレベルで要件を生成し、ドキュメント内のセキュリティ関連のコンテンツに焦点を当てます。生成する要件の数は、提供するコンテンツによって異なります。生成された要件を確認して、インテントが反映されていることを確認します。

要件を新しいアップロードに置き換える

新しいソースドキュメントをアップロードすると、パックの要件が再生成されます。

> **Important**
>
> 新しいソースドキュメントをパックにアップロードすると、 AWS Security Agent はそのパックのすべての要件を再生成します。手動で追加したものを含め、既存の要件はすべて置き換えられます。現在の要件を維持するには、新しいドキュメントをアップロードしないでください。

1. AWS コンソールで、 AWS セキュリティエージェントに移動します。
2. ナビゲーションペインで、セキュリティ要件を選択します。
3. カスタムセキュリティ要件パックタブを選択し、パックを選択します。
4. 新しいアップロードを開始し、新しいソースドキュメントをアップロードすると既存の要件が置き換えられることを確認します。

5. ファイルを選択し、要件の生成を選択します。

次の手順

- 生成された要件を確認して有効にします。「セキュリティ要件の管理」を参照してください。
- 設計レビューまたはコードレビューを作成して、パックに対してアプリケーションを評価します。

要件生成用のドキュメントを準備する

ドキュメントからセキュリティ要件を生成すると、 AWS Security Agent はドキュメントを読み取り、見つかったセキュリティ関連のコンテンツから構造化された要件を生成します。特定の方法でドキュメントをフォーマットしたり、適用性、コンプライアンス基準、修復ガイダンスを事前に記述したりする必要はありません。 AWS Security Agent は、提供されたコンテンツからドキュメントを取得します。既に持っているセキュリティドキュメントを、自分の言葉でアップロードします。

このページでは、 AWS Security Agent が正確で有用な要件を生成できるように、何を含めるかについて説明します。アップロード手順とファイル制限については、「ドキュメントからセキュリティ要件を生成する」を参照してください。

含める内容

AWS Security Agent は、セキュリティ上の期待を説明するコンテンツを探します。以下のトピックをカバーするドキュメントは、最も強力な要件を生成します。

- アクセスコントロールと認可
- 認証
- データ保護と暗号化
- ネットワークセキュリティ
- ログ記録と監査
- インシデントへの対応
- 脆弱性とパッチ管理
- ワークロードに適用されるコンプライアンス義務

適切なソースドキュメントには、セキュリティポリシー、エンジニアリング標準、コントロールカタログ、アーキテクチャガイドライン、安全なコーディング標準などがあります。

ワークロードレベルで書き込む

AWS Security Agent は、個々のワークロードまたはアプリケーションに適用される要件を生成します。これは、設計およびコードレビュー中に評価するのと同じ範囲です。安全なワークロードを構築して運用する方法を説明するコンテンツは、要件を生成します。マルチアカウント戦略、ルートユーザー管理、アカウントレベルのログ記録設定などの組織全体のガバナンストピックは、この範囲外であり、ワークロード要件を生成しません。

単一の実装ではなく、結果を記述する

1 つの所定の製品または設定ではなく、期待されるセキュリティ成果を記述します。 AWS Security Agent は、必要なセキュリティ機能に焦点を当てた要件を生成し、複数の有効な実装を許可します。例えば、「保管中のデータが暗号化されている」とすると、特定のサービスまたは設定に 1 つの名前を付けるステートメントよりも明確で広範に適用される要件が生成されます。

良い点がどのようなものか具体的に説明する

ドキュメントが期待される動作をより具体的に記述すればするほど、 AWS セキュリティエージェントはコンプライアンス基準を定義できます。可能な場合は、準拠設計が示すものと違反を示すものについて説明します。曖昧なステートメントや純粋に批判的なステートメントは、ワークロードに関連する具体的なものよりも、より少ない要件と弱い要件を生成します。

戻った内容を確認する

生成された各要件には、 AWS Security Agent がドキュメントから派生した名前、適用性、コンプライアンス基準、修復ガイダンスが含まれます。生成された要件を確認し、絞り込みが必要な要件を編集し、 AWS セキュリティエージェントが評価する要件を有効にします。詳細については、「セキュリティ要件の管理」を参照してください。

ユーザーとアクセスを管理する

各エージェントスペースにアクセスできるユーザーを制御し、AWS セキュリティエージェントが使用する IAM ロールと暗号化キーを設定します。

ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する

AWS セキュリティエージェントは、セットアップ時にエージェントスペースをどのように設定したかに応じて、ユーザーがウェブアプリケーションにアクセスするための 2 つの方法を提供します。

アクセス方法の概要

IAM アイデンティティセンター (SSO) - エージェントスペースの作成時に IAM アイデンティティセンターを有効にした場合、ユーザーは SSO を介してウェブアプリケーションに直接アクセスできます。AWS セキュリティエージェントコンソールを使用してエージェントスペースにユーザーを割り当てると、ユーザーはアイデンティティセンターの認証情報を使用してログインします。

管理者アクセス - AWS コンソールにアクセスできるユーザーは、AWS マネジメントコンソールのエージェントスペースの概要ページの管理者アクセスリンクからウェブアプリケーションを起動できます。

IAM アイデンティティセンター (SSO) でアクセス権を付与する

IAM アイデンティティセンターでエージェントスペースを設定した場合、AWS セキュリティエージェントコンソールを使用してエージェントスペースにユーザーを割り当てることができます。

AWS セキュリティエージェントコンソールを使用してユーザーを割り当てる

1. AWS セキュリティエージェント管理コンソールで、エージェントスペースに移動します。
2. ウェブアプリタブを選択します。
3. Users テーブルで、Add users を選択します。
4. IAM Identity Center から既存のユーザーを選択するか、新しいユーザーを作成します。
5. ユーザーの割り当てを確認します。

> **Tip**
>
> エージェントスペースに割り当てられたユーザーは、SSO 認証情報を使用して IAM アイデンティティセンター経由でログインすることで、ウェブアプリケーションにアクセスできます。

SSO を使用してウェブアプリケーションにアクセスする

ユーザーがエージェントスペースに割り当てられた後:

1. ユーザーは、エージェントスペースのウェブアプリケーション URL に移動します。

> **Tip**
>
> ウェブアプリ URL のコピーを選択して、AWS セキュリティエージェントコンソールのエージェントスペースの詳細ページでウェブアプリ URL を見つけます。ユーザーは、簡単にアクセスできるようにこの URL をブックマークする必要があります。

2. ユーザーは SSO 認証情報を使用してログインします。
3. 認証後、ユーザーはエージェントスペースを選択し、セキュリティ評価の実行を開始できます。

IAM のみのアクセスでアクセスを付与する

エージェントスペースに IAM 専用アクセスを設定した場合、AWS コンソールアクセスを持つユーザーは管理者アクセスリンクからウェブアプリケーションを起動できます。

管理者アクセスリンクを使用する

1. AWS セキュリティエージェントコンソールにログインします。
2. アクセスするエージェントスペースに移動します。
3. エージェントスペースランディングページのウェブアプリタブで、管理者アクセスボタンを選択してウェブアプリケーションを起動します。
4. ウェブアプリケーションが新しいタブで開き、ユーザーが自動的に認証されます。

> **Note**
>
> 管理者アクセスリンクは、適切な AWS セキュリティエージェントのアクセス許可を持つ AWS コンソールに対して既に認証されているユーザーのみが使用できます。この方法では、IAM Identity Center の設定は必要ありません。

次の手順

ユーザーにウェブアプリケーションへのアクセスを許可した後:

- ユーザーは侵入テストの設定と実行を作成および管理できます
- ユーザーが設計レビューを作成および管理できる

- ユーザーがセキュリティ検出結果と修復ガイダンスを表示できる
- セキュリティ検出結果の通知設定を構成する

AWS セキュリティエージェントの IAM ロールを作成する

AWS セキュリティエージェントは、次の 3 つの方法で IAM ロールを使用します。

1. アプリケーションロール: AWS セキュリティエージェントアプリケーションを作成するときに使用されます。IAM アイデンティティセンターと管理者アクセスリンクのユースケースでは、このロールを引き受けて、AWS セキュリティエージェント API とやり取りするアクセス許可を WebApp ユーザーに付与します。 APIs
2. ペネトレーションテストサービスロール: 使用可能なロールのリストとしてエージェントスペースを作成するときに指定します。後で、WebApp ユーザーは侵入テストを作成するときにこれらのロールのいずれかを選択します。AWS セキュリティエージェントサービスは、テスト中に AWS リソースにアクセスするためにこのロールを引き受けます。
3. アクターロール: ターゲットウェブアプリケーション (AWS API Gateway APIs など) へのリクエストを認証および認可するために使用されます。これらのロールは、エージェントスペースの作成時に提供されます。AWS セキュリティエージェントエージェントは、ターゲットアプリケーションを操作するアクターロールを引き受けます。

アプリケーションロール

アプリケーションロールは、サービスで AWS セキュリティエージェントアプリケーションを作成するときに使用されます。IAM アイデンティティセンターと管理者アクセスリンクの認証シナリオの場合、AWS セキュリティエージェントサービスは、このロールを引き受けて、AWS セキュリティエージェント API を操作するために必要なアクセス許可を WebApp ユーザーに付与します。 APIs

必要な許可

このロールには、次のアクセス許可が必要です。

- AWS Security Agent API オペレーションを呼び出す
- アプリケーション設定データの読み取りと書き込み
- ユーザーセッション情報にアクセスする
- WebApp ユーザーの認証トークンを管理する

信頼ポリシー

信頼ポリシーでは、AWS セキュリティエージェントサービスがこのロールを引き受けることを許可する必要があります。

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

アクセス許可ポリシー

ロールには、次のアクセス許可を含める必要があります。

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

> **Note**
>
> 特定のアプリケーション要件と最小特権の原則に基づいてアクセス許可をカスタマイズします。

侵入テストサービスロール

ペネトレーションテストサービスロールは、使用可能なロールのリストとしてエージェントスペースを作成するときに指定されます。WebApp ユーザーは、侵入テストを作成するときに、これらのロールのいずれかを選択します。次に、AWS セキュリティエージェントサービスは、AWS リソースにアクセスしてテストするためにこのロールを引き受けます。

必要な許可

このロールには、ペネトレーションテスト中に AWS リソースにアクセスして分析するためのアクセス許可が必要です。

- VPC 設定とネットワークトポロジの読み取りと説明
- EC2 インスタンス、セキュリティグループ、ネットワーク ACLs
- IAM ポリシーとリソースのアクセス許可を分析する
- CloudWatch ログとメトリクスの読み取り
- セキュリティテストに関連する AWS サービス設定にアクセスする

信頼ポリシー

信頼ポリシーは、AWS セキュリティエージェントサービスが侵入テストオペレーションのためにこのロールを引き受けることを許可する必要があります。

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

```
        "StringEquals": {
          "sts:ExternalId": "your-external-id"
        }
      }
    }
  ]
}
```

> **Note**
>
> クロスアカウントまたはサービスアクセスを許可する場合は、セキュリティを強化するために外部 ID を使用します。

アクセス許可ポリシー

ロールには、AWS リソースへの読み取り専用アクセスを含める必要があります。これらの管理ポリシーの使用を検討してください。

- `SecurityAudit` - セキュリティ監査用の AWS 管理ポリシー
- `ViewOnlyAccess` - ほとんどの AWS サービスへの読み取り専用アクセス

または、特定のアクセス許可を持つカスタムポリシーを作成します。

```
{
  "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**
>
> ペネトレーションテストの範囲に必要な最小限のアクセス許可のみを付与します。セキュリティテストに含める AWS のサービスに基づいて、アクセス許可を確認して調整します。

アクターロール

アクターロールは、ペネトレーションテスト中にターゲットウェブアプリケーションへのリクエストを認証および認可するために使用されます。これらのロールはエージェントスペースの作成時に提供され、AWS セキュリティエージェントは、ターゲットアプリケーションエンドポイント (AWS API Gateway APIs、Lambda 関数 URLs、その他の AWS がホストするアプリケーションなど) とやり取りするためにロールを引き受けます。

必要な許可

このロールには、次のアクセス許可が必要です。

- API Gateway エンドポイントを呼び出す
- Lambda 関数を実行する
- アプリケーション固有の AWS リソースにアクセスする
- ターゲットアプリケーションの認証メカニズムによる認証
- アプリケーションエンドポイントに対して HTTP オペレーションを実行する

信頼ポリシー

信頼ポリシーでは、AWS セキュリティエージェントサービスがこのロールを引き受けることを許可する必要があります。

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

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

アクセス許可ポリシー

アクセス許可は、ターゲットアプリケーションアーキテクチャによって異なります。一般的なシナリオの例を次に示します。

API Gateway アプリケーションの場合

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

Lambda 関数 URLs

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

```
        "lambda:InvokeFunctionUrl",
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:*:function:your-function-name"
    }
  ]
}
```

Cognito 認証を使用した Application Load Balancer の場合

```
{
  "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**
>
> ターゲットアプリケーションの認証および認可要件に一致するようにアクターロールのアクセス許可を設定します。ロールには、セキュリティエージェントが侵入テスト中にシミュレートするユーザーまたはサービスと同じレベルのアクセス権限が必要です。

AWS セキュリティエージェントのカスタマーマネージドキー

デフォルトでは、 AWS Security Agent はマネージド暗号化キーを使用して保管中のすべてのデータを暗号化 AWSします。オプションで、 AWS Key Management Service (AWS KMS) のカスタマーマネージドキーを使用してデータを暗号化し、リソースを保護する暗号化キーを完全に制御できます。

AWS セキュリティエージェントは、リソースレベルのカスタマーマネージドキーをサポートします。エージェントスペースや統合などの最上位リソースを作成するときは、KMS キーを指定して、そのリソースとそのサブリソースに属するすべてのデータを暗号化できます。例えば、エージェ

ントスペースの作成時にカスタマーマネージドキーを指定すると、エージェントスペース設定、侵入テスト設定、ジョブ、実行の詳細、セキュリティ検出結果、検出されたエンドポイント、スクリーンショットなど、そのエージェントスペースに関連付けられたすべてのデータが暗号化されます。S3 バケット、CloudWatch Logs ロググループ、Secrets Manager シークレットなど、独自のカスタマーマネージドキーで既に暗号化されている AWS リソースを指定することもできます。必要な KMS アクセス許可については、「」を参照してくださいthe section called “必要な KMS アクセス許可”。

カスタマーマネージドキーの仕組み

AWS セキュリティエージェントは AWS KMS 階層キーリングを使用して、カスタマーマネージドキーでデータを暗号化します。リソースの作成時に KMS キーを指定すると、サービスは KMS キーで保護されたブランチキーを作成し、Amazon DynamoDB ベースのキーストアに保存します。暗号化オペレーションごとに、階層キーリングはアクティブなブランチキーから一意のラッピングキーを取得し、リクエストごとに一意のデータ暗号化キーを暗号化します。

階層キーリングには次の利点があります。

- リソースごとの暗号化スコープ – 各最上位リソースには独自のブランチキーがあるため、異なるリソースのデータは別々のキーで暗号化されます。
- 自動キーローテーション – AWS セキュリティエージェントはブランチキーを定期的にローテーションします。ローテーション後、新しいデータは新しいブランチキーバージョンで暗号化され、古いバージョンは以前に暗号化されたデータを復号するために保持されます。

暗号化コンテキスト

AWS セキュリティエージェントは、KMS キーを使用するすべての暗号化オペレーションで暗号化コンテキストを使用します。暗号化コンテキストは、暗号化オペレーションに追加の認証データを提供するシークレット以外のキーと値のペアのセットです。

暗号化コンテキストキーは の形式に従います。ここで`aws:securityagent:_<resource-type>_`、 *`<resource-type>`* は暗号化されるリソースのタイプです (例: `agent-space`または`integration`)。値は、リソースの Amazon リソースネーム (ARN) です。

> **Note**
>
> 統合の場合、暗号化コンテキストキーには `aws-crypto-ec:` プレフィックスが含まれ、 の形式になります`aws-crypto-ec:aws:securityagent:integration`。

暗号化コンテキストを使用して、KMS キーポリシーを特定のリソースタイプに絞り込むことができます。詳細については、「the section called “必要な KMS アクセス許可”」を参照してください。

デフォルト暗号化

カスタマーマネージドキーを指定せずにリソースを作成すると、 AWS Security Agent は基盤となるストレージサービス (Amazon DynamoDB AWSおよび Amazon S3) が提供するマネージド暗号化キーを使用してリソースを暗号化します。デフォルトの暗号化に追加の設定は必要ありません。

アプリケーションのデフォルト KMS キー

デフォルトの KMS キーはアプリケーションレベルで設定できます。デフォルトの KMS キーが設定されている場合、リソースの作成時に KMS キーを明示的に指定しない場合、 AWS Security Agent はそれを新しいエージェントスペースと統合のフォールバックとして使用します。

デフォルトの KMS キーを設定するには、CLI または SDK を使用してアプリケーションを作成または更新するときに AWS `defaultKmsKeyId`パラメータを指定します。コンソールでは、アプリケーションのセットアップ中にデフォルトの KMS キーを指定するように求められます。

リソースの作成時に KMS キーを明示的に指定すると、アプリケーションレベルのデフォルトよりも優先されます。

前提条件

カスタマーマネージドキーを設定する前に、次の前提条件を満たしてください。

- KMS で対称暗号化 KMS AWS キーを作成します。キーは次の要件を満たしている必要があります。
  - キータイプ: 対称
  - キーの使用方法: 暗号化と復号
  - キー仕様: `SYMMETRIC_DEFAULT`
  - キーの状態: 有効

    手順については、「 Key Management Service https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.htmlデベロッパーガイド」の「キーの作成」を参照してください。 AWS
- KMS キーポリシーと IAM ポリシーを設定して、 AWS セキュリティエージェントに必要なアクセス許可を付与します。詳細については、「the section called “必要な KMS アクセス許可”」を参照してください。

必要な KMS アクセス許可

AWS セキュリティエージェントには、次の 2 つのシナリオで KMS アクセス許可が必要です。

- AWS セキュリティエージェント内でのデータの暗号化 – エージェントスペースまたは統合にカスタマーマネージドキーを指定する場合、サービスにはデータの暗号化オペレーションにそのキーを使用するためのアクセス許可が必要です。
- CMK で暗号化された AWS リソースの使用 – カスタマーマネージドキー (S3 バケット、CloudWatch Logs ロググループ、Secrets Manager シークレットなど) で暗号化された AWS リソースを提供する場合、サービスには、これらのキーを使用して暗号化されたリソースにアクセスするためのアクセス許可が必要です。

AWS セキュリティエージェントは、オペレーションに応じて異なる ID を使用して KMS キーにアクセスします。

- AWS マネジメントコンソールオペレーション – 管理者が AWS マネジメントコンソールでリソースを作成または管理する (エージェントスペースの作成やアプリケーションの更新など) 場合、サービスは管理者ロールを使用して KMS AWS を呼び出します。
- ウェブアプリケーションオペレーション – IAM Identity Center ユーザーが AWS Security Agent ウェブアプリケーションを介してデータにアクセスする場合 (侵入テスト結果の表示や侵入テストの開始など) 、サービスは AWS Security Agent のセットアップ中に作成されたアプリケーションロールを使用して KMS AWS を呼び出します。
- 顧客提供のリソースへのアクセス – サービスは、侵入テスト中に顧客提供の AWS リソース (S3 バケット、CloudWatch Logs ロググループ、Secrets Manager シークレットなど) にアクセスすると、侵入テストサービスロールを使用して KMS AWS を呼び出します。
- 非同期ワークフロー – ユーザーセッションの外部で実行されるバックグラウンドオペレーション (侵入テストの実行、コードレビュー処理、ブランチキーローテーションなど) の場合、サービスは独自のサービスプリンシパル (`securityagent.amazonaws.com`) を使用してユーザーに代わって KMS AWS を呼び出します。

以下のセクションでは、各 ID に必要なアクセス許可について説明します。

キーポリシー

KMS キーポリシーは、暗号化オペレーションにキーを使用するアクセス許可を AWS セキュリティエージェントに付与する必要があります。必要なキーポリシーステートメントは、暗号化するリソースタイプと、サービスに提供する CMK 暗号化 AWS リソースによって異なります。

エージェントスペースのキーポリシー

次のキーポリシーは、侵入テスト結果やスクリーンショットなど、エージェントスペースデータを暗号化および復号するために必要なアクセス許可を AWS セキュリティエージェントに付与します。

ポリシーで次のプレースホルダー値を置き換えます。

- *111122223333* – AWS アカウント ID
- *MyRole* – コンソールで AWS セキュリティエージェントの管理に使用する IAM ロール
- *MyApplicationRole* – AWS Security Agent のセットアップ中に作成されたアプリケーションロール
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン

```
{
  "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:ReEncryptTo",
```

```
        "kms:ReEncryptFrom",
        "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:ReEncryptTo",
        "kms:ReEncryptFrom"
      ],
      "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/*"
        }
      }
    },
    {
      "Sid": "AllowAsynchronousDataAccessForCodeRemediation",
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*"
    }
  ]
}
```

Important

ブランチキーをローテーションするには、KMS キーポリシーが AWS Security Agent サービスプリンシパル (`securityagent.amazonaws.com`) に `kms:GenerateDataKeyWithoutPlaintext`、`kms:ReEncryptTo`、および アクセス`kms:ReEncryptFrom`許可を付与する必要があります。そうしないと、ブランチキーのローテーションは失敗します。必要なアクセス許可の詳細については、「ブランチキーのローテーション」を参照してください。

統合のキーポリシー

次のキーポリシーは、コードレビューの結果を含む統合データの暗号化と復号化に必要なアクセス許可を AWS セキュリティエージェントに付与します。

ポリシーで次のプレースホルダー値を置き換えます。

- *111122223333* – AWS アカウント ID
- *MyRole* – コンソールで AWS セキュリティエージェントの管理に使用する IAM ロール
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン

```
{
  "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:ReEncryptTo",
        "kms:ReEncryptFrom"
      ],
      "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:ReEncryptTo",
        "kms:ReEncryptFrom"
      ],
      "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**
>
> 複数のリソースタイプに同じ KMS キーを使用する場合は、エージェントスペース、統合、セキュリティ要件パックのキーポリシーステートメントを 1 つのキーポリシーに結合できます。

セキュリティ要件パックのキーポリシー

次のキーポリシーは、 AWS セキュリティエージェントにセキュリティ要件パックデータの暗号化と復号に必要なアクセス許可を付与します。セキュリティ要件パックはウェブアプリケーションロールを使用しません。ポリシーは 2 つのアクセスパスを使用します。

- 同期パス – API を呼び出すと、サービスは転送された認証情報を使用してユーザーに代わって (`kms:ViaService`条件を使用して) KMS AWS を呼び出します。
- 非同期パス – パックを使用できる非同期オペレーションの場合、サービスは独自のサービスプリンシパルを使用して KMS AWS を直接呼び出します。

ポリシーで次の値を置き換えます。

- *111122223333* – AWS アカウント ID
- *MyRole* – セキュリティ要件パックオペレーションの呼び出しに使用する IAM ロール
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowKeyMetadataValidation",
      "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": "AllowUseOfHierarchicalKeyringForSecurityPacks",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::111122223333:role/MyRole"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:ReEncryptTo",
        "kms:ReEncryptFrom",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "securityagent.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:securityagent:security-requirement-pack":
 "arn:aws:securityagent:us-east-1:111122223333:security-requirement-pack/*"
        }
      }
    },
    {
      "Sid": "AllowAsynchronousDataAccessForSecurityPacks",
      "Effect": "Allow",
      "Principal": {
        "Service": "securityagent.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:ReEncryptTo",
        "kms:ReEncryptFrom"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:securityagent:security-requirement-pack":
 "arn:aws:securityagent:us-east-1:111122223333:security-requirement-pack/*"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:securityagent:us-east-1:111122223333:security-
requirement-pack/*"
        }
      }
    }
  ]
```

```
}
```

ポリシーには 3 つのステートメントが含まれています。

- AllowKeyMetadataValidation – リソースの作成時に CMK を指定するときに、kms:DescribeKey AWS セキュリティエージェントがカスタマーマネージドキーが存在し、有効になっており、対称暗号化を使用していることを検証できるようにします。kms:ViaService 条件を使用して、通話が サービスを通じて発信されるようにします。
- AllowUseOfHierarchicalKeyringForSecurityPacks – 階層キーリングオペレーションの権限 kms:GenerateDataKeyWithoutPlaintext (新しいブランチキーの作成)kms:ReEncryptTo、権限 kms:ReEncryptFrom (既存のブランチキーのローテーション) 、権限 kms:Decrypt (既存のブランチキーの取得) を付与します。これらのオペレーションは、同期パスを介して転送された認証情報を使用し、暗号化コンテキスト条件によってセキュリティ要件パックリソースに限定されます。
- AllowAsynchronousDataAccessForSecurityPacks – 発信者の認証情報が利用できない場合kms:GenerateDataKeyWithoutPlaintext、非同期オペレーションのために kms:Decrypt、kms:ReEncryptTo、、および を AWS セキュリティエージェントサービスプリンシパルkms:ReEncryptFromに付与します。

エージェントスペースキーポリシーとは異なり、セキュリティ要件パックポリシーにはウェブアプリケーションロールプリンシパルは含まれません。セキュリティ要件パックには、 AWS セキュリティエージェントのウェブアプリケーションではなく、管理者ロールを使用して AWS マネジメントコンソールからアクセスします。すべての同期オペレーションでは、単一の発信者ロールが使用されます (経由kms:ViaService)。

> **Note**
>
> エージェントスペースとセキュリティ要件パックの両方に同じ KMS キーを使用する場合は、両方のセクションのキーポリシーステートメントを 1 つのキーポリシーに結合できます。

CMK で暗号化された AWS リソースのキーポリシー

カスタマーマネージドキーで暗号化された AWS リソースを AWS セキュリティエージェントに提供する場合は、各リソースの KMS キーのキーポリシーを更新して、必要なアクセス許可を付与する必要があります。これは、次のリソースに適用されます。

- S3 バケット – カスタマーマネージドキー (SSE-KMS) で暗号化された S3 バケットから学習リソースを提供する場合、キーポリシーは、侵入テストサービスロールがオブジェクトを復号することを許可する必要があります。
- Secrets Manager シークレット – 侵入テスト認証情報がカスタマーマネージドキーで暗号化された Secrets Manager シークレットに保存されている場合、キーポリシーは侵入テストサービスロールにそれらのシークレットの復号を許可する必要があります。ウェブアプリケーションがユーザーに代わってシークレットを作成する場合、キーポリシーはアプリケーションロールにそれらのシークレットの暗号化を許可する必要があります。
- CloudWatch Logs ロググループ – 侵入テスト実行ログの保存に使用される CloudWatch Logs ロググループがカスタマーマネージドキーで暗号化されている場合、キーポリシーは、CloudWatch Logs がサービスロールを介して KMS キーを検証し、CloudWatch Logs サービスプリンシパルが暗号化オペレーションを実行することを許可する必要があります。

> **Important**
>
> AWS セキュリティエージェントは、ユーザーに代わって CloudWatch Logs ロググループと Secrets Manager シークレットを作成することもできます。KMS キーポリシーを設定するときは、これらのサービス作成リソースに対応するステートメントも含めます。

次のキーポリシーステートメントは、CMK で暗号化されたリソースに必要なアクセス許可を付与します。設定に適用されるステートメントのみを含めます。リソースがエージェントスペースに指定したのと同じ KMS キーを使用している場合は、これらのステートメントをそのキーのポリシーに追加します。リソースが別の KMS キーを使用している場合は、代わりにこれらのステートメントをそのキーのポリシーに追加します。

ポリシーで次のプレースホルダー値を置き換えます。

- *111122223333* – AWS アカウント ID
- *MyApplicationRole* – AWS Security Agent のセットアップ中に作成されたアプリケーションロール
- *MyPenTestServiceRole* – 侵入テストサービスロール
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン

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

```
  "Statement": [
    {
      "Sid": "AllowKmsKeyAccessForCreatingSecrets",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyApplicationRole"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "secretsmanager.*.amazonaws.com",
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:*"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyDecryptionForS3Objects",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyPenTestServiceRole"
      },
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "s3.*.amazonaws.com",
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::YOUR-BUCKET-NAME*"
        }
      }
    },
    {
```

```
      "Sid": "AllowKmsKeyDecryptionForSecretsManagerSecrets",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyPenTestServiceRole"
      },
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "secretsmanager.*.amazonaws.com",
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YOUR-SECRET-NAME*"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyValidationForCloudWatchLogsLogGroups",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/MyPenTestServiceRole"
      },
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "logs.*.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyAccessForCloudWatchLogsLogGroups",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
```

```
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:YOUR-LOG-GROUP-NAME*"
        }
      }
    }
  ]
}
```

Note

- `AllowKmsKeyDecryptionForS3Objects` ステートメントは、カスタマーマネージドキーで暗号化された S3 バケットから学習リソースを提供する場合にのみ必要です。S3 の SSE-KMS の設定の詳細については、「SSE-KMS によるデータの保護」を参照してください。
- `AllowKmsKeyDecryptionForSecretsManagerSecrets` ステートメントは、侵入テスト認証情報がカスタマーマネージドキーで暗号化された Secrets Manager シークレットに保存されている場合にのみ必要です。サービスロールは、侵入テスト中にシークレットを復号化するためのアクセスが必要です。シークレット暗号化の詳細については、Secrets Manager の「シークレットの暗号化と復号」を参照してください。
- `AllowKmsKeyValidationForCloudWatchLogsLogGroups` ステートメントは、CloudWatch Logs `kms:DescribeKey`がロググループに関連付けるときにサービスロールを介して KMS キーを検証できるように、サービスロールを付与します。
- `AllowKmsKeyAccessForCreatingSecrets` ステートメントは、ウェブアプリケーションがカスタマーマネージドキーを使用してユーザーに代わって Secrets Manager シークレットを作成する場合に必要です。KMS キーを使用してシークレットを暗号化`kms:Decrypt`するには、アプリケーションロールに `kms:GenerateDataKey`と が必要です。

- このAllowKmsKeyAccessForCloudWatchLogsLogGroupsステートメントは、CloudWatch Logs サービスプリンシパル (logs.*us-east-1*.amazonaws.com) に、ログデータの暗号化と復号に必要なアクセス許可を付与します。このステートメントは、既存のロググループを指定しない場合に AWS 、セキュリティエージェントがユーザーに代わってロググループを作成する場合にも必要です。詳細については、「KMS を使用して CloudWatch Logs AWS のログデータを暗号化する」を参照してください。
- 複数のリソースが同じ KMS キーを共有する場合、ステートメントを 1 つのキーポリシーに結合できます。

管理者ロール

管理者ロール (コンソールで AWS セキュリティエージェントの管理に使用する IAM ロール) に追加の IAM ポリシーは必要ありません。

アプリケーションロール

KMS キーポリシーに加えて、 AWS セキュリティエージェントのセットアップ時に指定されたアプリケーションロールに次の IAM ポリシーをアタッチします。このポリシーは、エージェントスペースデータの暗号化と復号、 AWS Secrets Manager でのシークレットの作成にカスタマーマネージドキーを使用するアクセス許可をロールに付与します。

ポリシーで次のプレースホルダー値を置き換えます。

- *111122223333* – AWS アカウント ID
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン
- の KMS キー ARNs Resource — KMS キーの ARNs

```
{
  "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**
>
> アプリケーションロールは、指定した IAM ロール、または AWS セキュリティエージェントのセットアップ中にコンソールが作成する IAM ロールです。ウェブアプリケーションは

このロールを引き受けて、侵入テストの実行ログを取得し、ユーザーに代わって Secrets Manager シークレットを作成します。

- `AllowKmsKeyAccessForCreatingSecrets` ステートメントは、ペネトレーションテスト用の認証リソースを設定し、既存のシークレットを指定する代わりに認証情報を直接入力する場合に必要です。ウェブアプリケーションはユーザーに代わってシークレットを作成し、アプリケーションロールは、エージェントスペースに指定されたカスタマーマネージドキーでシークレットを暗号化するために、このステートメントのアクセス許可を必要とします。エージェントスペースに使用される KMS キーと一致するように、このステートメントの `Resource` ARN を更新します。 ARNs

サービスロール

ペネトレーションテスト中、 AWS Security Agent は、 AWS リソースにアクセスするためのペネトレーションテストサービスロールを引き受けます。これらのリソースのいずれかがカスタマーマネージドキーで暗号化されている場合は、対応する KMS キーを使用するためのアクセス許可をサービスロールに付与する必要があります。これは、次のリソースに適用されます。

- S3 バケット – カスタマーマネージドキーで暗号化された S3 バケットから学習リソース (API ドキュメント、脅威モデル、ソースコードなど) を提供する場合、サービスロールには、そのバケット内のオブジェクトを復号するためのアクセス許可が必要です。S3 の SSE-KMS の設定の詳細については、「SSE-KMS によるデータの保護」を参照してください。
- Secrets Manager シークレット – 侵入テスト認証情報がカスタマーマネージドキーで暗号化された Secrets Manager シークレットに保存されている場合、サービスロールにはそれらのシークレットを復号するためのアクセス許可が必要です。シークレット暗号化の詳細については、Secrets Manager の「シークレットの暗号化と復号」を参照してください。
- CloudWatch Logs ロググループ – 侵入テスト実行ログの保存に使用される CloudWatch Logs ロググループがカスタマーマネージドキー (ユーザーに代わって AWS Security Agent によって作成されたロググループを含む) で暗号化されている場合、サービスロールにはキーを検証する`kms:DescribeKey`アクセス許可が必要です。CloudWatch Logs サービスプリンシパルは、ログデータの実際の暗号化と復号を処理し、これらのアクセス許可はキーポリシーを通じて付与されます (「」を参照the section called “キーポリシー”)。ログデータの暗号化の詳細については、「KMS を使用して CloudWatch Logs AWS でログデータを暗号化する」を参照してください。

> **Important**
>
> エージェントスペースに指定したものとは異なる KMS キーを使用してこれらのリソースを暗号化する場合は、侵入テスト中にサービスロールがアクセスするリソースを保護する KMS キーごとにサービスロールのアクセス許可を付与する必要があります。

次の IAM ポリシーをペネトレーションテストサービスロールにアタッチします。設定に適用されるステートメントのみを含めます。

ポリシーで次のプレースホルダー値を置き換えます。

- *111122223333* – AWS アカウント ID
- *us-east-1* – AWS セキュリティエージェントを使用する AWS リージョン
- の KMS キー ARNs Resource – 各リソースの暗号化に使用されるカスタマーマネージドキーの ARNs

```
{
  "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:YOUR-SECRET-NAME*"
        }
      }
    },
    {
      "Sid": "AllowKmsKeyValidationForCloudWatchLogsLogGroups",
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE-CLOUDWATCH-LOGS-KEY-ID"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        },
        "StringLike": {
          "kms:ViaService": "logs.*.amazonaws.com"
        }
      }
    }
  ]
}
```

Note

- `AllowKmsAccessForEncryptedS3Buckets` ステートメントは、カスタマーマネージドキーで暗号化された S3 バケットから学習リソースを提供する場合にのみ必要です。S3 バケットの暗号化に使用される KMS キーと一致するように `Resource` ARN を更新し、バケット名と一致するように`kms:EncryptionContext:aws:s3:arn`値を更新します。
- `AllowKmsAccessForEncryptedSecrets` ステートメントは、侵入テスト認証情報がカスタマーマネージドキーで暗号化された Secrets Manager シークレットに保存されている場合にのみ必要です。シークレットの暗号化に使用される KMS キーと一致するように `Resource` ARN を更新し、シークレット名と一致するように`kms:EncryptionContext:SecretARN`値を更新します。
- `AllowKmsKeyValidationForCloudWatchLogsLogGroups` ステートメントは、CloudWatch Logs ロググループが、ユーザーに代わって AWS Security Agent によって作成されたロググループなど、カスタマーマネージドキーで暗号化されている場合にのみ必要です。ロググループの暗号化に使用される KMS キーと一致するように `Resource` ARN を更新します。
- 複数のリソースが同じ KMS キーを共有する場合、 ステートメントを組み合わせて、共有キー ARN を `Resource`フィールドに一度一覧表示できます。

カスタマーマネージドキーを使用してリソースを作成する

AWS セキュリティエージェントを設定するとき、または個々のリソースを作成するときに、カスタマーマネージドキーを指定できます。KMS キーは、そのリソースとそのサブリソースに属するすべてのデータを暗号化します。

セットアップ中にデフォルトの KMS キーを設定する (コンソール)

デフォルトの KMS キーは、 AWS セキュリティエージェントの初期設定時に設定できます。このキーは、別のキーを指定しない限り、新しいエージェントスペースと統合のデフォルトとして使用されます。

1. AWS セキュリティエージェントのセットアップページで、暗号化 - オプションセクションを展開します。
2. 暗号化設定をカスタマイズ (アドバンスド) を選択します。
3. AWS KMS キーを選択するには、アカウントから KMS キーを選択するか、KMS キー ARN を入力します。

4. 残りのセットアップステップを完了し、AWS セキュリティエージェントのセットアップを選択します。

AWS セキュリティエージェントは KMS キーを検証して、KMS キーが存在し、有効になっており、対称暗号化を使用していることを確認します。検証が失敗した場合、セットアップは続行されず、エラーメッセージが表示されます。

カスタマーマネージドキーを使用してエージェントスペースを作成する (コンソール)

1. AWS セキュリティエージェントコンソールで、エージェントスペースページに移動します。
2. エージェントスペースの作成を選択します。
3. エージェントスペースの名前とオプションの説明を入力します。
4. 詳細設定 セクションを展開します。
5. データ暗号化で、次のいずれかを選択します。
   - アプリケーションのデフォルトキーを使用する – AWS Security Agent のセットアップ時に設定されたデフォルトの KMS キーを使用します。このオプションは、デフォルトキーが設定されている場合、デフォルトで選択されます。
   - 別のキーを使用する – このエージェントスペースに別の KMS キーを指定します。暗号化設定をカスタマイズ (アドバンスド) を選択し、AWS KMS キーを選択する で、アカウントから KMS キーを選択するか、ARN を入力します。
6. [作成] を選択します。

カスタマーマネージドキーとの統合を作成する (コンソール)

1. AWS セキュリティエージェントコンソールで、統合ページに移動します。
2. 統合を追加を選択し、プロバイダー (GitHub など) を選択します。
3. ステップ 1: プロバイダーアプリケーションをインストールして承認します。
4. ステップ 2: 登録の詳細で、登録名を入力し、プロバイダー設定を構成します。
5. データ暗号化セクションで、暗号化設定をカスタマイズ (詳細) を選択します。
6. AWS KMS キーを選択するには、アカウントから KMS キーを選択するか、KMS キー ARN を入力します。
7. 接続 を選択します。

カスタマーマネージドキー (AWS CLI または SDK) を使用してリソースを作成する

CLI または SDK AWS を使用してカスタマーマネージドキーを指定するには:

- を呼び出しCreateApplicationてアプリケーションのデフォルトの KMS キーを設定するときは、 defaultKmsKeyIdパラメータを含めます。このキーは、明示的な KMS キーなしでエージェントスペースまたは統合を作成するときにフォールバックとして使用されます。
- CreateAgentSpace または を呼び出しCreateIntegrationて特定のリソースを暗号化する場合は、 kmsKeyIdパラメータを含めます。これは、アプリケーションレベルのデフォルトよりも優先されます。

パラメータの詳細については、AWS 「 セキュリティエージェント API リファレンス」を参照してください。

KMS キーを指定しない場合、 AWS Security Agent はアプリケーションレベルのデフォルト KMS キーを設定している場合はそれを使用します。それ以外の場合、リソースは AWSマネージドキーで暗号化されます。

キーローテーション

AWS セキュリティエージェントは、ブランチキーを定期的に自動的にローテーションします。ブランチキーローテーションは、以前のすべてのバージョンを保持しながら、ブランチキーの新しいアクティブバージョンを作成します。ローテーション後:

- 新しいデータは、新しいブランチキーバージョンで暗号化されます。
- 以前に暗号化されたデータは、古いブランチキーバージョンを使用して復号できます。
- データの再暗号化は必要ありません。

ブランチキーのローテーションは、KMS キーのローテーションとは別のものです。カスタマーマネージドキーの自動 KMS キーローテーションを個別に有効にできます。詳細については、「 Key Management Service デベロッパーガイド AWS 」の「KMS キーのローテーション」を参照してください。

ブランチキーのローテーション後、以前のブランチキーのキャッシュされたコピーを新しい暗号化オペレーションに引き続き使用できる期間が短くなります。このウィンドウは、内部キャッシュtime-to-liveによって決定され、データのセキュリティには影響しません。

考慮事項

AWS Security Agent でカスタマーマネージドキーを使用する場合は、次の点に注意してください。

- カスタマーマネージドキーは新しいリソースにのみ適用されます。カスタマーマネージドキーを設定する前に作成された既存のリソースは、引き続き AWSマネージド暗号化を使用します。既存のデータの移行はありません。
- カスタマーマネージドキーは、作成時に指定されます。リソースの作成時に KMS キーを指定します。既存のリソースの KMS キーを追加または変更することはできません。
- 複数の KMS キーがサポートされています。リソースごとに異なる KMS キーを使用できます。たとえば、本番稼働用エージェントスペースには 1 つのキーを使用し、開発用エージェントスペースには別のキーを使用できます。
- KMS キーを無効化または削除すると、データにアクセスできなくなります。KMS キーの削除を無効化またはスケジュールすると、 AWS Security Agent はそのキーで暗号化されたデータを復号できません。キーが再び有効になるまで、影響を受けるリソースにアクセスできなくなります。KMS キーを完全に削除すると、そのキーで暗号化されたすべてのデータにアクセスできなくなります。
- キーポリシーのアクセス許可が必要です。KMS キーポリシーから必要なアクセス許可を削除すると、 AWS Security Agent は暗号化されたデータにアクセスできません。「」で説明されているように、キーポリシーが管理者ロール (AWS コンソールでサービスを管理するために使用される)、アプリケーションロール (ウェブアプリケーションで使用される)、侵入テストサービスロール (エージェントが侵入テストを実行すると仮定)、および AWS セキュリティエージェントサービスプリンシパルにアクセス許可を付与していることを確認しますthe section called “必要な KMS アクセス許可”。
- AWS KMS クォータが適用されます。KMS キーに対する暗号化オペレーションは、KMS AWS リクエストクォータにカウントされます。通常の使用では、階層キーリングアーキテクチャにより、ブランチキーキャッシュによる KMS 呼び出しが最小限に抑えられます。詳細については、 AWS「 Key Management Service デベロッパーガイド」の「リクエストクォータ」を参照してください。

トラブルシューティング

AWS セキュリティエージェントを使用する際によく見られるエラーの解決策を見つけます。

アクセスが拒否されました: 選択した GitHub アカウントタイプが正しくないか、指定された組織名が正しくありません

- 目的の GitHub 組織に AWS セキュリティエージェントアプリケーションをインストールしたが、Userの代わりに GitHub Account Typeを誤って に設定している Organization
- AWS Security Agent アプリケーションを目的の GitHub 組織にインストールし、 を GitHub Account Type に正しく設定しましたOrganizationが、Organization Nameフィールドを空白のままにするか、アプリケーションをインストールした組織と一致しない間違った組織名を入力しました

解決策:

1. GitHub に移動し、組織からアプリケーションをアンインストールします。詳細については、「the section called “ステップ 1: GitHub から AWS セキュリティエージェント GitHub アプリをアンインストールする”」を参照してください。
2. 統合ページに戻り、 をクリックして統合プロセスを再起動しAdd Integrations、アプリケーションをインストールして目的の GitHub 組織に再度承認します。
3. のドロップダウンOrganizationから を選択します。 GitHub account type
4. Organization Name 入力する が、アプリケーションをインストールした EXACT と同じであることを確認します。
5. Connect を選択して、GitHub 組織統合を作成します。

アクセス拒否: GitHub アプリを組織にインストールするためのアクセス許可が不十分

目的の GitHub 組織に AWS セキュリティエージェントアプリケーションをインストールしようとすると、インストールページのボタンに 2 つの異なるメッセージが表示されます。

- 組織Memberには Authorize & Request
- 組織Ownerには Install & Authorize

以下の手順に従って、自分が GitHub 組織の Owner Memberであるか であるかを確認できます。

1. github.com に移動します。

2. ウェブサイトでプロファイルを選択する
3. ドロップダウンメニューOrganizationsで に移動して選択します。
4. 組織のリストから AWS セキュリティエージェントをインストールする組織を見つけます。組織名Ownerの横にある Memberか かを指定します。

考えられる解決策

- 統合を作成する前に、インストールリクエストを所有者に承認してもらう
- 所有者に GitHub 組織のロールを から Member に更新Ownerさせ、統合プロセスを再開させます。

エージェントがペネトレーションテスト中にエンドポイントに接続できない

ペネトレーションテストエージェントが設定されたターゲット URL を呼び出すことができない場合、またはターゲットエンドポイントを正常にナビゲートできない場合:

- エンドポイントが設定されたターゲット URL 外のドメインを呼び出す場合は、ペンテスト設定で追加のドメインがアクセス可能な URLsとして追加されていることを確認します。
- ペネトレーションテストは現在、ポート 80 または 443 でトラフィックを処理する HTTP/HTTPS エンドポイントでのみ使用できます。
- WAF を設定している場合は、WAF が侵入テストトラフィックをブロックしていないことを確認します。User-Agent ヘッダーごとにペネトレーションテストトラフィックを許可リストに登録できます。デフォルトで に設定されます。 securityagent

追加のヘルプの取得

これらのトラブルシューティング手順を試した後も問題が続く場合:

- AWS Security Agent サービスのステータスページを確認する
- 複雑な設定の問題に関するサポートについては、AWS サポートにお問い合わせください。

ユーザーと開発者向け (ウェブアプリと IDE)

ウェブアプリケーション、IDE、または接続されたソースプロバイダーで設計レビュー、脅威モデル、コードレビュー、侵入テストを実行し、検出結果を確認して修正します。

AWS セキュリティエージェントウェブアプリにログインする

AWS セキュリティエージェントウェブアプリにログインして、次のようなタスクを完了する必要があります。

- 設計レビューを実行する
- 侵入テストを実行する

アクセス方法

AWS セキュリティエージェントは、組織が初期設定時にアクセスをどのように設定したか、および AWS コンソールにアクセスできるかどうかに応じて、ウェブアプリケーションにアクセスするためのさまざまな方法を提供します。

IAM アイデンティティセンター (SSO) - 組織が IAM アイデンティティセンターを有効にしている場合は、SSO 認証情報を使用してログインできます。ウェブアプリケーションにアクセスするには、IAM Identity Center で少なくとも 1 つのエージェントスペースに割り当てる必要があります。

管理者アクセス (IAM) - 適切なアクセス許可を持つ AWS コンソールアクセスがある場合は、任意のエージェントスペースページの管理者アクセスリンクからウェブアプリケーションを起動できます。このメソッドは、既存のコンソールセッションを通じて認証を提供します。

> **Note**
>
> 組織が使用するプライマリアクセス方法は、AWS セキュリティエージェントの初回セットアップ時に決定されました。ユーザーアクセスと割り当ての設定については、「」を参照してくださいthe section called “ユーザーに AWS Security Agent ウェブアプリへのアクセスを許可する”。

IAM Identity Center (SSO) でログインする

組織が IAM Identity Center アクセスを設定している場合は、複数のエントリポイントから SSO 認証情報を使用してウェブアプリケーションにログインできます。

前提条件

- IAM アイデンティティセンターで少なくとも 1 つのエージェントスペースに割り当てられている
- IAM Identity Center SSO 認証情報がある

ウェブアプリケーションにアクセスする

必要に応じて、次のいずれかの方法を選択します。

割り当てられているすべてのエージェントスペースを表示するには:

1. AWS セキュリティエージェントコンソールで、設定に移動します。
2. ウェブアプリケーションドメインを見つけ、URL をコピーします。

> **Tip**
>
> 簡単にアクセスできるように、この URL をブックマークします。これは、割り当てられているすべてのエージェントスペースを表示するユニバーサルエントリポイントです。

3. ウェブアプリケーション URL に移動します。
4. プロンプトが表示されたら、IAM アイデンティティセンターの認証情報を入力します。
5. 認証後、割り当てられているすべてのエージェントスペースのリストが表示されます。
6. 作業するエージェントスペースを選択します。

特定のエージェントスペースに直接アクセスするには (AWS コンソールアクセスが必要です) 。

1. AWS マネジメントコンソールにログインします。
2. AWS セキュリティエージェントコンソールに移動します。
3. アクセスするエージェントスペースに移動します。
4. 次のいずれかを選択します。
   - エージェントスペースの概要ページでウェブアプリケーションを起動する

- コードレビューセクションのウェブアプリを起動するボタン
- 「侵入テスト」セクションの「ウェブアプリを起動する」ボタン

5. プロンプトが表示されたら、IAM アイデンティティセンターの認証情報を入力します。
6. 認証後、ウェブアプリケーションのそのエージェントスペースに直接移動します。

> **Note**
>
> AWS コンソールにアクセスできない場合は、設定ページからウェブアプリケーションドメインを使用して、割り当てられたすべてのエージェントスペースにアクセスします。

管理者アクセス (IAM) を使用してログインする

適切な AWS セキュリティエージェントのアクセス許可を持つ AWS コンソールアクセスがある場合は、自動認証を使用して管理者アクセスリンクからウェブアプリケーションを起動できます。

前提条件

- AWS マネジメントコンソールにログインしている
- AWS Security Agent リソースにアクセスするための適切なアクセス許可がある

ウェブアプリケーションにアクセスする

次のいずれかの方法を選択してください。

特定のエージェントスペースにアクセスするには:

1. AWS マネジメントコンソールにログインします。
2. AWS セキュリティエージェントコンソールに移動します。
3. アクセスするエージェントスペースに移動します。
4. エージェントスペースの概要ページの管理者アクセスボタンを選択します。
5. ウェブアプリケーションが新しいタブで開き、そのエージェントスペースの自動認証が行われます。

> **Note**
>
> 管理者アクセスボタンは、組織がプライマリアクセス方法として IAM アイデンティティセンターを使用しているかどうかにかかわらず、AWS コンソールにアクセスできるユーザーが常に使用できます。

設計レビューを作成する

AWS セキュリティエージェントがレビューするファイルをアップロードして、組織のセキュリティ要件に照らして設計ドキュメントを評価します。設計レビューは、開発ライフサイクルの早い段階でセキュリティ問題を特定するのに役立ちます。これにより、最も費用対効果の高いアーキテクチャ上の問題に対処できます。

AWS セキュリティエージェントは、組織のセキュリティ要件に照らして設計ドキュメントを分析し、実装を開始する前にセキュリティ体制を改善するための詳細なセキュリティ検出結果を提供します。

この手順では、分析用の設計ファイルをアップロードして設計レビューを作成します。

前提条件

開始する前に、以下があることを確認してください。

- AWS Security Agent ウェブアプリケーションへのアクセス
- アップロードできる設計ドキュメント (DOC、DOCX、JPEG、MD、PDF、PNG、TXT)
- 各ファイルは 2MB 以下で、すべてのファイルの合計は 6MB である必要があります。
- 組織でどのセキュリティ要件が有効になっているかを理解する

ステップ 1: 設計レビューの作成を開始する

エージェントウェブアプリの設計レビューの作成ページに移動します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 設計レビューセクションに移動します。
3. 設計レビューの作成 を選択します。

> **Tip**
>
> 組織の有効なセキュリティ要件を表示するには、AWS セキュリティエージェントコンソールのセキュリティ要件ページに移動します。有効な要件を選択して、その詳細を表示します。これらの要件は、設計ファイルの分析に使用されます。

ステップ 2: 設計レビューに名前を付ける

この設計レビューの目的と範囲を特定するのに役立つわかりやすい名前を付けます。

1. 「設計レビュー名」セクションで、「名前」フィールドを見つけます。
2. 設計レビューのわかりやすい名前を入力します。

> **Note**
>
> 名前は、レビュー対象のプロジェクト、機能、またはコンポーネントを明確に識別する必要があります。最大 80 文字。

ステップ 3: 設計ファイルをアップロードする

AWS セキュリティエージェントがセキュリティコンプライアンスについて分析する設計ドキュメントをアップロードします。

1. レビューするファイルセクションで、ファイルの要件を確認します。

> **Important**
>
> 設計レビューごとに最大 5 つのファイルをアップロードできます。各ファイルは 2MB 以下で、すべてのファイルの合計は 6MB である必要があります。サポートされている形式: DOC、DOCX、JPEG、MD、PDF、PNG、TXT。

2. 次のいずれかの方法を使用してファイルをアップロードします。
   a. ドラッグアンドドロップ – ファイルをファイルドロップゾーン領域に直接ドラッグします。
   b. 参照 – ファイルの選択ボタンを使用して、コンピュータからファイルを参照して選択します。
3. 必要なファイルがすべてアップロードされていることを確認します。

> **Tip**
>
> 最良の結果を得るには、システムのセキュリティ関連のコンポーネントとデータフローを説明するアーキテクチャ図、設計仕様、技術ドキュメントを含めます。

ステップ 4: 設計レビューを開始する

必要な情報をすべて設定したら、設計ドキュメントのセキュリティ分析を開始します。

1. アップロードされたすべてのファイルと設定を確認して、正確性を確認します。
2. 設計レビューの開始 を選択します。
3. AWS セキュリティエージェントは、有効なセキュリティ要件に照らして設計ドキュメントを分析します。

> **Note**
>
> 設計レビュープロセスは通常、アップロードされたファイルの数とサイズに応じて数分で完了します。組織のセキュリティ要件に基づいてセキュリティ検出結果を受け取ります。

次の手順

設計レビューを開始した後:

- エージェントウェブアプリでレビューの進行状況をモニタリングする
- セキュリティ調査結果を確認する
- 開発チームと結果を共有する
- 設計で特定されたセキュリティ検出結果に対処する
- 設計ドキュメントを更新し、必要に応じて再送信する

設計レビューの結果を確認する

検出結果を確認すると、どのセキュリティ要件が満たされているか、どのセキュリティ要件に注意が必要か、および実装を開始する前に設計のセキュリティ体制を改善するために実行するアクションを理解するのに役立ちます。

この手順では、設計レビューの検出結果にアクセスしてフィルタリングし、解釈して、セキュリティの検出結果に効果的に対処する方法について説明します。

前提条件

開始する前に、以下があることを確認してください。

- 完了した設計レビュー
- AWS Security Agent ウェブアプリケーションへのアクセス
- 組織の有効なセキュリティ要件に精通していること

ステップ 1: 設計レビューにアクセスする

設計レビューに移動して、結果と概要情報を表示します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 設計レビューセクションに移動します。
3. リストから、調査する設計レビューを選択します。

> **Tip**
> 設計レビューの詳細ページには、レビューステータス、完了日、レビューされたファイル数の概要が表示されます。

ステップ 2: 結果の概要を確認する

大まかな概要を調べて、設計の全体的なセキュリティ体制を理解します。

1. 概要セクションを見つけます。
2. コンプライアンスステータスカテゴリごとのカウントを確認します。準拠、非準拠、データ不足、該当なし。

> **Note**
>
> 概要には各ステータスタイプのカウントが表示され、注意が必要な検出結果の数をすばやく評価できます。各ステータスの詳細な説明については、「ステップ 4」を参照してください。

ステップ 3: 結果をフィルタリングしてナビゲートする

フィルタリングおよび検索機能を使用して、特定の検出結果またはコンプライアンスステータスに焦点を当てます。

1. 結果の確認セクションで、フィルターコントロールを見つけます。
2. ステータスでフィルタリングするには:
   a. ステータスドロップダウンメニューを選択します。
   b. 特定のコンプライアンスステータスを選択すると、そのステータスの結果のみが表示されます。
3. 特定のセキュリティ要件を検索するには:
   a. 検索フィールドにキーワードを入力します。
   b. 入力すると結果が自動的に更新されます。
4. ページ分割コントロールを使用して、複数の検出結果ページ間を移動します。

> **Tip**
>
> まず非準拠ステータスでフィルタリングして、設計ですぐに注意が必要な検出結果を優先します。

ステップ 4: コンプライアンスステータスを理解する

各検出結果には、設計が特定のセキュリティ要件にどのように対応するかを示すコンプライアンスステータスが表示されます。

- 準拠 — 設計が分析に基づいてセキュリティ要件を満たしている
- 非準拠 – 設計がセキュリティ要件に違反しているか、適切に対処していない

- データが不十分 – アップロードされたファイルには、コンプライアンスを判断するのに十分な情報がありません
- 該当なし – セキュリティ要件はシステム設計には適用されません

> **Important**
>
> 非準拠および不十分なデータステータスにはアクションが必要なため、それらに焦点を当てます。設計を更新して非準拠の検出結果に対処し、追加の設計ドキュメントをアップロードして不十分なデータの検出結果を解決します。

ステップ 5: 結果の詳細を表示する

個別の検出結果を選択して、詳細な根拠と修復ガイダンスを表示します。

1. 検出結果テーブルで、セキュリティ要件名を選択します。
2. 以下を含む検出結果の詳細を確認します。
   - 評価される特定のセキュリティ要件
   - 検出結果がコンプライアンスステータスを受信した理由を説明するコメント。欠落しているものや非準拠のものに関する具体的な詳細が含まれます。
   - 検出結果に対処するための推奨修復ガイダンス
   - セキュリティ要件に関する組織の内部ドキュメントまたは標準へのリンク

> **Note**
>
> コメントは、AWS セキュリティエージェントの推論を特定の詳細で説明しています。不十分なデータ検出結果の場合、コメントは「設計ドキュメントは認証メカニズムに言及していません」や「保管中のデータ暗号化に関する情報が見つかりません」など、欠落している情報を特定します。

ステップ 6: 検出結果に対処する

設計のセキュリティ体制を改善するために注意が必要な検出結果に対してアクションを実行します。

非準拠の検出結果の場合:

1. 推奨される修復ガイダンスを確認してください。
2. リンクされた内部ドキュメントで追加のコンテキストを確認します。
3. セキュリティ要件に対応するために設計ドキュメントを更新します。
4. 今後の参照用に行った変更を文書化します。

データ検出結果が不十分の場合:

1. コメントをよく読み、欠落している特定の情報を理解してください。
2. 欠落している詳細を含む設計ドキュメントを作成または更新します。
3. 更新されたファイルを再送信する準備をします。

次の手順

設計結果を確認した後:

- チームと共有するための CSV ファイルとして検出結果レポートをダウンロードする
- 設計ドキュメントを更新して非準拠の検出結果に対処する
- データ不足の検出結果に関する追加ドキュメントを作成する
- 開発チームと結果を共有して議論する
- この設計レビューのクローンを作成して、元のドキュメントがプリロードされた新しいレビューを作成します。これにより、名前を更新し、分析を再実行して改善点を検証できます。
- コンプライアンス要件を満たす設計の実装に進む

セキュリティ要件の管理の詳細については、「」を参照してくださいthe section called “セキュリティ要件を管理する”。

設計レビューの作成の詳細については、「」を参照してくださいthe section called “設計レビューを作成する”。

脅威モデルを作成する

AWS Security Agent ウェブアプリケーションに脅威モデルを作成して、アプリケーションのアーキテクチャを分析し、セキュリティの脅威を特定します。脅威モデルは 2 つの主な出力を生成します。1 つはシステムの構築と動作を説明するシステム概要、もう 1 つはシステムの攻撃方法を説明する一連の脅威で、それぞれに重要度レベル、STRIDE 分類、実用的な推奨事項があります。

脅威モデルは 2 種類の入力を受け入れ、どちらかまたは両方を指定できます。

- スコープドキュメント – 脅威モデルが焦点を当てる内容を定義する技術設計ドキュメント、API 仕様、またはアーキテクチャドキュメント。指定した場合、エージェントはこれらのドキュメントで説明されているコンテキストに分析の範囲を絞り込みます。ファイル (DOC、DOCX、JPEG、MD、PDF、PNG、TXT) をアップロードしたり、S3 URIs を指定したり、統合を通じて Confluence ページを接続したりできます。
- ソース – 接続されたリポジトリ (GitHub、GitHub Enterprise Server、GitLab、GitLab Self-Managed、または Bitbucket) または S3 の場所からのソースコード。AWS セキュリティエージェントはソースコードを読み取り、既存のシステムを理解します。スコープドキュメントと組み合わせると、ソースコードは現在の実装に関するコンテキストを提供します。

この手順では、入力とアクセス許可を設定して脅威モデルを作成し、実行を開始します。

前提条件

開始する前に、以下があることを確認してください。

- AWS Security Agent ウェブアプリケーションへのアクセス
- 少なくとも次のいずれか。
  - ソースとして使用する接続されたリポジトリ (GitHub、GitHub Enterprise Server、GitLab、GitLab Self-Managed、または Bitbucket) (「」を参照the section called “脅威モデリングを有効にする”)
  - ソースコードを含む S3 バケット
  - スコープドキュメントまたは Confluence 統合としてアップロードするドキュメントを設計する

> **Tip**
>
> エージェントスペースに接続されているリポジトリまたは S3 バケットがすでにある場合 (コードレビューや侵入テストのセットアップなど)、脅威モデルをすぐに作成できます。追加のセットアップは必要ありません。

AWS Security Agent が脅威モデルを実行する方法

入力によって、AWS セキュリティエージェントが脅威モデルを実行する方法が決まります。シナリオは 3 つあります。

指定した入力	AWS セキュリティエージェントが脅威モデルを実行する方法
ソースのみ (ソースコード)	AWS セキュリティエージェントは、ソースコードを分析してアプリケーションの構造を理解し、コンポーネントを分類し、データフローを抽出し、脅威モデルを構築し、システムの実際の実装方法に基づいて脅威を特定します。
スコープドキュメントのみ (設計ドキュメント)	AWS セキュリティエージェントは、ドキュメントで説明されている設計に焦点を当てた脅威モデルを実行します。スコープドキュメントで説明されているアーキテクチャ、データフロー、コンポーネントに基づいて脅威を識別します。これは、コードが存在する前に設計を脅威モデリングするのに役立ちます。
ソースドキュメントとスコープドキュメントの両方	AWS セキュリティエージェントは、スコープドキュメントで説明されているコンテキスト (新機能の設計ドキュメントなど) に脅威モデルをスコープします。ソースコードを使用して既存のシステムを理解し、その設計が既に実装されているかどうかにかかわらず、スコープドキュメントで説明されている設計を脅威モデル化します。

脅威モデルページにアクセスする

ウェブアプリケーションの脅威モデリングセクションに移動します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 左側のサイドバーで、脅威モデルを選択します。
3. 既存の脅威モデルのリストと、そのタイトル、最新の実行ステータス、重要度別の脅威数が表示されます。

脅威モデルを作成する

入力とアクセス許可を設定して、新しい脅威モデルを設定します。

1. 脅威モデルページで、脅威モデルの作成を選択します。

脅威モデルの詳細を設定する

脅威モデルのタイトルを指定します。

1. タイトルフィールドに、脅威モデルのわかりやすい名前を入力します。

> **Tip**
>
> モデル化されるアプリケーションまたはサービスを識別する名前を使用します。たとえば、「billing-service」または「checkout-api」です。

スコープドキュメントを追加する

アーキテクチャドキュメント、API 仕様、設計提案など、脅威モデルが何に焦点を当てているかを定義する技術設計ドキュメントを提供します。スコープドキュメントが提供されると、エージェントはその分析をこれらのドキュメントで説明されているコンテキストにスコープします。ソースを指定する場合、スコープドキュメントはオプションです。

1. スコープドキュメントセクションで、次の 1 つ以上の方法を使用してドキュメントを追加します。
   - ファイルのアップロード – 設計ドキュメントを直接アップロードします (DOC、DOCX、JPEG、MD、PDF、PNG、TXT)。
   - S3 URIs 接続された S3 バケットに保存されているドキュメントを指す S3 URIs を入力します。
   - Confluence ページ – Confluence 統合がエージェントスペースに接続されている場合は、Confluence ワークスペースからページを選択します。

> **Tip**
>
> 最良の結果を得るには、システムのコンポーネント、データフロー、信頼境界を説明するアーキテクチャ図、API 仕様、技術ドキュメントを含めます。

ソースの追加

AWS セキュリティエージェントが既存のシステムを理解するために使用するソースコードを選択します。スコープドキュメントと組み合わせると、ソースコードは現在の実装に関するコンテキストを提供します。エージェントはそれを使用して、すでに存在するものを理解します。スコープドキュメントを指定する場合、ソースはオプションです。

1. ソースセクションで、次の 1 つ以上の方法を使用してソースコードを追加します。

統合されたリポジトリ

エージェントスペースに接続されているリポジトリ (GitHub、GitHub Enterprise Server、GitLab、GitLab Self-Managed、または Bitbucket) から選択します。

1. ソースとして含める各リポジトリの横にあるチェックボックスをオンにします。
2. 検索フィールドを使用して、特定のリポジトリを名前で検索します。

> **Note**
>
> エージェントスペースに接続されているリポジトリのみがここに表示されます。リポジトリを追加するには、管理者にエージェントスペース設定の更新を依頼してください。

S3 ソース

接続された S3 URIs を追加します。 S3

1. 含めるソースコードの場所ごとに S3 URI を入力します。

AWS リソースを設定する

この脅威モデルの IAM サービスロールとオプションの CloudWatch ロググループを選択します。

1. サービスロールドロップダウンで、設定したサービスロールから IAM ロールを選択します。

> **Note**
>
> サービスロールには、S3 のソースコードにアクセスして CloudWatch ログに書き込むためのアクセス許可が必要です。サービスロールは、AWS マネジメントコンソールのエージェントスペースのセットアップ中に設定されます。

2. (オプション) ロググループドロップダウンで、CloudWatch ロググループを選択して脅威モデルの実行ログを保存します。

脅威モデルを作成する

1. の設定を確認します。
2. 「脅威モデルの作成」を選択します。

実行を開始できる脅威モデルの詳細ページにリダイレクトされます。

脅威モデルを実行する

脅威モデルを作成したら、実行を開始して分析を開始します。

1. 脅威モデルの詳細ページで、実行の開始を選択します。
2. AWS セキュリティエージェントは、ソースとスコープのドキュメントの分析を開始します。

実行する脅威モデルの横にある実行の開始を選択して、脅威モデルリストページから実行を開始することもできます。

脅威モデルの実行をモニタリングする

実行時に脅威モデルの進行状況を追跡します。

実行ステータス

実行ページヘッダーにステータスインジケータが表示されます。

- 進行中 – 脅威モデルエージェントが実行中です。
- Stopping – キャンセルがリクエストされました。エージェントは停止する前に現在のタスクを終了しています。

- 完了 – 実行は正常に終了しました。
- 失敗 – 実行でエラーが発生しました。エラーメッセージが詳細とともに表示されます。問題を解決した後、新しい実行を開始します。
- Stopped – 実行はユーザーによってキャンセルされました。停止前に生成された脅威は保持されます。

ページタブを実行する

実行の詳細ページで、タブ間を移動します。

- 概要 – 実行の概要 (ジョブ ID、開始時刻、ステータス、期間) を重要度レベルの円グラフで表示し、重要度 (重大、高、中、低) 別の脅威の内訳を示します。概要の下に、各 STRIDE カテゴリに分類される脅威の数を示す STRIDE 脅威カテゴリの棒グラフを表示します。実行が完了したら、エージェントによって生成されたシステムの概要を表示します。
- 設定 – この実行に使用されたソース、ドキュメント、アクセス許可 (サービスロール、CloudWatch ロググループ) を表示します。
- プリフライト – インフラストラクチャのセットアップのログ記録、S3 ソースアクセスの検証 (該当する場合) 、環境のセットアップのテストなど、脅威分析が開始される前に実行されるプリフライトチェックのステータスを表示します。進行状況バーには、完了したチェックの数が表示されます。
- ログ – 実行中にエージェントが実行したタスクのフィルタリング可能なリストを表示します。タスクを選択すると、詳細なログ出力がサイドパネルに表示されます。
- 脅威 – 実行中に特定された脅威を表示します (「」を参照the section called “脅威モデルからの脅威を確認する”)。

実行履歴

各脅威モデルは、すべての実行の履歴を保持します。脅威モデルの詳細ページで、次の操作を行います。

- 実行テーブルには、開始時刻、ステータス、期間、脅威数、ID を含むすべての実行が一覧表示されます。
- 実行の開始時刻リンクを選択すると、詳細が表示されます。

> **Tip**
>
> ソースコードを更新した後、またはスコープドキュメントを修正した後、同じ脅威モデルを再実行して、システムの概要と脅威が時間の経過とともにどのように変化するかを確認します。

脅威モデルを編集する

脅威モデルの設定を変更するには:

1. 脅威モデルの詳細ページで、編集 を選択します。
2. 必要に応じて、タイトル、ソース、スコープドキュメント、または AWS リソースを更新します。
3. [Save changes] (変更の保存) をクリックします。

> **Note**
>
> 脅威モデルを編集しても、以前に完了した実行には影響しません。これらは、実行時に使用された設定スナップショットを保持します。

脅威モデルを削除する

脅威モデルとそれに関連するすべての実行と脅威を削除するには:

1. 脅威モデルの詳細ページで、アクションメニューを開き、削除を選択します。
2. 削除を確定します。

> **Important**
>
> 実行が現在進行中の場合は、脅威モデルを削除する前に実行を停止する必要があります。

次の手順

脅威モデルを実行した後:

- システムの概要と脅威を確認する (「」を参照the section called “脅威モデルからの脅威を確認する”)
- 変更を加えた後に脅威モデルを再実行して、脅威が対処されていることを確認する
- アプリケーションの進化に合わせてソースとスコープのドキュメントを調整する

脅威モデルからの脅威を確認する

脅威モデルの実行が完了したら、システムの概要と脅威を確認して、アプリケーションの攻撃方法と対処方法を理解します。システムの概要は、アプリケーションのアーキテクチャ、信頼境界、データフロー、セキュリティ体制を説明する包括的なドキュメントです。各脅威には、ステートメント、重要度レベル、STRIDE 分類、影響を受けるアセット、それに対処するための推奨事項が含まれます。

前提条件

開始する前に、以下があることを確認してください。

- 完了した脅威モデルの実行
- AWS Security Agent ウェブアプリケーションへのアクセス

ステップ 1: 脅威モデルの実行にアクセスする

完了した脅威モデルの実行に移動します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 左側のサイドバーで、脅威モデルを選択します。
3. 調査する脅威モデルを選択します。
4. 実行テーブルで、開始時刻リンクを選択して、完了した実行を選択します。

ステップ 2: システムの概要を確認する

システムの概要は、AWS セキュリティエージェントがシステムを理解する方法の包括的な説明です。これは、以下を含めることができる構造化ドキュメントです。

- 目的 – システムが何をし、誰にサービスを提供するか。
- 機能 – システムが提供する主要な機能。

- 設計インテント – 脅威モデル化されている設計変更または機能 (スコープドキュメントが提供される場合) 。
- アーキテクチャ – デプロイパターンや通信プロトコルなど、システムの構築方法。
- コンポーネント – 目的と主要なやり取りを含むシステムコンポーネントのテーブル。
- 信頼境界 — 各交差に存在する保護など、セキュリティコンテキストが変化する場所。
- データフロー – 各ステップでのプロトコル、認証情報、保護など、データがどのようにシステムを移動するかの詳細な説明。
- セキュリティ体制 — 現在の認証、暗号化、アクセスコントロールのメカニズム。
- 機密アセット – 分類ポイントと露出ポイントを含む、保護が必要なデータと認証情報。
- 主な前提 — エージェントがシステムに関して行ったセキュリティ関連の前提。

システムの概要を確認するには:

1. 概要タブを選択します。
2. Run summary セクションを確認します。このセクションには、ジョブ ID、開始時刻、ステータス、期間が表示されます。
3. 重要度レベルのグラフを確認します。このグラフには、重要度 (重大、高、中、低) 別の脅威の内訳と、数と割合が表示されます。
4. 各 STRIDE カテゴリ (なりすまし、改ざん、否認、情報開示、サービス拒否、特権昇格) に該当する脅威の数を示す脅威カテゴリチャートを確認します。
5. システムの概要セクションで、エージェントの完全な分析をお読みください。

> **Tip**
>
> システムの概要がシステムを正確に反映していない場合は、入力を絞り込み、関連するリポジトリをソースとして追加するか、より完全なスコープドキュメントをアップロードして、脅威モデルを再実行します。

ステップ 3: 脅威を確認する

Threats タブに移動して、実行中に特定されたすべての脅威を表示します。

1. 脅威タブを選択します。

2. 脅威はリストとして表示され、各カードには脅威のタイトル、重要度バッジ、ステータス、最後に更新されたタイムスタンプが表示されます。脅威を重要度、ステータス、またはタイトルでフィルタリングできます。
3. リストから脅威を選択すると、右側のパネルにその詳細が表示されます。

脅威の重大度

各脅威には重要度レベルが割り当てられます。

- 重大 – 即時のアクションが必要です。悪用すると、システムが完全に侵害される可能性があります。
- 高 – 迅速な注意が必要です。悪用すると、セキュリティに大きな影響を与える可能性があります。
- 中 – 妥当な期間内に対処する必要があります。全体的なセキュリティリスクに寄与します。
- 低 – 定期的なメンテナンスの一環として対処でき、即時リスクを最小限に抑えます。

脅威の詳細

脅威を選択すると、右側のパネルにその詳細が表示されます。詳細については、以下のセクションにまとめられています。

メタデータ行:

- 重要度 – エージェントによって割り当てられたリスクレベル (重大、高、中、低) 。
- STRIDE カテゴリ – 脅威分類: なりすまし、改ざん、否認、情報開示、サービス拒否、または特権の昇格。
- 作成日時 – 脅威が特定された日時。
- 最終更新日 – 脅威が最後に変更された日時。

説明セクション:

- ステートメント – 脅威の自然言語の説明。脅威のソースができること、影響、それを可能にする条件。
- ソース – 脅威のアクターまたはオリジン (たとえば、「認証されたユーザー」または「外部の攻撃者」) 。

- アクション – 脅威ソースができること (たとえば、「検索パラメータに SQL クエリを挿入する」) 。
- 影響 – 脅威アクションの直接的な結果 (「顧客レコードへの不正アクセス」など) 。

リファレンスセクション:

- 影響を受けるアセット – 脅威の影響を受ける特定のアセット (「顧客支払いレコード」や DynamoDB テーブル」など) 。
- 前提条件 – 脅威を悪用可能にするために当てはまる必要がある条件。
- 影響を受ける目標 – 影響を受けるセキュリティ目標: 機密性、完全性、可用性、認可、認証、拒否の禁止。
- 証拠 – 脅威をサポートするソースコードファイルパス。リポジトリ内の特定のファイルにリンクします。
- アンカー – 脅威をソースコード内の特定のノードにリンクするコードリファレンス。

推奨事項セクション:

- 推奨事項 — 脅威に対処するための実用的なガイダンス。

ステップ 4: 脅威を手動で作成する

エージェントが特定しなかった脅威を追加できます。例えば、手動レビュー中に検出された脅威や外部ソースから検出された脅威などです。

1. 完了した実行の脅威タブで、脅威の作成を選択します。
2. 脅威の詳細を入力します。
   - ステートメント – 脅威の自然言語の説明。
   - 重要度 – 重大、高、中、または低を選択します。
   - ソース – 脅威のアクターまたはオリジン。
   - 前提条件 – 脅威が悪用可能になるために必要な条件。
   - アクション – 脅威ソースができること。
   - 影響 – 脅威アクションの直接的な結果。
   - 影響を受けるアセット – 影響を受ける特定のアセット (カンマ区切り) 。

- 影響を受けるセキュリティ目標 – 機密性、完全性、可用性、認可、認証、または否認防止から選択します。
- STRIDE カテゴリ – 該当するカテゴリを選択します。
- 推奨事項 – 脅威に対処するためのガイダンス。

3. [作成] を選択します。

手動で作成された脅威は、エージェントが生成した脅威とともに脅威リストに表示されます。

ステップ 5: 脅威を編集してトリアージする

脅威を確認するときは、詳細を編集し、ステータスを更新して進捗状況を追跡できます。

1. リストから脅威を選択します。
2. 脅威の詳細パネルの編集アイコンを選択して、編集フォームを開きます。
3. 次のフィールドを変更できます。
   - ステータス – 脅威のライフサイクルを追跡します。
     - オープン — 脅威が確認され、注意が必要です (デフォルト)。
     - 解決済み – 問題を修正しました。
     - Dismissed – 脅威を確認し、該当しないと判断しました。
   - 重要度 – エージェントの評価がコンテキストと一致しない場合は、重要度レベルを調整します。
   - ステートメント – 脅威の説明を絞り込みます。
   - ソース、前提条件、アクション、影響 – ドメインの知識に基づいて脅威の詳細を更新します。
   - 影響を受けるアセット – 影響を受けるアセットを追加または削除します (カンマ区切り)。
   - 影響を受けるセキュリティ目標 – 影響を受けるセキュリティ目標 (機密性、整合性、可用性、認可、認証、否認防止) を選択します。
4. [適用] を選択して変更を保存します。

ステップ 6: レポートをダウンロードする

実行が完了したら、システムの概要と特定されたすべての脅威をまとめた PDF レポートをダウンロードできます。

1. 完了した実行ページで、レポートの生成を選択します。

2. PDF がコンピュータにダウンロードされます。

ステップ 7: プリフライトチェックとログを確認する

エージェントがどのように結論に達したかを調査するか、または部分的な障害をデバッグする必要がある場合:

1. プリフライトタブを選択すると、脅威分析が開始される前に実行されるプリフライトチェックのステータスが表示されます。各チェックにはステータス (完了、進行中、または保留中) が表示され、進行状況バーには完了したチェックの数が表示されます。完了したチェックを展開して、詳細なログ出力を表示できます。
2. ログタブを選択すると、実行中にエージェントが実行したタスクのフィルタリング可能なリストが表示されます。リストから任意のタスクを選択すると、詳細なログ出力がサイドパネルに表示されます。

次の手順

脅威モデルの結果を確認した後:

- エージェントの推奨事項に基づいて、重要度の高い脅威に最初に対処する
- 修正を実装する際に脅威のステータスを更新する
- 新しい脅威モデルを実行して、変更が特定された脅威に対処していることを確認する
- アプリケーションの進化に応じてソースとスコープのドキュメントを調整する (「」を参照the section called “脅威モデルを作成する”)

プルリクエストでコードセキュリティの検出結果を確認する

リポジトリのプルリクエストのコードレビューを有効にすると、AWS セキュリティエージェントはプルリクエストを自動的に分析し、セキュリティ検出結果をソースコントロールプロバイダーに直接投稿します。これにより、開発者はプルリクエストを離れることなく、通常のワークフロー内のセキュリティ問題に対処できます。

> **Note**
>
> このページは、GitHub プルリクエスト、GitLab マージリクエスト、Bitbucket プルリクエストに適用されます。エクスペリエンスはすべてのプロバイダーで類似しています。

プルリクエストでのコードレビューの仕組み

コードレビューが有効になっているリポジトリでプルリクエスト (または GitLab のマージリクエスト) を送信すると、AWS セキュリティエージェントは自動的に分析を開始します。

1. プルリクエスト分析トリガー - コードレビュー機能を有効にしたリポジトリでプルリクエストが「レビュー準備完了」とマークされると、コードレビューがトリガーされます。ドラフトプルリクエストは分析されません。
2. 分析確認 - AWS セキュリティエージェントがプルリクエストの分析を開始すると、「AWS セキュリティエージェントがコードを分析しています...」という最初のコメントが投稿されます。これにより、分析が開始され、進行中であることがわかります。
3. レビューの完了 - 分析が完了すると、AWS セキュリティエージェントは結果を含むレビューをプルリクエストに投稿します。すべてのセキュリティ検出結果は 1 回のレビューでまとめてバッチ処理され、プルリクエストを整理し、通知を最小限に抑えます。

コードレビュー結果について

AWS セキュリティエージェントは、分析中に検出した内容に応じてさまざまなタイプの結果を提供します。

セキュリティの問題が見つかった場合

AWS セキュリティエージェントは、コード変更でセキュリティの問題を特定した場合、以下を含むレビューを投稿します。

- 概要 - 特定された問題のタイプとその潜在的な影響を説明する、すべてのセキュリティ検出結果の概要
- 個々の検出結果 - 詳細なセキュリティ検出結果は、メインレビューでスレッドコメントとして表示され、各検出結果には以下が含まれます。
  - セキュリティ問題の説明
  - 問題が見つかったコード内の場所
  - 問題に対処する方法を説明する修復ガイダンス
  - コードレビュー設定に基づく関連コンテキスト (セキュリティ要件違反、一般的な脆弱性、またはその両方)

> **Note**
>
> 分析されるセキュリティ問題のタイプは、コードレビュー設定によって異なります。セキュリティ要件の検証を設定した場合、検出結果は組織のカスタムセキュリティ要件を参照します。セキュリティ脆弱性の検出結果を設定した場合、検出結果は一般的なセキュリティの脆弱性を特定します。コードレビュー設定の詳細については、「」を参照してくださいthe section called “GitHub リポジトリのプルリクエストコードレビューを有効にする”。

セキュリティ上の問題が見つからない場合

AWS Security Agent が分析を完了し、コードの変更でセキュリティの問題が見つからない場合は、「問題は特定されていません」というコメントを投稿します。これにより、レビューが正常に終了し、コードの変更によって、設定したコードレビュー設定に基づいてセキュリティ検出結果がトリガーされなかったことが確認されます。

セキュリティ検出結果への対応

AWS Security Agent によって投稿されたセキュリティ検出結果を確認したら、ソースコントロールプロバイダーで直接アクションを実行できます。

- 検出結果に対処する - 検出結果に記載されている修復ガイダンスに基づいてコードを更新し、新しいコミットをプルリクエストにプッシュします。AWS セキュリティエージェントは、更新されたコードを分析します。
- 会話の解決 - セキュリティ検出結果に対処したら、会話を解決済みとしてマークして進行状況を追跡します。

> **Tip**
>
> 各検出結果には、特定されたセキュリティ問題に合わせた特定の修復ガイダンスが含まれています。このガイダンスをよく読んで、セキュリティリスクとその効果的な対処方法を理解してください。

コードレビューの結果のフィルタリング

リポジトリにfiltering.mdファイルを追加することで、AWS セキュリティエージェントがコードを分析する方法をカスタマイズできます。このファイルを使用すると、コードベースに関するコンテキストを提供し、分析からファイルやフォルダを除外することで、誤検出を減らすことができます。

フィルタリングファイルの作成

リポジトリのルートにある .awssecurityagent ディレクトリfiltering.mdに という名前のファイルを作成します。

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

AWS セキュリティエージェントは、プルリクエストを分析するときに、リポジトリのメインブランチ (mainや などmainline) からこのファイルを読み取ります。

ファイル構造

このfiltering.mdファイルは、AWS セキュリティエージェントが認識する特定のセクションで標準の Markdown 形式を使用します。ファイルには、Code Review見出しの後に次のセクションの 1 つまたは両方を含める必要があります: IgnorePatternsおよび ContextHints (スペースなし) 。

次の例は、filtering.mdファイルの完全な構造を示しています。

```
# 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.
```

パターンを無視する

このセクションでは、コードレビュー中に AWS セキュリティエージェントがスキップするファイルとフォルダIgnorePatternsを指定します。を使用してglob patterns、分析から除外するパスを定義します。

フォーマット要件:

- 各パターンは独自の行にある必要があります。
- 各パターンを空の行で区切ります。これにより、GitHub またはコードレビューツールで表示したときにファイルが正しくレンダリングされます。
- パターンは標準の glob 形式に従います。たとえば、 はすべてのマークダウンファイル**/*.mdと一致し、 はルートの /myapp/src/フォルダ内のすべての.snapファイル/myapp/src/**/*.snapと一致します。
- このセクションでは、最大 1000 個の無視パターンがサポートされています。

コンテキストヒント

このセクションではContextHints、AWS セキュリティエージェントがより正確な評価を行うのに役立つコードベースに関する追加のコンテキストを提供します。コンテキストヒントを使用して、アーキテクチャ上の決定、セキュリティ例外、または検出結果の解釈方法に影響を与える可能性のあるその他の情報について説明します。

フォーマット要件:

- 各ヒントはダッシュ (-) で始まり、スペースが続く必要があります。
- 各ヒントは、500 文字に制限された自由形式のテキストの 1 行として記述します。
- 各ヒントは、コードベースに関する特定のコンテキストを 1 つ記述する必要があります。
- このセクションでは、最大 20 のコンテキストヒントをサポートしています。

コンテキストヒントは、AWS セキュリティエージェントが最初の分析を完了すると適用され、特定のユースケースに適用されない検出結果をフィルタリングするのに役立ちます。

次の手順

コードセキュリティの検出結果を確認した後:

- 修復ガイダンスに基づいてコードを更新する
- 新しいコミットをプッシュして変更の再分析をトリガーする
- 必要に応じてコードレビュー設定を調整する (「」を参照the section called “GitHub リポジトリのプルリクエストコードレビューを有効にする”)
- 組織のセキュリティ要件を確認して検証基準を理解する
- デプロイされたアプリケーションの包括的なセキュリティ検証のための侵入テストを検討する

コードレビューを作成する

AWS Security Agent ウェブアプリケーションでコードレビューを作成して、ソースコードリポジトリと S3 ソースでセキュリティの脆弱性をスキャンします。コードレビューは、コードベース全体で包括的な静的分析を実行し、セキュリティの問題を特定し、修復ガイダンスを提供します。

個々のコード変更を分析するプルリクエストベースのコードレビュー (「」を参照the section called “プルリクエストでコードセキュリティの検出結果を確認する”) とは異なり、オンデマンドコードレビューは完全なソースコードをスキャンしてセキュリティの脆弱性を特定し、組織のセキュリティ要件への準拠を検証します。

この手順では、ソースコード入力の選択、アクセス許可の設定、レビューの実行により、コードレビューを作成します。

前提条件

開始する前に、以下があることを確認してください。

- AWS Security Agent ウェブアプリケーションへのアクセス
- エージェントスペース内の少なくとも 1 つの接続された GitHub リポジトリまたは S3 バケット

> **Tip**
>
> すでに GitHub リポジトリがエージェントスペースに接続されている場合、コードレビューはすぐに使用できます。追加のセットアップは必要ありません。エージェントスペースペー

ジのコードレビューカードからウェブアプリで開始を選択するか、ウェブアプリケーションを直接起動します。
追加のソースを接続したり、S3 バケットを設定したりする必要がある場合は、「」を参照してくださいthe section called “コードレビューを有効にする”。

コードレビューページにアクセスする

ウェブアプリケーションのコードレビューセクションに移動します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 左側のサイドバーで、コードレビューを選択します。
3. ソース情報、最終実行ステータス、結果の概要を含む既存のコードレビューのリストが表示されます。

コードレビューを作成する

ソースコードの入力とアクセス許可を設定して、新しいコードレビューを設定します。

1. コードレビューページで、コードレビューの作成を選択します。

コードレビューの詳細を設定する

タイトルを指定し、確認するソースコードを選択します。

1. タイトルフィールドに、コードレビューのわかりやすい名前を入力します。

> **Tip**
> レビューのアプリケーション、リポジトリ、または範囲を識別する名前を使用します。たとえば、「billing-service-security-review」または「infrastructure-code-audit」です。

2. Sources セクションで、このレビューのソースコード入力を選択します。2 つのタブから選択します。

GitHub リポジトリ

エージェントスペースに接続されているリポジトリから選択します。

1. GitHub リポジトリタブを選択します。
2. 統合リポジトリテーブルで、レビューに含める各リポジトリの横にあるチェックボックスをオンにします。
3. 検索フィールドを使用して、特定のリポジトリを名前で検索します。

> **Note**
>
> コードレビュー設定を介してエージェントスペースに接続されたリポジトリのみがここに表示されます。リポジトリを追加するには、管理者コンソールで管理を選択するか、管理者にエージェントスペース設定の更新を依頼します。

S3 ソース

エージェントスペースに接続されている S3 バケットから ZIP ファイルを選択します。エージェントスペース管理者は、使用可能な S3 バケットを設定します。これらのバケットのいずれかに保存されている ZIP ファイルは、コードレビューのソースとして使用できます。

1. S3 ソースタブを選択します。
2. レビューに含める各 ZIP ファイルの S3 URI を入力します。最大 30 個の S3 ソースを追加できます。

> **Note**
>
> S3 ソースは、エージェントスペースに接続されている S3 バケットに保存されている ZIP ファイルである必要があります。追加のバケットを使用可能にするには、「」を参照してくださいthe section called “コードレビューを有効にする”。

アクセス許可の設定

このコードレビュー用に IAM サービスロールとオプションの CloudWatch ロググループを選択します。

1. アクセス許可セクションで、サービスロールドロップダウンを見つけます。
2. 設定したサービスロールから IAM ロールを選択します。

> **Note**
>
> サービスロールには、S3 でソースコードにアクセスし、CloudWatch ログ、およびコードレビューに必要なその他の AWS リソースに書き込むためのアクセス許可が必要です。サービスロールは、AWS マネジメントコンソールでのコードレビューのセットアップ中に設定されます。

3. (オプション) CloudWatch ロググループドロップダウンで、コードレビュー実行ログを保存するロググループを選択します。

> **Note**
>
> ロググループを選択しない場合、AWS セキュリティエージェントはコードレビューログを保存するためのデフォルトのロググループを作成します。

自動コード修復を設定する

自動修復を有効にして、レビューが完了するとすぐに AWS セキュリティエージェントがすべての検出結果のコード修正を生成できるようにします。

1. 自動コード修復セクションで、自動コード修復を有効にするチェックボックスをオンにします。

AWS セキュリティエージェントが修正を配信する方法は、ソースによって異なります。

- プライベート GitHub リポジトリ – AWS セキュリティエージェントは、修正を含むプルリクエストをリポジトリに送信します。
- パブリック GitHub リポジトリ – 修正前に脆弱性が開示されないようにするため、AWS セキュリティエージェントはプルリクエストを開きません。代わりに、ウェブアプリケーションからダウンロードしてプライベートに適用できる検出結果に、提案された差分をアタッチします。
- S3 ソース – コード修復は使用できません。検出結果の詳細を確認し、手動で修正を適用します。

> **Important**
>
> プライベートリポジトリに送信された修復プルリクエストは、読み取りアクセス権を持つすべてのユーザーに表示されます。マージする前に変更を確認します。自動コード修復は、GitHub リポジトリがソースとして選択されている場合にのみ使用できます。

> **Note**
>
> 無効にしても、レビューの完了後に GitHub がソースとする個々の検出結果のコード修復を手動でトリガーできます。

シミュレートされた検証を設定する

シミュレートされた検証を有効にして、検出された脆弱性が実行中のアプリケーションで悪用可能かどうかを動的に確認します。

1. シミュレートされた検証セクションで、シミュレートされた検証を有効にするチェックボックスをオンにします。

有効にすると、AWS セキュリティエージェントは静的分析が完了した後にシミュレートされた環境をプロビジョニングします。ソースコードをオンボードし、環境内でサービスを開始し、スキャン中に見つかった脆弱性を悪用しようとします。その後、検出結果は、脆弱性が正常に悪用されたかどうかを示す検証ステータスで更新されます。

> **Note**
>
> シミュレートされた検証は現在、自己完結型のドッカー化可能なアプリケーションでのみ使用できます。ソースとして複数のリポジトリが選択されている場合、シミュレートされた検証は使用できません。

> **Important**
>
> シミュレートされた検証により、コードレビューの実行に処理時間が追加されます。検証ステップは環境をプロビジョニングし、悪用の試みを実行します。これには、検出結果の数とアプリケーションの複雑さに応じて、通常 1～3 時間かかります。

許可されたネットワーク送信先

シミュレートされた環境には、事前定義された許可リストに制限されたアウトバウンドネットワークアクセスがあります。アプリケーションは、検証中に次のドメインに到達できます。

次の表に、許可されたドメインとその目的を示します。

ドメイン	目的
`.amazonaws.com` , `.aws.amazon.com`	AWS サービス
`.public.ecr.aws`	Amazon ECR Public
`.docker.com` , `.docker.io`	Docker Hub
`.github.com` , `.githubusercontent.com`	GitHub
`.gitlab.com`	GitLab
`.npmjs.com` , `.npmjs.org`	npm レジストリ
`.pypi.org` , `.pypi.python.org` , `.pythonhosted.org`	Python パッケージインデックス
`.crates.io` , `.rustup.rs`	Rust パッケージ

ドメイン	目的
.maven.org , .gradle.org	Java/Gradle パッケージ
.nuget.org	.NET パッケージ
.rubygems.org , .ruby-lang.org	Ruby パッケージ
.golang.org , .pkg.go.dev , .goproxy.io	Go パッケージ
.nodejs.org , .yarnpkg.com	Node.js
.alpinelinux.org , .debian.org , .ubuntu.com , .centos.org , .fedoraproject.org	Linux ディストリビューションリポジトリ
.cloudfront.net	CloudFront ディストリビューション
.google.com , .googleapis.com	Google APIs
.microsoft.com , .visualstudio.com	Microsoft のサービス
.sourceforge.net , .bitbucket.org	ソースホスティング

> **Note**
>
> アプリケーションがこのリストにないドメインへのネットワークアクセスを必要とする場合、接続はネットワークファイアウォールによってブロックされます。外部 APIsまたは上記に記載されていないサービスに依存するアプリケーションは、シミュレートされた環境で正しく起動しない場合があります。

コードレビューを作成する

1. 設定を確認して、精度を確認します。
2. コードレビューの作成 を選択します。

コードレビューの詳細ページにリダイレクトされ、レビュー実行を開始できます。

コードレビューを実行する

コードレビューを作成したら、実行を開始して分析を開始します。

1. コードレビューの詳細ページで、レビューの開始を選択します。
2. AWS セキュリティエージェントがソースコードの分析を開始します。

実行するコードレビューの横にあるレビューの開始を選択して、コードレビューリストページからレビューを開始することもできます。

コードレビューの実行をモニタリングする

コードレビューの実行の進行状況を追跡します。

実行フェーズを確認する

コードレビューの実行は、次のフェーズを通じて進行し、進行状況インジケータとして表示されます。

1. プリフライト – AWS セキュリティエージェントはソースコードへのアクセスを検証し、テスト環境を設定します。プリフライトチェックには以下が含まれます。
   - サービスインフラストラクチャのセットアップ
   - S3 ソースアクセスの検証
   - テスト環境のセットアップ
2. 静的分析 – AWS セキュリティエージェントは、ソースコードをスキャンして、セキュリティの脆弱性と要件違反がないか確認します。
3. シミュレートされた検証 (オプション) – シミュレートされた検証が有効になっている場合、AWS セキュリティエージェントはシミュレートされた環境をプロビジョニングし、アプリケーションをデプロイします。次に、静的分析中に検出された脆弱性を悪用しようとします。このステップ

では、結果がターゲットランタイムで悪用可能かどうかを確認します。このフェーズは、コードレビューの作成中にシミュレートされた検証を有効にした場合にのみ表示されます。

4. 確定中 – AWS セキュリティエージェントは検出結果をコンパイルし、結果の概要を生成します。

実行の詳細を表示する

実行の詳細ページで、タブ間を移動して進行状況をモニタリングします。

- コードレビューの実行 – 実行 ID、作成時刻、ステータス、期間、タスク時間、重要度レベルの内訳、リスクタイプのグラフを含む実行の概要を表示します。
- プリフライト – 各検証ステップのプリフライトチェックの進行状況とステータスを表示します。
- コードレビューログ – レビュー中に AWS セキュリティエージェントが特定および実行したタスクと、各ステップの詳細なタスクログを表示します。
- シミュレートされた検証 – シミュレートされた検証が有効になっている場合は、プロビジョニングステータス、個々の検出結果の検証タスク、およびその結果を表示します。
- 結果 – レビューの完了後にセキュリティ結果を表示します (「」を参照the section called “コードレビューの結果を確認する”)。

実行履歴

各コードレビューでは、すべての実行の履歴が保持されます。コードレビューの詳細ページで、次の操作を行います。

- 最新の実行セクションには、最新の実行とその開始時刻、ステータス、期間、ID が表示されます。
- すべての実行テーブルには、開始時刻、ステータス、期間、結果の概要、ID を含む以前の実行がすべて一覧表示されます。
- Monitor run を選択して、最新のアクティブな実行の詳細を表示します。
- 実行の開始時刻リンクを選択すると、詳細が表示されます。

次の手順

コードレビューを実行した後:

- セキュリティ検出結果とその修復ガイダンスを確認する (「」を参照the section called “コードレビューの結果を確認する”)

- 自動プルリクエストまたは手動修正による検出結果の修正 (「」を参照the section called “コードレビューの検出結果の修正”)
- 修正を実装した後に追加のレビューを実行して修正を検証する
- コードベースの進化に合わせてコードレビュー設定またはソースを調整する

コードレビューの結果を確認する

コードレビューの実行が完了したら、実行の概要とセキュリティの検出結果を確認して、ソースコードの脆弱性を理解します。各検出結果には、セキュリティ問題の優先順位付けと対処に役立つ説明、重要度評価、コードの場所、リスクの推論、推奨される修正が含まれています。

前提条件

開始する前に、以下があることを確認してください。

- 完了または進行中のコードレビューの実行
- AWS Security Agent ウェブアプリケーションへのアクセス

ステップ 1: コードレビューの実行にアクセスする

完了したコードレビューの実行に移動して、概要と検出結果を表示します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 左側のサイドバーで、コードレビューを選択します。
3. 調査するコードレビューを選択します。
4. すべての実行 テーブルで、開始時刻のリンクをクリックして、完了した実行を選択します。または、Monitor run を選択して最新の実行を表示します。

ステップ 2: 実行の進行状況をモニタリングする

ステップインジケータを使用して、コードレビューの実行の進行状況を追跡します。

1. ページヘッダーの下にある水平ステップインジケータを見つけます。
2. 各フェーズのステータスを確認します。

- プリフライト – ソースコードへのアクセスを検証し、スキャン環境を設定します。チェックには、サービスインフラストラクチャのセットアップ、S3 ソースアクセスの検証、GitHub アクセスチェックを含むテスト環境のセットアップが含まれます。
- 静的分析 – ソースコードをスキャンして、セキュリティの脆弱性と要件違反がないか確認します。
- シミュレートされた検証 (オプション) – 有効にすると、 はシミュレートされた環境をプロビジョニングし、検出された脆弱性を実行中のアプリケーションに対して悪用しようとします。
- Finalizing – 結果をコンパイルし、結果の概要を生成します。

> **Note**
>
> 各ステップには、ステータスインジケータ (完了または進行中) が表示されます。すべてのフェーズで完了と表示されると、実行は完了です。シミュレートされた検証ステップは、コードレビューの作成中にシミュレートされた検証を有効にした場合にのみ表示されます。

ステップ 3: 実行の概要を確認する

コードレビューの実行タブに移動して、高レベルの結果を表示します。

1. コードレビューの実行タブを選択します。
2. Run summary セクションには、実行ステータス、期間、その他の高レベルの詳細が表示されます。また、重要度レベルとリスクタイプ別に分類されたセキュリティ検出結果のダッシュボードも提供します。
3. AWS セキュリティエージェントによるアプリケーションの概要は、実行が完了したときにコードレビュースキャンの概要を提供します。

ステップ 4: プリフライト結果を確認する

プリフライトタブに移動して、すべてのソースアクセスチェックに合格したことを確認します。

1. プリフライトタブを選択します。
2. 完了したチェックの数を示すプリフライト進行状況インジケータを確認します。
3. 各チェックに成功ステータスが表示されていることを確認します。

- サービスインフラストラクチャのセットアップ — テスト環境の準備が整っていることを確認します
- S3 ソースアクセスの検証 – S3 ソースコードへのアクセスを確認します (該当する場合)
- テスト環境のセットアップ – 分析環境が設定されていることを確認します

> **Note**
>
> プリフライトチェックが失敗した場合、実行は静的分析に進みません。チェックの詳細を確認してアクセスの問題を特定して解決し、新しい実行を開始します。

ステップ 5: コードレビューログを確認する

コードレビューログタブに移動して、分析中に AWS セキュリティエージェントが実行したタスクを表示します。

1. コードレビューログタブを選択します。
2. 左パネルのタスクのリストを参照します。
3. タスクを選択すると、右側のパネルに詳細なログが表示されます。

> **Tip**
>
> 検索フィールドを使用して、名前でタスクをフィルタリングします。ログは、AWS セキュリティエージェントが調査した内容とその結論にどのように達したかについてのインサイトを提供します。

ステップ 6: 検出結果に移動する

結果タブを選択して、実行のすべてのセキュリティ結果を表示します。

> **Note**
>
> デフォルトの信頼度フィルター

デフォルトでは、エージェントの信頼度が高い結果のみが表示されます。エージェントの信頼度が中または低の検出結果と誤検出を表示するには、未検証の検出結果の非表示トグルをオフにします。

1. 結果タブを選択します。
2. 検出結果は分割ビューに表示され、左側に検出結果リスト、右側に選択した検出結果の詳細が表示されます。
3. 各検出結果カードに表示される情報を確認します。
   - 検出結果のタイトル — セキュリティ上の問題をまとめたわかりやすい名前
   - 重要度バッジ – 色分けされた重要度インジケータ:
     - 重大 (赤) – 即時のアクションが必要。悪用するとシステム侵害につながる可能性がある
     - 高 (赤) – 迅速な注意が必要です。悪用すると、セキュリティに大きな影響を与える可能性があります
     - 中程度 (オレンジ) – 妥当な期間内に対処する必要があります。全体的なセキュリティリスクに寄与します
     - 低 (黄色) – 定期的なメンテナンスの一環として対処でき、即時リスクを最小限に抑える
   - 最終更新日 — 検出結果が最後に更新された時刻のタイムスタンプ

ステップ 7: 結果の詳細を確認する

個々の検出結果を選択して、各脆弱性に関する包括的な情報を表示します。

1. 左側のパネルで結果を選択すると、右側のパネルにその詳細が表示されます。
2. 使用可能なアクションを確認します。
   - 検出結果の解決 — 対処した後、検出結果を解決済みとしてマークします。
   - コードの修正 – 修正を含むプルリクエストを生成する (GitHub ソースで利用可能)
3. この検出結果は正確ですか? を使用してフィードバックを提供しますか? – はい または いいえ を選択して、将来の分析精度を向上させます。
4. 概要セクションのキー属性を確認します。
   - エージェントの信頼度 – この検出結果における AWS セキュリティエージェントの信頼レベル
   - 重要度 – 色分けされたバッジを持つリスクレベル
   - リスクタイプ – セキュリティリスクのカテゴリ

- 検証ステータス – シミュレートされた検証が有効になっている場合、検出結果が実行中の環境で悪用可能であることが確認されているかどうかを示します。
  - 確認済み – 脆弱性がシミュレートされた環境で正常に悪用されました
  - 再現されない – 脆弱性をターゲットランタイムで悪用できませんでした
  - 検証に失敗しました – エラーのため、検証の試行は決定されませんでした
  - 未検証 – この検出結果に対してシミュレートされた検証が実行されませんでした。これは、検証が有効になっていない場合、または検証ステップがこの検出結果に到達する前にタイムアウトした場合に発生します。
- 解決済み – 結果が解決されたかどうか (はい/いいえ)
- 最終更新日 — 最新の更新のタイムスタンプ

5. 説明セクションを展開して以下を読みます。
   - セキュリティ問題の詳細な説明
   - 脆弱性を悪用する方法
   - アプリケーションへの潜在的な影響
   - エージェントが検出結果を確認するために実行した検証ステップ
6. コードの場所セクションを展開して以下を表示します。
   - 問題が特定された特定のファイルパス
   - 各場所で見つかった内容の簡単な説明
   - 関連するコードにリンクされている行番号バッジ
7. 証拠セクションを展開して以下を確認します。
   - 検出結果をサポートする特定のコード、設定、または観測値
   - 各項目が脆弱性を示している理由の簡単な説明
   - 各証拠の影響を受けるファイルと行番号
8. 推奨される修正セクションを展開して、以下を表示します。
   - AWS セキュリティエージェントが検出結果を修正するために推奨する変更
   - 推奨される各変更のコードまたは設定の例

> **Tip**
>
> コードの場所を使用して、リポジトリ内の影響を受けるファイルにすばやく移動します。各場所には、ファイル全体を読み取らずに問題を理解するのに十分なコンテキストが含まれています。

ステップ 8: 結果の優先順位付けと対処

検出結果を検証してアクションを実行し、脆弱性を修正し、アプリケーションのセキュリティ体制を改善します。

エージェントの信頼度が高い重要度の高い検出結果の場合:

1. 説明とコードの場所のセクションを徹底的に確認します。
2. 修正コードを使用して修正を含むプルリクエストを生成するか、そのオプションを有効にした場合は自動修復 PRsを確認します。
3. フォローアップコードレビューの実行を計画して、修正が有効であることを確認します。

エージェントの信頼性が高い中重要度の検出結果の場合:

1. リスク許容度とビジネスコンテキストに基づいて優先順位を付けます。
2. 定期的な開発スプリント計画に修復タスクを含めます。
3. 複数の中重要度の検出結果が一緒になって高リスクになるかどうかを検討してください。

中または低信頼度の検出結果の場合:

1. 説明とコードの場所セクションを確認して、脆弱性が発生する前提と条件を検証します。
2. 脆弱性が有効な場合は、重要度とリスク許容度に基づいて優先順位を付け、修復タスクを検討します。

ステップ 9: 修復の進行状況を追跡する

検出結果インターフェイスと再実行を使用して、対処された脆弱性を追跡します。

1. 修正を実装するときは、Resolve 検出結果を使用して、検出結果を対応済みとしてマークします。

2. 新しいコードレビューを実行して、修正によって結果が解決され、新しい問題が発生しないことを確認します。
3. コードレビューの詳細ページの「すべての実行」テーブルを確認して、実行間の結果を経時的に比較します。

> **Tip**
>
> 修復プルリクエストをマージしたら、同じコードレビューの新しい実行を開始し、脆弱性が解決されたことを確認します。実行間の検出結果数を比較して、進行状況を追跡します。

次の手順

コードレビューの結果を確認した後:

- 即時修復のために高い信頼性で重要度の高い結果に優先順位を付ける
- Remediate コードを使用して GitHub ソースの検出結果の自動修正を生成する (「」を参照the section called “コードレビューの検出結果の修正”)
- 修正を実装した後に追加のコードレビューを実行して修正を検証する
- コードベースの進化に合わせてコードレビューソースまたは設定を調整する (「」を参照the section called “コードレビューを有効にする”)

コードレビューの検出結果の修正

コードレビューからセキュリティ検出結果を確認したら、AWS セキュリティエージェントを使用して GitHub リポジトリソースで検出結果のコード修正を生成できます。プライベートリポジトリの場合、AWS セキュリティエージェントは提案された修正を含むプルリクエストを開きます。パブリックリポジトリの場合、AWS セキュリティエージェントはローカルに適用できる検出結果にダウンロード可能な差分をアタッチします。S3 ソースの結果は手動で修正する必要があります。

前提条件

開始する前に、以下があることを確認してください。

- 結果を含む完了したコードレビューの実行
- コードレビューのソースとして接続された GitHub リポジトリ

- AWS Security Agent ウェブアプリケーションへのアクセス
- アプリケーションのアーキテクチャとセキュリティ要件に精通していること

コード修復の仕組み

検出結果のコード修復をトリガーすると、AWS セキュリティエージェントは検出結果とそのコードの場所を分析し、コード修正を生成します。修正の配信方法は、ソースによって異なります。

- プライベート GitHub リポジトリ – AWS セキュリティエージェントは、関連するリポジトリにプルリクエストを送信します。プルリクエストには、セキュリティの問題と行われた変更の説明が含まれます。
- パブリック GitHub リポジトリ – AWS セキュリティエージェントは、脆弱性が修正される前に開示されないように、提案された差分を結果にアタッチします。差分はウェブアプリケーションからダウンロードし、 を使用してローカルに適用できますgit apply /path/to/code_remediation_changes.diff。
- S3 ソース – コード修復は使用できません。結果の説明、コードの場所、リスクの推論を使用して、修正を手動で適用します。

> **Important**
>
> AWS Security Agent によって作成されたプルリクエストは、リポジトリへの読み取りアクセス権を持つすべてのユーザーに表示されます。マージする前に変更を確認し、アプリケーションの要件と一致していることを確認します。

自動コード修復

コードレビューの作成時に自動コード修復を有効にした場合、AWS セキュリティエージェントは、レビューが完了するとすぐに、対象となるすべての検出結果の修正を生成します。追加のアクションを実行する必要はありません。プルリクエストは接続されたプライベート GitHub リポジトリに表示され、差分は実行が確定するにつれてパブリック GitHub リポジトリの検出結果に表示されます。

手動コード修復

自動コード修復が無効になっている場合は、GitHub ソースから個々の検出結果の修復をトリガーできます。

1. 完了したコードレビュー実行の結果タブに移動します。
2. 修正する結果を選択します。
3. 検出結果の詳細パネルで、コードの修正を選択します。
4. AWS セキュリティエージェントはコード修正を生成し、リポジトリにプルリクエストを送信するか、リポジトリの可視性に応じてダウンロード可能な差分を結果にアタッチします。

修復プルリクエストを確認する

AWS セキュリティエージェントが修復プルリクエストをプライベート GitHub リポジトリに送信した後:

1. GitHub リポジトリに移動します。
2. AWS セキュリティエージェントによって作成されたプルリクエストを見つけます。
3. 以下を含む変更を確認します。
   - セキュリティの検出結果と修正について説明する説明
   - 脆弱性に対処するコードの変更
   - 修復アプローチに関連するコンテキスト
4. 修正が適切であればプルリクエストをマージするか、閉じて代替ソリューションを実装します。

> **Tip**
> 修復プルリクエストをマージしたら、新しいコードレビュー実行を開始し、修正によって検出結果が解決され、新しいセキュリティ問題が発生しないことを確認します。

修復差分を適用する

パブリック GitHub リポジトリの検出結果については、ダウンロード可能な差分をローカルに適用します。

1. 検出結果の詳細パネルで、コード修復セクションから差分をダウンロードします。
2. リポジトリのローカルクローンで、 を実行しますgit apply /path/to/code_remediation_changes.diff。
3. 適用された変更を確認し、アプリケーションに対してテストして、通常の開発ワークフローを通じてコミットします。

制限事項

- コード修復は、GitHub リポジトリソースの検出結果でのみ使用できます。S3 ソースの結果は手動で修正する必要があります。
- AWS セキュリティエージェントは、脆弱性の分析に基づいて修正を生成します。マージまたはコミットする前にすべての変更を確認して、アプリケーションに適していることを確認します。
- 一部の複雑な検出結果では、自動修正以外の手動介入が必要になる場合があります。

次の手順

検出結果を修正した後:

- GitHub リポジトリでプルリクエストを確認してマージするか、通常のワークフローを通じて適用された差分をコミットします。
- 新しいコードレビューを実行して修正を検証し、残りの問題をチェックする
- 修復を確認した後、ウェブアプリケーションで検出結果を解決する
- 必要に応じてコードレビューのソースまたは設定を調整する (「」を参照the section called “コードレビューを有効にする”)

S3 で差分コードスキャンを実行する

差分 (差分) コードスキャンを実行して、完全なリポジトリスキャンを実行するのではなく、ソースコード内の変更された行のみを分析します。差分スキャンはフルスキャンよりも高速で、特定のコード変更を対象とした結果を生成するため、開発ワークフローでの事前マージ検証に最適です。

サードパーティーのソースコントロールプロバイダーイベントによって自動的にトリガーされるプルリクエストベースのコードレビュー (「」を参照the section called “プルリクエストでコードセキュリティの検出結果を確認する”) とは異なり、差分スキャンは AWS セキュリティエージェント API または SDK を介してプログラムで開始されます。統合差分ファイルを S3 にアップロードし、コードレビュージョブを開始するときに参照します。

差分スキャンの仕組み

差分スキャンは、リポジトリの全コンテキストでコードの変更を分析します。ソースコードが S3 にアップロードされたコードレビューリソースを作成すると、差分スキャンは、差分内の変更された行に特に結果に焦点を当てながら、完全なリポジトリをコンテキストとして使用します。これにより、

スキャンは、認証フローの破損、モジュール間の安全でないデータ処理、既存の脆弱性を公開する変更など、変更が既存のコードとどのように相互作用するかによって発生するセキュリティ問題を特定できます。

スキャンプロセス: 。エージェントスペースに接続されている S3 バケットに、統合差分ファイル (の出力`git diff`) をアップロードします。差分 の S3 の場所を指す`diffSource`パラメータを使用して `StartCodeReviewJob` API を呼び出します。AWS セキュリティエージェントは、セキュリティの脆弱性と組織のセキュリティ要件への準拠について、差分で変更されたコードのみを分析します。検出結果は、重要度評価、コードの場所、修復ガイダンスなど、変更された行に限定されます。

前提条件

開始する前に、以下があることを確認してください。

- コードレビューが有効になっているエージェントスペース (「」を参照the section called “コードレビューを有効にする”)
- ターゲットリポジトリ用に作成済みのコードレビューリソース。完全なソースコードが エージェントスペースに接続された S3 バケットにアップロードされています。完全なリポジトリは、変更が既存のコードとどのように相互作用するかをより詳細に分析できるコンテキストを提供します。コードレビューを作成し、ソースコードをアップロードする手順the section called “コードレビューを作成する”については、「」を参照してください。
- エージェントスペースに接続された S3 バケット
- S3 バケットにアップロードして を呼び出すための IAM アクセス許可 `securityagent:StartCodeReviewJob`
- コード変更から生成された統合差分ファイル (の出力など`git diff main..feature-branch`)

> **Tip**
>
> 最も正確な結果を得るには、コードレビューリソースに、完全なソースコードの最新バージョンが S3 にアップロードされていることを確認します。差分スキャンでは、この完全なリポジトリをコンテキストとして使用して、変更された行を分析するときにアプリケーションアーキテクチャ、データフロー、既存のセキュリティコントロールを理解します。

ステップ 1: 差分ファイルを生成する

スキャンするコード変更を表す統合差分を生成します。

```
git diff main..feature-branch > changes.diff
```

または、ステージングされた変更の差分を生成します。

```
git diff --cached > changes.diff
```

> **Tip**
>
> 差分ファイルは、標準の統一差分形式である必要があります。AWS セキュリティエージェントは、差分ヘッダーのファイルパスと変更された行を解析して、分析の範囲を絞り込みます。

ステップ 2: 差分を S3 にアップロードする

エージェントスペースに接続されている S3 バケットに差分ファイルをアップロードします。

```
aws s3 cp changes.diff s3://my-security-agent-bucket/diffs/changes.diff
```

> **Important**
>
> S3 バケットは、すでにエージェントスペースに接続されているバケットである必要があります。エージェントスペースに関連付けられた IAM サービスロールには、この場所への読み取りアクセス権が必要です。S3 バケットを接続する手順the section called “コードレビューを有効にする”については、「」を参照してください。

ステップ 3: 差分コードレビュージョブを開始する

`diffSource` パラメータを使用して `StartCodeReviewJob` API を呼び出し、差分スキャンを開始します。

AWS CLI の使用

```
aws securityagent start-code-review-job \
  --agent-space-id "your-agent-space-id" \
  --code-review-id "your-code-review-id" \
  --diff-source '{"s3Uri": "s3://my-security-agent-bucket/diffs/changes.diff"}'
```

AWS SDK の使用 (Python)

```
import boto3

client = boto3.client('securityagent')

response = client.start_code_review_job(
    agentSpaceId='your-agent-space-id',
    codeReviewId='your-code-review-id',
    diffSource={
        's3Uri': 's3://my-security-agent-bucket/diffs/changes.diff'
    }
)

print(f"Job started: {response['codeReviewJobId']}")
```

AWS SDK の使用 (JavaScript/TypeScript)

```
import { SecurityAgentClient, StartCodeReviewJobCommand } from '@aws-sdk/client-
securityagent';

const client = new SecurityAgentClient({ region: 'us-east-1' });

const response = await client.send(new StartCodeReviewJobCommand({
  agentSpaceId: 'your-agent-space-id',
  codeReviewId: 'your-code-review-id',
  diffSource: {
    s3Uri: 's3://my-security-agent-bucket/diffs/changes.diff',
  },
}));

console.log(`Job started: ${response.codeReviewJobId}`);
```

API はすぐに `codeReviewJobId`と `status` を返します`IN_PROGRESS`。

ステップ 4: スキャンをモニタリングする

完了するまでコードレビュージョブのステータスをポーリングします。

```
aws securityagent get-code-review-job \
  --agent-space-id "your-agent-space-id" \
  --code-review-id "your-code-review-id" \
```

```
  --code-review-job-id "the-job-id"
```

ジョブは、フルコードレビューと同じフェーズで進行します。プリフライト → 静的分析 → 確定です。差分スキャンは、通常、コード行の分析が少ないため、フルスキャンよりも速く完了します。

ステップ 5: 検出結果を確認する

ジョブが完了したら、コードレビュージョブの結果を一覧表示します。

```
aws securityagent list-findings \
  --agent-space-id "your-agent-space-id" \
  --code-review-id "your-code-review-id" \
  --code-review-job-id "the-job-id"
```

各検出結果には以下が含まれます。

- セキュリティ問題の説明
- 重要度レベル (重大、高、中、低)
- 差分内の特定の行を参照するコードの場所
- 潜在的な影響を説明するリスク推論
- 修正ガイダンス

結果の理解と対応の詳細については、「」を参照してくださいthe section called “コードレビューの結果を確認する”。

クォータと制限

- 差分内の変更されたファイルの最大数: 3,000 ファイルまたは 5 MB の合計差分サイズ
- 検出結果はスキャンごとに 30 に制限され、重要度 (重大 > 高 > 中 > 低) によって優先順位が付けられます。

次の手順

差分スキャンを実行した後:

- 検出結果を確認し、修復ガイダンスを適用する (「」を参照the section called “コードレビューの結果を確認する”)

- すべてのプルリクエストで自動コードレビューのプルリクエストコメントを有効にする (「」を参照the section called “GitHub リポジトリのプルリクエストコードレビューを有効にする”)
- 完全なコードレビューを定期的に実行して、個々の変更以外の問題を検出する (「」を参照the section called “コードレビューを作成する”)
- IDE 統合を使用して、開発環境から直接差分スキャンを実行する (「」を参照the section called “IDE からコードセキュリティスキャンを実行する”)

IDE からコードセキュリティスキャンを実行する

Kiro または Claude コードを使用して、IDE から直接 AWS セキュリティエージェントのコードセキュリティスキャンを実行します。IDE 統合により、開発環境を離れることなく、ローカルソースコードをスキャンしてセキュリティの脆弱性を確認し、変更されたコードでのみ差分スキャンを実行し、設計ドキュメントで脅威モデルレビューを実行できます。検出結果は修復ガイダンスとともにコードとともに表示され、IDE から直接自動修正を適用できます。

IDE 統合の仕組み

IDE 統合では、AWS セキュリティエージェント MCP (モデルコンテキストプロトコル) サーバーを使用して IDE を AWS セキュリティエージェントサービスに接続します。

1. MCP サーバーは、ソースコードを ZIP アーカイブにパッケージ化します。
2. アーカイブは、サーバーが AWS アカウントで自動プロビジョニングする S3 バケットにアップロードされます。
3. サーバーは AWS セキュリティエージェント API を呼び出してスキャンを開始します。
4. AWS セキュリティエージェントは、セキュリティの脆弱性と組織のセキュリティ要件への準拠についてコードを分析します。
5. 検出結果は、コードの場所、説明、重要度評価、修復提案とともに IDE に返されます。

MCP サーバーは、エージェントスペースのプロビジョニング、S3 バケットの作成、IAM ロールのセットアップ、コードパッケージング、スキャンポーリングなど、すべてのオーケストレーションを処理するため、ローカルで設定される AWS 認証情報のみが必要です。

> **Note**
>
> 唯一の前提条件は、ローカル環境 (、AWS SSO、環境変数など) `aws configure`で設定された AWS 認証情報です。MCP サーバーは、初回使用時にエージェントスペース、IAM サービスロール、および S3 バケットを自動プロビジョニングします。

前提条件

開始する前に、以下があることを確認してください。

- AWS 認証情報は、次のアクセス許可でローカルに設定されます。
  - `iam:CreateRole`、 `iam:PutRolePolicy` (サービスロールの 1 回限りのセットアップ用)
  - `s3:CreateBucket`, `s3:PutObject`, `s3:PutPublicAccessBlock`, `s3:PutLifecycleConfiguration`
  - `securityagent:CreateAgentSpace`, `securityagent:UpdateAgentSpace`, `securityagent:ListAgentSpaces`, `securityagent:BatchGetAgentSpaces`
  - `securityagent:CreateCodeReview`, `securityagent:StartCodeReviewJob`, `securityagent:BatchGetCodeReviewJobs`
  - `securityagent:ListFindings`, `securityagent:BatchGetFindings`
  - `securityagent:StartCodeRemediation` (自動修正の場合)
  - `sts:GetCallerIdentity`
- uv がインストールされました (Python パッケージランナー)
- Python 3.10 以降
- 次のいずれかの IDEs
  - Kiro (AWS Security Agent Power をインストール)
  - Claude Code (AWS セキュリティエージェントプラグインをインストールする)

MCP サーバーをインストールする

Kiro

Kiro マーケットプレイスから AWS Security Agent Power をインストールします。Power は、自動セキュリティスキャンの提案のために MCP サーバー設定、ステアリング手順、ライフサイクルフックをバンドルします。

または、プロジェクトの で MCP サーバーを手動で設定します.kiro/mcp.json。

```
{
  "mcpServers": {
    "security-agent": {
      "command": "uvx",
      "args": ["awslabs.security-agent-mcp-server@latest"],
      "env": {
        "AWS_PROFILE": "default",
        "AWS_REGION": "us-east-1",
        "FASTMCP_LOG_LEVEL": "ERROR"
      }
    }
  }
}
```

Claude Code

AWS Security Agent プラグインをインストールします。これにより、セットアップ、フルスキャン、差分スキャン、脅威モデルレビュー、修復のスキルが得られます。

プロジェクトの で MCP サーバーを設定します.mcp.json。

```
{
  "mcpServers": {
    "awslabs.security-agent-mcp-server": {
      "command": "uvx",
      "args": ["awslabs.security-agent-mcp-server@latest"],
      "env": {
        "AWS_PROFILE": "default",
        "AWS_REGION": "us-east-1",
        "FASTMCP_LOG_LEVEL": "ERROR"
      }
    }
  }
}
```

> **Tip**
>
> Python をローカルにインストールしない場合は、Docker 経由で MCP サーバーを実行することもできます。Docker 設定については、MCP サーバードキュメントを参照してください。

初回の設定

初回使用時に、MCP サーバーは必要な AWS リソースを自動的にプロビジョニングします。

[リソース]	名前の規則	目的
エージェントスペース	ユーザー選択または自動作成	スキャン、レビュー、およびペンテスト用のコンテナ
IAM サービスロール	`SecurityAgentScanRole`	アップロードされたコードを読み取る`securityagent.amazonaws.com` ために が引き受ける
S3 バケット	`security-agent-scans-{account}-{region}`	zip 形式のソースコードを保存 (30 日間の自動有効期限ライフサイクル)

セットアップを明示的にトリガーするには、IDE にお問い合わせください。

```
Set up the security agent
```

セットアップでは、AWS 認証情報の検証、エージェントスペースの作成または再利用、IAM ロールと S3 バケットのプロビジョニング、準備状況の確認を行います。AWS マネジメントコンソールのエージェントスペースがすでにある場合、セットアップでそれらが表示され、使用するエージェントスペースが尋ねられます。

> **Note**
>
> まだ設定されていない場合、セットアップは最初のスキャンで自動的に実行されます。個別に実行する必要はありません。

フルセキュリティスキャンを実行する

プロジェクト全体でセキュリティの脆弱性をスキャンします。

```
Scan this project for security issues
```

IDE:

1. ソースコードをアーカイブします (`.git`、`node_modules`、ビルドアーティファクト、その他の必須ではないディレクトリを除く)
2. アーカイブを S3 にアップロードします
3. 完全なコードレビュージョブを開始します
4. 完了のポーリング (通常は約 1 時間)
5. 重要度別にグループ化された結果を表示します

スキャンの進行状況

フルスキャンは通常、コードベースのサイズに応じて約 1 時間かかります。IDE は 5 分ごとに進行状況を確認し、結果の準備ができたら通知します。ステータスはいつでもリクエストできます。

```
How's the scan going?
```

差分スキャンを実行する

開発中のフィードバックを高速化するには、git ref 以降に変更されたコードのみをスキャンします。

```
Scan my changes against main for security issues
```

デフォルトでは、ベース ref を指定しない場合、スキャンはコミットされていない変更を と比較しますHEAD。別のベースを明示的に指定できます。

```
Diff scan my uncommitted changes
```

差分スキャンは、完全なリポジトリコンテキストと git diff パッチの両方をアップロードし、変更された行のみに焦点を当てた分析を実行します。結果は通常 5～15 分で届きます。

S3 差分スキャン API の詳細については、「」を参照してくださいthe section called “S3 で差分コードスキャンを実行する”。

脅威モデルレビューを実行する

STRIDE 手法を使用して、セキュリティ体制の変更について設計ドキュメントを分析します。

```
Run a threat model review on my spec
```

IDE は `requirements.md`および `design.md` ファイル (通常は の下`.kiro/specs/`) を識別し、ソースコードと一緒にアップロードして、脅威モデル分析を実行します。結果は以下を識別します。

- STRIDE で分類されたセキュリティ脅威 (なりすまし、改ざん、否認、情報開示、サービス拒否、特権の昇格)
- 各脅威の重要度評価
- コードベースの特定のアセットへの影響
- 緩和に関する推奨事項

> **Tip**
>
> 仕様から実装タスクを生成する前に、脅威モデルレビューを実行します。これにより、コードが書き込まれる前にセキュリティ設計の問題がキャッチされます。

検出結果を確認する

スキャンが完了すると、結果は重要度別にグループ化された IDE に表示されます。

```
# CRITICAL: SQL Injection in user-service.ts
   File: src/api/user-service.ts:45
   User input flows directly into SQL query without parameterization

# HIGH: Hardcoded credentials in config.ts
```

```
   File: src/config/database.ts:12
   Database password stored in source code
```

また、詳細なレポートは、リスクタイプ、信頼スコア、コードの場所、修復コードなど、各検出結果の完全な情報.security-agent/findings-{scan_id}.mdとともに に書き込まれます。

前回のスキャンの結果を表示するには:

```
Show my security findings
```

自動修正を適用する

検出結果を確認したら、自動修復を適用します。

```
Fix the security findings
```

IDE は、重要度 (重大 → 高 → 中 → 低) 別に検出結果のトップダウンを実行し、各検出結果の修正ガイダンスを使用してコード修正を適用します。すべての修正が適用されると、検証差分スキャンが自動的に実行され、問題が解決されたことを確認します。

個々の検出結果を修正することもできます。

```
Fix the SQL injection in user-service.ts
```

> **Important**
>
> コミットする前に、必ず自動修正を確認してください。修正によって既存の機能が損なわれたり、リグレッションが発生したりしないことを確認します。

自動スキャン提案 (Kiro)

Kiro Power を使用する場合、AWS セキュリティエージェントは、セキュリティに敏感なコード変更を行った後に、差分スキャンを自動的に提案できます。Power は、完了した各コーディングターンを評価するフックをインストールし、変更が以下に影響する場合にスキャンを提案します。

- 認証または認可ロジック
- 暗号化またはシークレット処理

- 入力の検証またはデータのサニタイズ
- ネットワークリクエストまたは外部統合
- セキュリティ設定 (CORS、ヘッダー、セッション処理)

を削除することで、いつでも自動提案をオプトアウトできます`.kiro/hooks/security-diff-scan-suggester.kiro.hook`。

使用可能なスキャンタイプ

[スキャンタイプ]	時間	ユースケース	コマンド
フルスキャン	約 1 時間	コードベース全体の包括的なレビュー	「プロジェクトをスキャンしてセキュリティ上の問題がないか調べる」
Diff スキャン	5 ~ 15 分	コードのみ変更 (コミット前、PR 前)	「変更をスキャンする」または「メインとの差スキャン」
脅威モデル	5 ~ 30 分	設計ドキュメント (requirements.md、design.md)	「自分の仕様で脅威モデルレビューを実行する」

環境変数

可変	説明	デフォルト
`AWS_REGION`	SecurityAgent API コールの AWS リージョン	`us-east-1`
`AWS_PROFILE`	AWS 認証情報プロファイル名	デフォルトプロファイル
`FASTMCP_LOG_LEVEL`	MCP サーバーログレベル (DEBUG、INFO、WARNING、ERROR)	`WARNING`

トラブルシューティング

「設定されていません。最初にセットアップを実行します。」

`.security-agent/config.json` が見つからないか、エージェントスペースが存在しなくなりました。IDE にセットアップの実行を依頼します。

```
Set up the security agent
```

AccessDenied でスキャンが失敗する

AWS 認証情報に、「前提条件」セクションに記載されている必要な IAM アクセス許可があることを確認します。最も一般的な欠落しているアクセス許可は です `iam:CreateRole` (初回セットアップでのみ必要) 。

スキャンがタイムアウトするか、時間がかかりすぎる

- 大規模なコードベースのフルスキャンには 1 時間以上かかる場合があります
- 反復開発に差分スキャンを使用する — 数分で完了します
- ソースアーカイブに不要なディレクトリが含まれていないことを確認します (MCP サーバーは一般的なパターンを自動的に除外します) 。

空の差分 — スキャンの変更なし

コミットされていない変更なしで差分スキャンを実行すると、MCP サーバーは「HEAD に対する変更なし」を報告し、スキャンを開始しません。最初に変更をコミットまたはステージングするか、別のベース ref を指定します。

クォータと制限

- ソースアーカイブの最大サイズ: 2 GB
- S3 バケットライフサイクル: 30 日後にアップロードされたコードが自動削除

次の手順

最初の IDE セキュリティスキャンを実行した後:

- 自動 GitHub 統合のプルリクエストコードレビューコメントを有効にする (「」を参照the section called “GitHub リポジトリのプルリクエストコードレビューを有効にする”)
- 組織固有のポリシー検証のセキュリティ要件を設定する (「」を参照the section called “セキュリティ要件を管理する”)
- 定期的なフルスキャンを実行して、コードベース全体の問題を検出する (「」を参照the section called “コードレビューを作成する”)
- 実装前に新しい機能仕様で脅威モデルレビューを使用する

侵入テストを作成する

テストスコープ、ターゲットドメイン、AWS リソースアクセスを設定して、ウェブアプリケーションの自動侵入テストを設定します。侵入テストは、検証済みドメインに対する実際の攻撃シナリオをシミュレートすることで、実行中のアプリケーションのセキュリティ脆弱性を特定するのに役立ちます。

AWS セキュリティエージェントは、設定されたスコープとアクセス許可に基づいてウェブアプリケーションに対して包括的なセキュリティテストを実行し、攻撃者が脆弱性を発見する前に、悪用可能な脆弱性に関する詳細な検出結果を提供します。

この手順では、テストの詳細を設定し、テストスコープを定義し、必要なアクセス許可を設定して、ペネトレーションテストを作成します。

前提条件

開始する前に、以下があることを確認してください。

- AWS Security Agent ウェブアプリケーションへのアクセス
- テスト用に少なくとも 1 つの検証済みドメイン
- AWS セキュリティエージェントに適切なアクセス許可を持つ IAM ロール
- アプリケーションのアーキテクチャと重要なパスの理解

侵入テストの作成を開始する

エージェントウェブアプリの侵入テストの作成ページに移動します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。

2. 侵入テストセクションに移動します。
3. 「侵入テストを作成する」を選択します。

> **Tip**
>
> ペネトレーションテストに含めることができるのは、検証済みドメインのみです。管理者に AWS マネジメントコンソールでドメインを検証するように依頼します。「the section called "ペネトレーションテスト用のアプリケーションドメインを有効にする"」を参照してください。

侵入テストに名前を付ける

このペネトレーションテストの目的と範囲を特定するのに役立つわかりやすい名前を入力します。

1. 侵入テスト名フィールドに、侵入テストのわかりやすい名前を入力します。

   Example

   名前は、テスト対象のアプリケーション、環境、またはコンポーネントを明確に識別する必要があります。最大 100 文字。

ペネトレーションテストスコープを設定する

テストするドメインと URL パスを定義し、テスト境界を制御するためにオプションの除外を設定します。

ターゲットドメインを追加する

セキュリティの脆弱性についてアクティブにテストされる検証済みドメインを指定します。

1. 「侵入テストスコープ」セクションで、ターゲット URLs。
2. Verified domains セクションを展開して、使用可能なドメインを表示します。
3. ターゲット URL フィールドに、ターゲットドメイン URL を入力します。

> **Important**
>
> 検証済みドメインのみをテストできます。URL は、AWS セキュリティエージェントで以前に検証したドメインにある必要があります。検証済みドメインのサブドメインには、個別の検証は必要ありません。

4. 複数のターゲットドメインを追加するには:
   a. [ドメインを追加する] を選択します。
   b. 追加の各ドメイン URL を入力します。
5. ターゲットドメインを削除するには、ドメイン URL の横にある削除を選択します。

> **Tip**
>
> 最良の結果を得るには、APIs のサブドメイン、認証サービス、コンテンツ配信など、アプリケーションのユーザーフローの一部であるすべてのドメインを含めます。検証済みの親ドメインのサブドメインには、個別の検証は必要ありません。

リスクタイプを除外する (オプション)

アプリケーションに当てはまらない場合は、テストから除外する特定のリスクカテゴリを選択します。

1. Exclude risk types フィールドを見つけます。
2. ドロップダウンを選択して、利用可能なリスクタイプを表示します。
3. ペネトレーションテストから除外するリスクタイプを 1 つ以上選択します。

> **Note**
>
> リスクタイプを除外すると、テストの範囲が制限されます。アプリケーションに関連しないリスクタイプ、または個別にテストするリスクタイプのみを除外します。

out-of-scope URL パスを追加する (オプション)

ペネトレーションテスト中にテストすべきでない URL パスを指定します。AWS セキュリティエージェントは、指定されたパスとその下にネストされたすべてのパスを除外します。たとえば、 をout-of-scope`https://example.com/admin`として追加すると、 `https://example.com/admin/tools` もout-of-scope。

1. Out-of-scopeURLs。
2. Out-of-scope URLs入力フィールドに、除外する URL パス (`/admin/delete`や など`/api/reset`) を入力します。

> **Warning**
>
> Out-of-scopeのパスは脆弱性についてテストされません。破壊的な操作や機密性の高い管理機能など、テスト中にアクセスすべきではないパスのみを除外してください。

3. out-of-scopeのパスを複数追加するには:
   a. URL の追加 を選択します。
   b. 追加の各パスを入力します。
4. パスを削除するには、パスの横にある削除を選択します。

アクセス可能なドメインを追加する (オプション)

テストには必要ですが、脆弱性テストのターゲットではないドメインを指定します。

1. アクセス可能な URLsセクションを見つけます。
2. アクセス可能な URLs の入力フィールドに、テスト中にアクセス可能なドメインを入力します。

> **Note**
>
> ターゲットドメイン外のサードパーティーサービス (Okta、Auth0、Stripe など) にアクセス可能なドメインを追加します。これは、AWS セキュリティエージェントがテスト中にログインとナビゲーションのためにこれらの URLs にアクセスできるようにするために必要です。AWS セキュリティエージェントはこれらのドメインをペネトレーションテストしません。これらのドメインはアクセス目的でのみ使用されます。アクセス可能なドメインは、ターゲットとは異なるドメインに属していても、所有権の検証を必要としません。ターゲットドメインのみが、検証済みの所有権を必要とします。

3. 複数のアクセス可能なドメインを追加するには:
   a. URL の追加 を選択します。
   b. 追加の各ドメインを入力します。
4. ドメインを削除するには、ドメインの横にある削除を選択します。

カスタム HTTP ヘッダーを追加する (オプション)

侵入テスト中に AWS セキュリティエージェントによって行われたリクエストに追加されるカスタム HTTP ヘッダーを指定します。

1. カスタム HTTP ヘッダーセクションを見つけます。
2. カスタム HTTP ヘッダー入力フィールドに、アウトバウンドリクエストに関連付けられるヘッダー名と値を入力します。

> **Note**
> デフォルトでは、AWS セキュリティエージェントは、別のカスタム User-Agent ヘッダー値が指定されていない限り、セキュリティエージェントに設定された User-Agent のカスタムヘッダーを追加します。

3. 複数のカスタムヘッダーを追加するには:
   a. [ヘッダーの追加] を選択します。
   b. 追加の各カスタムヘッダーを入力します。
4. カスタムヘッダーを削除するには、ヘッダーの横にある削除を選択します。

IAM ロールを設定する

このペネトレーションテスト用に事前設定されたサービスロールを選択します。AWS セキュリティエージェントは、エージェントスペースをセットアップするときに管理者が IAM ロールを設定するエージェントスペースベースのアクセス許可モデルを使用します。すでに設定され、すぐに使用できるロールから選択します。

1. アクセス許可セクションで、サービスロールドロップダウンを見つけます。
2. 必要な AWS リソースへのアクセス権を AWS セキュリティエージェントに付与する IAM ロールを選択します。

> **Important**
>
> 選択した IAM ロールには、VPC リソース、CloudWatch Logs、およびペネトレーションテストに必要なその他の AWS サービスにアクセスするためのアクセス許可が必要です。ロールに AWS セキュリティエージェントとの正しい信頼関係があることを確認します。

3. CloudWatch ロググループのドロップダウンを見つけます。
4. 侵入テストログが保存されるロググループを選択します (オプション) 。

> **Note**
>
> 選択した CloudWatch ロググループは、行われたリクエスト、受信したレスポンス、検出された脆弱性など、侵入テスト実行の詳細なログを保存します。
> ロググループを選択しない場合、新しい CloudWatch ロググループが `/aws/securityagent` プレフィックスで自動的に作成され、侵入テストログが保存されます。

自動コード修復

自動修復を有効にするチェックボックスをオンにします。

> **Important**
>
> ソースコードリポジトリのセキュリティ検出結果を修復するために、AWS セキュリティエージェントはリポジトリにプルリクエストを送信する場合があります。プルリクエストは、リポジトリへの読み取りアクセス権を持つすべてのユーザーに表示される場合があります。

VPC リソースを設定する (オプション)

ターゲットドメインがプライベートで VPC 内でホストされている場合は、AWS セキュリティエージェントがペネトレーションテストを実行する VPC 設定を構成します。このステップは、パブリックにアクセスできないアプリケーションにのみ必要です。

> **Note**
>
> ターゲットドメインがパブリックにアクセスできる場合は、このステップをスキップします。VPC 設定は、Amazon Virtual Private Cloud 内でホストされているプライベートアプリケーションのテストにのみ必要です。

ペネトレーションテスト環境の VPC、サブネット、セキュリティグループを選択します。

1. VPC セクションで、VPC ID ドロップダウンを見つけます。
2. ターゲットドメインがホストされている VPC を選択します。

> **Important**
>
> 選択した VPC には、ステップ 3 で指定したターゲットドメインが含まれている必要があります。AWS セキュリティエージェントがアプリケーションにアクセスできるように、VPC に適切なルーティングとネットワーク設定があることを確認します。

3. Subnets ドロップダウンを見つけます。
4. ペネトレーションテストを実行するサブネットを 1 つ以上選択します。

> **Note**
>
> ターゲットアプリケーションへのネットワークアクセスを持つサブネットを選択します。ペネトレーションテストは、これらのサブネットにデプロイされたリソースから実行されます。

5. セキュリティグループのドロップダウンを見つけます。
6. ペネトレーションテストのネットワークアクセスを制御するセキュリティグループを選択します。

> **Important**
>
> 選択したセキュリティグループは、ターゲットドメインとアクセス可能なドメインへのアウトバウンドトラフィックを許可する必要があります。セキュリティグループのルールで、包括的なテストに必要なネットワークアクセスが許可されていることを確認します。

認証情報を設定する (オプション)

ターゲットドメインに認証が必要な場合は、AWS セキュリティエージェントがペネトレーションテスト中にアプリケーションの保護領域にアクセスできるようにする認証情報を指定します。このステップは、ユーザー認証を必要とするアプリケーションにのみ必要です。

> **Note**
>
> ターゲットドメインが認証を必要としない場合、またはテストするすべての領域がパブリックにアクセスできる場合は、このステップをスキップします。AWS セキュリティエージェントがアプリケーションの認証済みセクションをテストする必要がある場合にのみ、認証情報を設定します。

認証情報の追加

AWS セキュリティエージェントがアプリケーションへのアクセスに使用する認証情報を指定します。

1. 認証情報 #1 セクションで、認証情報の入力方法を選択します。
   - 認証情報の入力 - 認証情報を AWS セキュリティエージェントに直接入力します。
   - 詳細設定 - 機密情報については、AWS Secrets Manager や AWS Lambda 関数などの詳細オプションを使用します。詳細については、「the section called “ペネトレーションテスト用の認証情報を提供する”」を参照してください。

     > **Tip**
     >
     > 本番環境または機密認証情報の場合は、高度な設定オプションを使用して、AWS Secrets Manager または Systems Manager パラメータストアに保存されている認証情報を安全に参照することをお勧めします。

認証情報の詳細を入力する

認証されたアカウントのユーザー名とパスワードを入力します。

1. ユーザー名フィールドに、認証用のユーザー名を入力します。
2. パスワード フィールドに、認証用のパスワードを入力します。

> **Important**
>
> 指定する認証情報に、テストする領域に適したアクセスレベルがあることを確認します。認証情報は、管理者権限ではなく、一般的なユーザーのアクセスレベルを表す必要があります。

アクセスドメインの選択

これらの認証情報を認証に使用するターゲットドメインを指定します。

1. アクセスドメインドロップダウンで、これらの認証情報を使用するドメインを選択します。

> **Note**
>
> 異なる認証情報を必要とする複数のターゲットドメインがある場合は、この認証情報設定を完了した後に別の認証情報を追加をクリックして、認証情報セットを追加できます。

エージェントログインプロンプトを設定する (オプション)

アプリケーションの認証プロセスを通じて AWS セキュリティエージェントをガイドする手順を提供します。

1. 認証フローに特定の手順が必要な場合は、 エージェントログインプロンプトセクションを展開します。
2. 提供された認証情報をアプリケーションのログインフローで使用する方法を説明する手順を入力します。

> **Note**
>
> エージェントログインプロンプトは、アプリケーションに認証情報を適用する方法をエージェントに指示します。これは、複雑な認証フロー、複数ステップのログインプロセス、または標準以外のログイン手順を使用するアプリケーションに役立ちます。「/login に移動し、「Email」フィールドにユーザー名を入力し、パスワードを入力して「Sign In」を選択する」などのstep-by-stepの手順を含めます。

複数の認証情報を追加する (オプション)

アプリケーションで複数の認証情報セットが必要な場合、または異なるドメインに個別の認証が必要な場合は、追加の認証情報セットを追加します。

1. 最初の認証情報設定が完了したら、別の認証情報を追加を選択します。
2. 追加の認証情報セットごとに認証情報設定手順を繰り返します。
3. 認証情報セットを削除するには、認証情報ヘッダーの横にある削除を選択します。

> **Tip**
>
> さまざまなユーザーロールをテストするとき、複数の認証済みドメインにアクセスするとき、またはアプリケーションでロールベースのアクセスコントロールを検証するときに、複数の認証情報を設定します。

追加のリソースをアタッチする (オプション)

AWS セキュリティエージェントがより徹底的かつ正確な侵入テストを実施するのに役立つ補足リソースを提供します。追加のリソースには、アーキテクチャ図、API ドキュメント、設定ファイル、GitHub リポジトリ、またはアプリケーションに関するコンテキストを提供する S3-hostedマテリアルが含まれます。

> **Note**
>
> 追加のリソースはオプションですが、推奨されます。アプリケーションに関する包括的な情報を提供すると、徹底的なテストカバレッジを確保し、誤検出を減らし、より実用的な結果を得ることができます。

侵入テストにリソースを追加する

既存のリソースを選択するか、ペネトレーションテストのガイドに役立つ新しいファイルをアップロードします。

1. 接続されたリソースセクションでは、次のことができます。
   - Select from available を選択して、AWS セキュリティエージェントに既に接続されているリソース (GitHub リポジトリや S3 バケットなど) から選択します。

- アップロードを選択して、ローカルシステムから新しいファイルを直接追加します。

> **Tip**
>
> 便利なリソースには、API ドキュメント、アーキテクチャ図、OpenAPI/Swagger 仕様、設定ファイル、認証フロー図、およびアプリケーションの構造と動作を説明するその他の資料が含まれます。

使用可能なリソースから選択する

AWS セキュリティエージェントと既に統合されているリソースから選択します。

1. 既存のリソースから選択を選択します。
2. 次のような接続されたソースから使用可能なリソースのリストを参照します。
   - GitHub リポジトリタブの GitHub リポジトリ
   - S3 バケット
   - 以前にアップロードされたファイル
   - ドキュメントリポジトリ
3. ペネトレーションテストに含めるリソースを選択します。
4. ペネトレーションテストに追加 を選択して、選択したリソースをアタッチします。

Example

AWS セキュリティエージェントがアプリケーションコンテキストの理解を深め、プルリクエストを通じてready-to-implementできるコード修正を生成できるように (有効になっている場合) 、関連する GitHub リポジトリを選択してペンテストに追加することをお勧めします。

> **Note**
>
> 使用可能なソースから選択されたリソースは、元の場所と同期されたままになります。GitHub リポジトリまたは S3 ファイルを更新すると、ペネトレーションテストでは更新されたバージョンが使用されます。

> **Note**
>
> ペンテストに関連付けられたプライベート VPC があり、GitHub リポジトリが設定されている場合は、GitHub リソースをプルするためにプライベート VPC 経由で GitHub にアクセスできることを確認してください。ほとんどの場合、VPC NAT Gateway 経由のアウトバウンドトラフィックがデフォルトで許可されていることを確認するか、GitHub IPs のアウトバウンドトラフィックを許可するように特定のルールを設定する必要があります (GitHub Meta API Endpoint を参照) 。

新しいリソースをアップロードする

ローカルシステムから直接ファイルをアップロードするか、AWS セキュリティエージェントにプレーンテキストコンテンツを提供します。

1. アップロード を選択します。
2. 次のいずれかの入力方法を選択します。
   - ローカルファイルをアップロードする - ローカルシステムから 1 つ以上のファイルを選択します。
   - プレーンテキストの貼り付け - テキストコンテンツを入力フィールドに直接入力または貼り付けます。アップロード を選択します。
3. 次に、追加を選択してアップロードを完了します。
4. アップロードされたリソースは、Connected resources テーブルに表示されます。

> **Tip**
>
> 個別のファイルを作成せずに API エンドポイントリスト、URL パターン、テスト手順、またはその他のテキストベースの情報をすばやく提供する場合は、プレーンテキストオプションを使用します。

> **Important**
>
> アップロードされたファイルと貼り付けられたコンテンツに、本番認証情報、プライベートキー、個人を特定できる情報 (PII) などの機密情報が含まれていないことを確認します。設定ファイルとドキュメントのサニタイズされたバージョンを使用します。

既存のリソースを接続する

既存のリソースは、以前に AWS セキュリティエージェントにアップロードしたもの、S3 バケット、および統合された GitHub リポジトリから取得できます。既存のリソースから選択を選択して選択します。

接続されたリソースを管理する

ペネトレーションテストにアタッチされたリソースを確認、整理、削除します。

接続されたリソーステーブルには、ペネトレーションテストに含まれるすべてのリソースが以下の情報とともに表示されます。

- 名前 - ファイル名またはリソース識別子
- タイプ - リソースカテゴリ (アップロードされたファイル、S3 リソース、GitHub リポジトリなど)

リソースを管理するには:

1. チェックボックスを使用して 1 つ以上のリソースを選択します。
2. 選択したリソースをデタッチするには、ペネトレーションテストから削除を選択します。

> **Note**
>
> 列ヘッダーをクリックして、テーブルを名前またはタイプでソートできます。これにより、多くのファイルを操作するときにリソースを整理できます。

侵入テストを作成する

ペネトレーションテスト設定を確定して起動します。

すべての設定を設定したら、ペネトレーションテストを作成する準備が整います。

1. すべての設定セクションを確認して、正確性を確認します。
2. 以下のオプションのいずれかを選択してください。
   - ペネトレーションの作成を選択して、設定をすぐに実行せずに保存します。
   - Create and execute を選択して設定を保存し、すぐにペネトレーションテストを開始します。
   - キャンセルを選択して、ペネトレーションテスト設定を破棄します。

Important

侵入テストを実行する前に、以下を確認してください。

- すべてのターゲットドメインが正しく検証され、アクセス可能である
- IAM ロールに適切なアクセス許可がある
- Out-of-scopeパスは、破壊的なオペレーションのテストを防ぐために適切に設定されています
- すべてのターゲットドメインでセキュリティテストを実行する権限がある

Note

ペネトレーションテストが開始されたら、「ペネトレーションテストの実行」セクションから進捗状況をモニタリングできます。テストが完了するまでに数時間かかる場合があります。ほとんどは、アプリケーションの範囲と複雑さに応じて、16 時間以内に完了します。

ペネトレーションテストの結果を確認する

AWS セキュリティエージェントがペンテストを開始した後、ペネトレーションテストログページでペンテストの実行をリアルタイムでモニタリングします。AWS セキュリティエージェントは、ペンテスト中にすべてのアクションを記録します。完了後、アプリケーションの概要、特定されたエンドポイントのカバレッジ、セキュリティ検出結果のリスク評価を含むペンテストの概要を確認します。

セキュリティの検出結果を評価して、アプリケーションの脆弱性に対処します。各検出結果には、影響評価、重大度評価、サポート証拠、修復プルリクエストの詳細 (自動コード修復が有効になっている場合) が含まれます。

前提条件

開始する前に、以下があることを確認してください。

- 完了した、または進行中のペネトレーションテストの実行
- AWS Security Agent ウェブアプリケーションへのアクセス

ステップ 1: ペネトレーションテストの実行にアクセスする

ペネトレーションテストの実行に移動して、概要、ログ、検出結果ページを表示します。

1. AWS セキュリティエージェントのウェブアプリケーションにログインします。
2. 侵入テストセクションに移動します。
3. 検査するペネトレーションテストの実行をリストから選択します。

> **Tip**
>
> 侵入テストの詳細ページには、テストのステータス、完了日、特定された検出結果の数の概要が表示されます。

ステップ 2: テストの進行状況をモニタリングする

ステップインジケータを使用して、侵入テストの実行の進行状況を追跡します。

1. ページヘッダーの下にある水平ステップインジケータを見つけます。
2. 各テストフェーズのステータスを確認します。
   - プリフライト – 初期セットアップと接続チェック
   - 静的分析 – コードと設定の分析
   - ペンテスト – ランタイムテストと脆弱性スキャン
   - 最終化 — 最終検証とレポート生成

> **Note**
>
> 各ステップには、ステータスインジケータ (完了、進行中、保留中) が表示されます。検出結果はテストプロセス全体で検出および検証され、各フェーズが完了すると新しい脆弱性が表示されます。

ステップ 3: 侵入テスト実行の概要タブに移動する

1. Run Summary セクションには、テストのステータス、期間、その他の高レベルの詳細が表示されます。また、重要度レベルとリスクタイプ別に分類されたセキュリティ検出結果のダッシュボードも提供します。
2. AWS セキュリティエージェントによるアプリケーションの概要は、ペネトレーションテストの実行の概要を提供します。
3. AWS セキュリティエージェントによって検出されたエンドポイントは、ペンテストの実行中に AWS セキュリティエージェントによって検出およびテストされたすべてのエンドポイントのリストを提供します。

ステップ 4: 侵入テストログタブに移動する

ペンテスト中に AWS セキュリティエージェントが実行したすべてのアクションの詳細ログにアクセスします。

1. アクションは、アクションタイプとリスクタイプ別に分類されます。
2. 特定のアクションを選択して、詳細なログを表示します。
   - テストの概要 – エージェントアクションと結果の概要
   - 侵入テストログ – すべてのテストアクティビティの詳細ログ

> **Note**
>
> 検証アクションは、各カテゴリの検出結果を検証するログを提供します

> **Note**
>
> 検出結果タブには、左側に検出結果リスト、右側に選択した検出結果の詳細を含む分割ビューが表示されます。

ステップ 5: 検出結果に移動する

リスト内の各検出結果には、重要度をすばやく評価するための重要な情報が表示されます。

> **Note**
>
> デフォルトの信頼度フィルター
>
> デフォルトでは、エージェントの信頼度が高い結果のみが表示されます。エージェントの信頼度が中または低の検出結果と誤検出を表示するには、未検証の検出結果の非表示トグルをオフにします。

各検出結果カードに表示される情報を確認します。

- 検出結果名 – 脆弱性のタイトルと識別子
- 信頼バッジ – 検出結果におけるエージェントの信頼レベルを示します (高、中、低)
- 重要度バッジ – 色分けされたリスクレベルを表示します。
  - 重大 (赤) – 即時のアクションが必要。悪用するとシステムの侵害につながる可能性があります
  - 高 (赤) – 迅速な注意が必要です。悪用すると、セキュリティに大きな影響を与える可能性があります
  - 中程度 (オレンジ) – 妥当な期間内に対処する必要があります。全体的なセキュリティリスクに寄与します
  - 低 (黄色) – 定期的なメンテナンスの一環として対処でき、即時リスクを最小限に抑える
  - 情報 (青) – 情報目的、即時リスクは最小限またはまったくない
- 最終更新タイムスタンプ – 検出結果が最後に変更または検証された日時を示します。
- 説明プレビュー – 脆弱性の簡単な概要

> **Important**
>
> 重大度が重大または高のバッジと高い信頼度で検出結果に優先順位を付けます。これらのバッジは、即時修復を必要とする検証済みの脆弱性を表すためです。

ステップ 6: 結果の詳細を確認する

個々の検出結果を選択して、各脆弱性に関する包括的な情報を表示します。

1. 左側のパネルで結果名を選択すると、その詳細が右側のパネルに表示されます。
2. 検証ステータスを確認します。

> **Note**
>
> 検出結果に「この検出結果は AWS セキュリティエージェントによってまだ検証されていません」と表示されている場合は、脆弱性検出がまだ確認されていることを意味します。これらの検出結果には、手動検証が必要になる場合があります。

3. 上部に表示されるキー属性を確認します。
   - エージェントの信頼度 – この検出結果における AWS セキュリティエージェントの信頼レベル
   - 重要度 – 色分けされたバッジを持つリスクレベル
   - ログの検索 – 「トレースアクションとログ」を選択して、詳細な実行ログと証拠を表示します。
   - リスクタイプ – セキュリティリスクのカテゴリまたはタイプ (認証バイパス、SQL インジェクションなど)
4. 説明セクションを展開して以下を読みます。
   - 脆弱性の詳細な説明
   - 脆弱性の仕組み
   - セキュリティリスクを表す理由
   - アプリケーションへの潜在的な影響
5. リスクの理由セクションを展開して、重要度の計算を理解します。
   - CVSS (共通脆弱性評価システム) メトリクスの内訳
   - 攻撃ベクトル (AV) — 脆弱性を悪用する方法
   - 攻撃の複雑さ (AC) — エクスプロイトの難易度

- 必要な権限 (PR) – 必要なアクセスレベル
- ユーザーインタラクション (UI) – ユーザーアクションが必要かどうか
- スコープ (S) — 脆弱性が他のコンポーネントに影響を与えるかどうか
- 機密性、完全性、可用性への影響

6. 「再現するステップ」セクションを展開して以下を表示します。
   - 脆弱性を再作成するための詳細な技術的手順
   - リクエストとレスポンスの例
   - 問題をトリガーする特定のパラメータまたは条件
7. 検証スクリプトセクションには、結果を再現するための実行可能な方法が用意されています。このセクションを展開して (利用可能な場合) 、以下を表示します。
   - 手順 – 検証スクリプトをセットアップして実行する方法
   - 環境変数 – スクリプトを実行する前に設定する必要がある変数を一覧表示します。機密値はセキュリティのために秘匿化されます。
   - ダウンロードスクリプト – 実行可能検証スクリプトのダウンロードを選択します。

> **Note**
>
> 検証スクリプトは、確認された脆弱性に対してのみ使用できます。エージェントは、実行可能再生成スクリプトを正常に生成して検証する必要があります。

> **Note**
>
> 検証スクリプトは生成 AI を使用して生成されます。実行前にスクリプトを確認し、テストが許可されているシステムに対してのみ実行します。AI 生成スクリプトの責任あるテストに関するガイダンスについては、AWS の責任ある AI ポリシーを参照してください。

> **Tip**
>
> 検証スクリプトは、結果を個別に再現するための実行可能な方法を提供します。独自の認証情報を使用して必要な環境変数を設定し、ターゲットシステムに対してスクリプトを実行して脆弱性が存在することを確認します。

> **Tip**
>
> 「トレースアクションとログ」リンクを使用して、脆弱性を示す HTTP リクエスト、レスポンス、悪用の試みなど、完全な証拠パッケージにアクセスします。

検出結果を編集する

検出結果を編集して詳細を修正したり、エージェントの評価を絞り込んだりできます。次のフィールドは編集可能です。

- name — 検出結果のタイトル
- description — セキュリティ脆弱性の 詳細な説明
- status — 検出結果の 現在のステータス
- riskType — 特定されたセキュリティリスクのタイプ
- riskLevel — 重要度レベル (重大、高、中、低、情報)
- riskScore — 数値リスクスコア
- 推論 — 割り当てられたリスクスコアの根拠
- attackScript — 脆弱性を示す Proof-of-conceptコード
- customerNote — 編集の根拠を説明する オプションのメモ

検出結果を編集するには:

1. 検出結果の詳細ページに移動します。
2. 編集アイコンを選択して、前のリストのいずれかのフィールドを変更します。
3. 変更内容を保存します。

> **Note**
>
> 編集はすぐに保存され、結果に反映されます。Findings Personalization が有効になっている場合、変更したすべての編集可能なフィールドを使用して、今後の実行に関するエージェントの設定を絞り込みます。

元のエージェントバージョンを表示する

結果を編集すると、結果の詳細ヘッダーにバージョンセレクターが表示されます。これにより、元のエージェント評価を表示できます。

1. 結果の詳細ヘッダーでバージョンセレクターを見つけます。
2. Original を選択すると、エージェントによって最初に生成された結果が表示されます。
3. 最新 を選択して、編集が適用された現在のバージョンに戻ります。

パーソナライズされた検出結果を確認する

検出結果のパーソナライゼーションを有効にすると、AWS セキュリティエージェントは編集から検出結果までを学習します。エージェントは、これらの設定を今後のペネトレーションテストの実行で同様の結果に適用します。前回の編集に基づいて結果を調整すると、詳細パネルにパーソナライゼーションの変更セクションが表示されます。この機能を有効にする方法については、「結果のパーソナライゼーションを有効または無効にする」を参照してください。

1. 結果の詳細パネルでパーソナライゼーションの変更セクションを見つけます。

> **Note**
>
> パーソナライゼーションの変更セクションは、AWS セキュリティエージェントが以前の編集に合致した検出結果にのみ表示されます。学習した設定に一致しない検出結果は、このセクションなしで、エージェントが生成した検出結果とまったく同じように表示されます。

2. 変更内容と理由の概要を確認します。概要には、調整された各属性が、元のエージェント生成値、パーソナライズされた値、推論とともに一覧表示されます。例えば、次のようになります。

前の編集に基づいて、システムが最初に生成した検出結果に次の変更が行われました。リスクレベルが MEDIUM から CRITICAL に変更されました。リスクスコアが 5.3 から 9.5 に変更されました。理由: パブリックにアクセス可能なソースマップを介したアプリケーションソースコードの完全な公開を反映するように重要度をアップグレードしました。

1. 概要を確認して、結果に対する対処方法を決定する前に、結果がどのように調整されたかを理解します。

2. パーソナライズされた検出結果を上書きするには、他の検出結果と同様に再度編集します (重要度やリスクスコアの変更など) 。最新の編集が常に優先されます。AWS セキュリティエージェントはそこから学習し、更新された設定を次の実行に適用して、以前のルールを置き換えます。

> **Note**
>
> パーソナライゼーションの変更により、riskLevel、riskScore、名前、説明、推論、attackScriptなどの検出結果属性が調整され、AWS セキュリティエージェントが編集から学習した標準が反映されます。以前に編集したフィールドは、今後の実行で同様の結果に基づいて調整される場合があります。

パーソナライゼーションが編集から学習する方法

検出結果のパーソナライゼーションを有効にすると、AWS セキュリティエージェントは検出結果に加えたすべての編集から学習し、今後の実行に同様の調整を適用します。編集したフィールド (前の結果の編集セクションを参照) は、エージェントの学習した設定を絞り込むために使用されます。

> **Note**
>
> ステータスフィールドでは、結果を FALSE_POSITIVE としてマークするだけが学習可能な設定として扱われます。他のステータス値を変更しても、学習はトリガーされません。

次のペンテストの実行では、エージェントは学習した設定に対して新しい検出結果を評価し、必要に応じて調整を適用します。パーソナライゼーションの変更セクションは、調整されたすべての結果に表示され、変更された内容が説明されます。

> **Note**
>
> パーソナライゼーションは、最後に完了したペンテスト実行で行われた編集からのみ学習します。この機能は、学習のためにペンテストが開始される時点で有効にする必要があります。後で同じ結果を編集する場合、最新の編集が優先されます 。エージェント は最新の決定を反映するように設定を更新します。

結果のパーソナライゼーションを有効または無効にする

検出結果 パーソナライゼーションはデフォルトで有効になっています。無効化または再有効化するには:

1. ペンテスト設定に移動します。
2. Findings Personalization トグルを見つけます。
3. Findings Personalization を有効にするには、トグルをオンにします。無効にするには、オフにします。

無効にすると、結果は未加工のエージェント生成状態になり、パーソナライゼーションは適用されません。以前に学習した設定は保持され、機能を再度有効にすると再度適用されます。

ステップ 7: CVSS メトリクスを解釈する

CVSS メトリクスを理解することで、真の重要度を評価し、修復作業に優先順位を付けることができます。

理由セクションを確認するときは、以下の主要なメトリクスに注意してください。

- 攻撃ベクトル (Network/Adjacent/Local/物理) — 攻撃をリモートで実行する方法を示します
- 攻撃の複雑さ (低/高) — 特殊な条件を悪用する必要があるかどうかを示します
- 必要な権限 (None/Low/High) – 攻撃者が必要とするアクセスレベルを特定します
- ユーザーインタラクション (なし/必須) – エクスプロイトにユーザーの関与が必要かどうかを判断します
- Confidentiality/Integrity/Availabilityへの影響 (None/Low/High) – システムのセキュリティへの影響を測定します

> **Important**
>
> ネットワーク攻撃ベクトル、複雑さの低い、機密性/整合性の高い影響の検出結果は、即時の対応を必要とする最も危険な脆弱性を表します。

ステップ 8: 結果の優先順位付けと対処

検出結果に対してアクションを実行して脆弱性を修正し、アプリケーションのセキュリティ体制を改善します。

高い信頼度を持つ重大度および高の調査結果の場合:

1. 説明とステップを確認して、セクションを徹底的に再現します。
2. 「トレースアクションとログ」リンクから詳細なログにアクセスして、完全な証拠を収集します。
3. 次のいずれかの方法を使用してready-to-implementコード修正にアクセスします。
   - 自動修復の場合: 修復セクションのプルリクエストリンクを使用します。
   - 手動リクエストの場合: 結果ページで「修復コード」を選択してプルリクエストをリクエストする 前提条件:
   - 管理者は、AWS セキュリティエージェントコンソールで GitHub リポジトリのコード修復を有効にする必要があります
   - リポジトリはペンテスト設定に含める必要があります
4. フォローアップのペネトレーションテストを計画して、修正が有効であることを確認します。

重要度が中および低の場合:

1. リスク許容度とビジネスコンテキストに基づいて優先順位を付けます。
2. 定期的な開発スプリント計画に修復タスクを含めます。
3. 複数の重要度の低い検出結果が一緒になってリスクが高くなるかどうかを検討してください。
4. 受け入れられたリスクがあれば、適切な根拠を添えて文書化します。

> **Important**
>
> 重要度の低い検出結果は無視しないでください。複数の重要度の低い脆弱性を連鎖させることで、特にソーシャルエンジニアリングや物理アクセスと組み合わせると、より深刻なエクスプロイトが発生することがよくあります。

ステップ 9: 修復の進行状況を追跡する

検出結果インターフェイスを使用して、対処された脆弱性と追加のアクションが必要な脆弱性を追跡します。

1. 修復を進めるときは、「再現する手順」セクションに戻って修正を確認します。
2. 将来のリファレンス監査とコンプライアンス監査のために、各検出結果の修復アプローチを文書化します。

> **Tip**
>
> 脆弱性に対処するために行われたコードの変更、設定の更新、アーキテクチャ上の決定など、各検出結果を解決にマッピングする修復ログを維持します。

次の手順

ペネトレーションテストの結果を確認した後:

- 即時修復のために、重要かつ重要度の高い検出結果を高い信頼性で優先順位付けする
- テスト環境で検証スクリプトをダウンロード、レビュー、実行して、スクリプトをテストし、脆弱性を特定する
- 検出結果の詳細と証拠へのリンクを使用して、問題管理システムで追跡チケットを作成する
- 特定された脆弱性に対処するための修正とセキュリティコントロールを実装する
- ペネトレーションテストの実行の進行状況インジケータで新しく検出された脆弱性をモニタリングする
- 脆弱性が適切に修正されていることを確認するためのフォローアップ侵入テストをスケジュールする
- 検出結果に基づいてアプリケーションセキュリティテストプロセスと脅威モデルを更新する
- CVSS メトリクスを確認して、アプリケーションの全体的なセキュリティ体制を理解する

ペネトレーションテストの実行の詳細については、「」を参照してくださいthe section called “侵入テストを作成する”。

セキュリティエージェントのライフサイクルを理解する方法の詳細については、「」を参照してくださいthe section called “リソース階層とライフサイクルを理解する”。

ペネトレーションテスト用の認証情報を提供する

AWS セキュリティエージェントがウェブアプリケーションの認証済み領域をテストできるようにする認証情報を提供します。認証情報がない場合、エージェントはパブリックにアクセス可能なページと APIs。

認証情報を設定する

1. 侵入テストの作成ワークフローで、認証認証情報 - オプションセクションを見つけます。
2. 認証情報 #1 セクションで、認証情報の入力方法を選択します。
   - 認証情報の入力 - 認証情報を直接入力します。開発環境とテスト環境に最適です。
   - 詳細設定 - AWS ネイティブの認証情報管理を使用します。本番環境と機密認証情報に推奨されます。

高度なオプション

詳細設定を選択した場合は、3 つの認証情報戦略から選択できます。

- IAM ロールの引き受け - AWS Cognito または IAM 認証を使用するアプリケーションの場合
- AWS Secrets Manager - 暗号化とローテーションによる安全な認証情報ストレージ用
- Lambda 関数 - 動的な認証情報の生成または複雑な認証フローの場合

認証情報を直接入力する

1. 認証情報の入力を選択します。
2. ユーザー名とパスワードを入力します。
3. アクセス URL ドロップダウンで、これらの認証情報を使用する URL を選択します。これは、ターゲットエンドポイントのリストから選択する必要があります。
4. (オプション) 2FA - オプションフィールドで、2 要素認証を必要とするアプリケーションに TOTP シークレットを指定します。次のいずれかを行うことができます。
   - TOTP シークレットを直接入力するか (例: `JBSWY3DPEHPK3PXP`)、完全な `otpauth://totp/` URI を入力します (例: `otpauth://totp/Example:user@example.com?secret=JBSWY3DPEHPK3PXP&issuer=Example`)。
   - アップロードアイコンを選択して、認証アプリケーションのセットアップページから QR コードイメージをアップロードします。QR コードはローカルでスキャンされ、TOTP URI は自動的に抽出されます。

TOTP シークレットを指定すると、エージェントは新しいワンタイムコードを自動的に生成し、ログイン中に 2FA プロンプトが検出されると入力します。

5. (オプション) アプリケーションに複雑な認証フローがある場合は、エージェントスペースのログインプロンプトを展開して、特定のログイン手順を指定します。

> **Important**
>
> 個人アカウントや管理アカウントではなく、代表的なアクセス権を持つテストアカウントを使用します。

詳細設定を使用する

高度な認証情報戦略 (Secrets Manager、Lambda、または IAM ロール) を選択すると、AWS セキュリティエージェントはサービスロールを使用して、設定された AWS リソースから直接認証情報を取得します。エージェントスペースのログインプロンプトを使用して、これらの認証情報をアプリケーションに適用する方法をエージェントに提供します。例えば、どのログイン URL に移動するか、どのフォームフィールドに入力するかなどです。

> **Note**
>
> Secrets Manager のシークレットと Lambda 関数は、AWS セキュリティエージェントのセットアップと同じ AWS アカウントに存在する必要があります。クロスアカウント認証情報は現在サポートされていません。

1. 詳細設定を選択します。
2. ユーザーアクセス戦略ドロップダウンで、次のいずれかを選択します。

エージェントが引き受けることができる IAM ロールを選択する

このオプションは、AWS Cognito、IAM 認証付き API Gateway、またはその他の AWS ネイティブ認証システムを使用するアプリケーションに使用します。IAM ロールには、AWS セキュリティエージェントがロールを引き受けることを可能にする信頼関係と、アプリケーションの認証システムにアクセスするためのアクセス許可が必要です。

接続された AWS Secrets Manager から静的認証情報を選択する

このオプションを使用して、暗号化、ローテーション、アクセス監査を使用して AWS Secrets Manager から認証情報を安全に取得します。

IAM ロールには `secretsmanager:GetSecretValue`および アクセス`secretsmanager:DescribeSecret`許可が必要です。

エージェントは Secrets Manager から直接シークレット値を取得します。エージェントスペースのログインプロンプトを使用して、アプリケーションのログインフローにこれらの認証情報を適用する方法をエージェントに指示します。例えば、どの URL に移動するか、どのフォームフィールドに入力するかなどです。エージェントがログインプロンプトで指定した手順に従って形式を解釈するため、任意の形式を使用してシークレットを保存できます。

たとえば、エージェントが https://example.com/login でユーザー名/パスワードログインフォームを送信する場合、 フィールド`username`と `password`フィールドを使用してシークレットを JSON としてフォーマットできます。アプリケーションで TOTP ベースの 2FA が必要な場合は、`totpSecret`TOTP シークレットのフィールドを直接含めるか、完全な URI `otpauth://totp/` を含めます。

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

次に、認証手順 を設定します。アクセス URL を `https://example.com` (またはターゲットエンドポイントのリストから選択された他の URL) に設定します。エージェントスペースのログインプロンプトに「https://example.com/login に移動し、指定されたユーザー名とパスワードをフォームに入力します」と入力します。

別の例として、代わりに HTTP ヘッダーに提供される API キーがある場合は、プレーンテキストとして保存できます。

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

次に、認証手順 を設定します。エージェントスペースのログインプロンプトにX-API-Key ヘッダーをすべてのリクエストで指定された API キーに設定します」と入力します。

> **Important**
>
> TOTP ベースの 2FA のみがサポートされています。SMS、E メール、プッシュ通知、ハードウェアキー、OAuth 認証はサポートされていません。

利用可能な Lambda 関数を選択して認証情報を動的に取得する

このオプションは、複雑な認証システム、動的な認証情報生成、または外部 ID プロバイダーとの統合に使用します。

IAM ロールには アクセス`lambda:InvokeFunction`許可が必要であり、関数は 30 秒以内に完了する必要があります。

ペネトレーションテストが実行されると、AWS セキュリティエージェントは AWS Lambda 関数を同期的に呼び出し、ペネトレーションテストと解決される特定の認証情報を識別するイベントに合格します。

```
{
  "pentest_arn": "arn:aws:securityagent:us-west-2:111122223333:pentest/pt-
abcd1234-5678-90ab-cdef-EXAMPLE11111",
  "actor_identifier": "Credential1"
}
```

- `pentest_arn` – 認証情報が解決されるペネトレーションテストの Amazon リソースネーム (ARN)。これを使用して、関数内のリクエストの範囲指定、認可、または監査を行います。
- `actor_identifier` – コンソールでこの認証情報に割り当てた認証情報名 (例: `Credential1`)。これを使用して、1 つの関数が複数の認証情報を提供するときに正しい認証情報を返します。

エージェントは、関数の出力を認証情報として直接使用します。エージェントスペースのログインプロンプトを使用して、AWS Secrets Manager と同じように、アプリケーションにこれらの認証情報を適用する方法をエージェントに指示します。Lambda 関数の出力とサポートされている認証タイプのフォーマット方法の例the section called “接続された AWS Secrets Manager から静的認証情報を選択する”については、「」を参照してください。

複数の認証情報を設定する

さまざまなユーザーロールまたは認証システムをテストするには:

1. 別の認証情報を追加する を選択します。
2. いずれかの入力方法を使用して追加の認証情報を設定します。
3. 認証情報を削除するには、認証情報セクションの「削除」を選択します。

ログインの最適化

ログインの最適化により、AWS セキュリティエージェントは以前のペンテスト実行からナビゲーションパターンを学習し、後続の実行に適用できます。これらのパターンには、ログインフローとマルチステップ認証ワークフローが含まれます。これにより、エージェントが認証方法の再検出に費やす時間が短縮され、スキャンが高速化され、テストカバレッジの一貫性が向上します。

ログイン最適化の仕組み

最初のペンテストの実行時に、エージェントは指定した認証情報を使用してアプリケーションのログインフローをナビゲートする方法を検出します。成功したナビゲーションステップを記録し、学習したログインワークフローをキャプチャするスキルファイルを生成します。

後続の実行では、エージェントは保存されたスキルファイルを読み取り、学習したナビゲーションパターンを直接適用し、検出フェーズをスキップします。つまり、次のようになります。

- 認証テストまでの時間を短縮 — エージェント は、ゼロから探索するのではなく、実証済みのナビゲーションパターンをすぐに適用します。
- より一貫した動作 — エージェント は代替手段を探索するのではなく、同じ実証済みのパスに従います。

> **Note**
>
> ログイン最適化は、実行内の最初のログインセッション中に学習します。同じ実行の後続のログインセッションは、最初のセッションで学習した内容の恩恵を受けます。今後の実行では、すべてのログインセッションが保存されたスキルの恩恵を受けます。

ログイン最適化を有効または無効にする

ログインの最適化はデフォルトで有効になっています。無効化または再有効化するには:

1. ペンテスト設定で、認証認証情報 - オプションセクションに移動します。
2. ログイン最適化の切り替えを見つけます。

3. ログインの最適化を有効にするには、トグルをオンにします。無効にするには、無効にします。

無効にすると、エージェントは実行のたびにログインフローをゼロから再検出します。以前に学習したナビゲーションスキルは保持され、機能を再度有効にすると再び適用されます。

> **Tip**
>
> アプリケーションのログインフローが大幅に変更された場合 (再設計されたログインページや新しい認証ステップなど) 、エージェントは次の実行時に学習したスキルを更新することで自動的に適応します。最適化を手動でリセットする必要はありません。

ペネトレーションテストの検出結果の修正

ペネトレーションテストの結果を表示するときは、AWS セキュリティエージェントの検出結果の修正をリクエストできます。プライベート GitHub リポジトリの場合、AWS セキュリティエージェントは提案された修正を含むプルリクエストを開きます。パブリックリポジトリの場合、修復はローカルに適用できるダウンロード可能な差分ファイルとして利用できます。

AWS マネジメントコンソールで検出結果の修正を有効にする必要があります。(「the section called “ユーザーが侵入テストとコードレビューの検出結果の修正を開始できるようにする”」を参照してください。) ユーザーは、AWS セキュリティエージェントウェブアプリから特定の検出結果の修正を開始できます。

前提条件

開始する前に、以下があることを確認してください。

- 完了した、または進行中のペネトレーションテストの実行
- AWS Security Agent ウェブアプリケーションへのアクセス
- アプリケーションのアーキテクチャとセキュリティ要件に精通していること

ステップ 1: 自動修復を有効または無効にする

侵入テストを作成または変更するときに、コード修復オプションを設定できます。自動修復を有効にすると、エージェントがペンテスト中に検出結果を確認した場合、AWS セキュリティエージェントは自動的に関連する GitHub リポジトリの修復を試みます。コード修復を手動で開始することもでき

ます。侵入テストの詳細を編集するビューで、自動コード修復セクションで、コード修復を有効または無効にします。

ステップ 2: コード修復用のリポジトリを選択する

1. 最後のステップまで次へ 追加の学習リソースを選択します。
2. リソースから選択を選択します。
3. GitHub リポジトリを選択します。
4. コード修復に使用するリポジトリを選択します。
5. 侵入テストを保存します。
6. 正常に関連付けられたリポジトリは、「浸透テスト学習リソース」タブに表示されます。

ステップ 3: 侵入テストを開始し、検出結果を表示する

侵入テストを実行して検出結果を検出します。詳細については、「the section called “ペネトレーションテストの結果を確認する”」を参照してください。

ステップ 4: コード修復を開始して表示する

1. 検出結果に移動します。
2. 自動コード修復を有効にした場合、AWS セキュリティエージェントが検出結果を確認すると、コード修復が開始されます。
3. コード修復を手動で開始する場合は、コード修復ボタンを選択します。
4. 結果のコード修復セクションで、コード修復ステータスとプルリクエストへのリンクを表示できます。GitHub リポジトリがパブリックの場合、コード修復はプルリクエストの代わりにダウンロード可能なファイルとして利用できます。を実行して`git apply /path/to/code_remediation_changes.diff`、変更をローカルでリポジトリに適用できます。

セキュリティとリファレンス

AWS セキュリティエージェントのセキュリティガイダンス、サービスクォータ、およびドキュメント履歴。

AWS セキュリティエージェントのセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS セキュリティエージェントを実行するインフラストラクチャを保護する AWS 責任があります。これには、サービスインフラストラクチャ、AI モデル、侵入テストエージェントが含まれます。「AWS 」 コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。
- クラウド内のセキュリティ - お客様は以下の事項について責任を負います。
  - IAM ポリシーとアクセス許可による AWS セキュリティエージェントへのアクセスの管理
  - 設計ドキュメント、コードリポジトリ、侵入テスト用のアプリケーション URLs など、サービスに提供するコンテンツの保護
  - モニタリングするリポジトリとアプリケーションの設定
  - サービスによって提供されるセキュリティ検出結果の確認と対応
  - 提供された修復ガイダンスに基づくアプリケーションの保護
  - お客様のデータの機密性、企業の要件、該当する法律および規制

このドキュメントは、AWS セキュリティエージェントを使用する際の責任共有モデルの適用方法を理解するのに役立ちます。

AWS セキュリティエージェントと AI によるペネトレーションテストのセキュリティ上の考慮事項

AWS セキュリティエージェントは、すべての環境の開発ライフサイクルを通じてアプリケーションをプロアクティブに保護するフロンティアエージェントです。要件に合わせてカスタマイズされた自

動セキュリティレビューを実施し、セキュリティチームがレビュー中に自動的に検証される標準を一元的に定義します。セキュリティエージェントは、アプリケーションに合わせてカスタマイズされたオンデマンドのペネトレーションテストを実行し、検証済みのセキュリティリスクを検出して報告します。このアプローチは、包括的なセキュリティカバレッジを提供しながら、開発速度に合わせてアプリケーション全体にセキュリティの専門知識を拡張します。セキュリティを設計からデプロイまで統合することで、脆弱性を早期かつ大規模に防止できます。

セキュリティチームは、承認された認可ライブラリ、ログ記録標準、データアクセスポリシーという組織のセキュリティ要件を AWS コンソールで一度定義します。AWS セキュリティエージェントは、開発中にこれらのセキュリティ要件を自動的に適用し、アーキテクチャドキュメントとコードを標準に照らして評価し、違反を検出したときに特定のガイダンスを提供します。これにより、チーム間で一貫したセキュリティ適用が提供され、開発速度に合わせてレビューがスケーリングされます。

デプロイの検証のために、AWS セキュリティエージェントは侵入テストを定期的にボトルネックからオンデマンド機能に変換します。セキュリティチームは、ターゲット URLs、認証の詳細、ソースコード、ドキュメントを提供します。AWS セキュリティエージェントは、アプリケーションの深い理解を深め、高度な攻撃チェーンを実行して脆弱性を検出および検証し、チームが必要に応じてテストできるようにします。

主な機能

AWS セキュリティエージェントは、開発ライフサイクル全体にわたる包括的なセキュリティ機能を提供します。

セキュリティレビューの設計

AWS セキュリティエージェントは、設計ドキュメントに関するオンデマンドのセキュリティフィードバックを提供し、コードが記述される前に組織のセキュリティ要件への準拠を評価します。セキュリティチームは、ウェブアプリケーションを介して設計ドキュメントをアップロードします。このアプリケーションでは、エージェントはセキュリティ要件と照らし合わせてそれらを分析し、修復ガイダンスを使用して検出結果を表示します。これにより、数時間にわたる手動レビューが集中分析に迅速化され、修復が最も効率的である場合にチームがセキュリティ上の懸念に対処できるようになります。

コードセキュリティレビュー

AWS セキュリティエージェントは、プルリクエストまたはアップロードされたコードを分析して、組織のセキュリティ要件や、入力検証の欠落や SQL インジェクションリスクなどの一般的なセキュリティ上の問題がないかを確認します。エージェントは、コードリポジトリプラットフォーム内で直

接修復ガイダンスを提供します。セキュリティチームは、重要な問題の監視を維持しながら、モニタリングするリポジトリを設定し、すべてのコードベースで評価をスケーリングします。

オンデマンドのペネトレーションテスト

AWS セキュリティエージェントは、カスタマイズされた複数ステップの攻撃シナリオを通じて検証されたセキュリティの脆弱性を検出してレポートするオンデマンドの侵入テストを提供します。AWS セキュリティエージェントは、提供されたドキュメントと認証情報からアプリケーションコンテキストを開発する特殊な AI エージェントをデプロイし、高度な攻撃チェーンを実行して、従来のツールが見逃す複雑な脆弱性を特定します。影響分析、再現可能な攻撃パス、ready-to-implementコード修正を含む検出結果を文書化し、ペネトレーションテストを数週間から数時間に高速化し、アプリケーションポートフォリオ全体で検証をスケーリングします。

よくある質問

セキュリティとコントロール

AWS セキュリティエージェントは、システムへのアクセスをどのように認証および維持しますか?

ペネトレーションテストは、実行時にユーザーのシステムに認証できる AWS セキュリティエージェントの唯一の機能です。AWS セキュリティエージェントは、ペンテストを開始する前に、静的なユーザー名とパスワード認証情報 (Secrets Manager に保存) または認証情報ベンダー (Lambda 関数として) の形式の認証情報を構成として受け入れます。これらの認証情報は、ペンテストのライフサイクルを通じてユーザーのシステム/アプリケーションの通常の機能を実行するために使用されます。ペンテストの目的で、適切な範囲のアクセス許可を持つ新しい認証情報を作成することをお勧めします。

ユーザーは、意図しないシステムへの影響を防ぐためにテストの範囲と深さを制御できますか?

AWS セキュリティエージェントを使用すると、お客様はエンドポイントで調査する脆弱性の特定のカテゴリを選択できます。ユーザーはout-of-scopeURLs を指定して、AWS セキュリティエージェントがそれらのターゲットに対して侵入テストを実行できないようにすることができます。https://docs.aws.amazon.com/securityagent/latest/userguide/perform-penetration-test.html

AWS セキュリティエージェント自体がセキュリティリスクをもたらす可能性がありますか?

AWS セキュリティエージェントは、セキュリティリスクを検出するように指示されますが、意図的に影響を最小限に抑えたペイロード (SQL インジェクション攻撃が検出されたときにテーブルを削除するのではなく、SQL バージョンを抽出するなど) を使用するように指示されます。AWS セキュリ

ティエージェントは、ターゲットアプリケーションに対して過剰な負荷を発生させるなどのリスクのある動作を防ぐために、決定的なガードレールに制限されています。ガードレールが導入されている間は、意図しないビジネスロジックインタラクションや明白でないビジネスロジックインタラクションが発生する可能性があるため、本番稼働前の環境に対して侵入テストを行うことを常にお勧めします。

AWS セキュリティエージェントはどのようなデータを収集し、どこに保存しますか?

AWS セキュリティエージェントを使用すると、ユーザーはアーティファクトをアップロードして、テスト対象のアプリケーションに関するコンテキストを提供できます。データ保護の詳細については、「」を参照してくださいthe section called “データ保護”。AWS セキュリティエージェントは、推論リクエストを処理するために、地理的に最適なリージョンを自動的に選択します。これにより、利用可能なコンピューティングリソース、モデルの可用性を最大化し、最高のカスタマーエクスペリエンスを実現します。データはリクエストが発生したリージョンにのみ保存されますが、入力プロンプトと出力結果はそのリージョン外で処理される場合があります。すべてのデータは Amazon の安全なネットワーク経由で暗号化されて送信されます。詳細については、「クロスリージョン推論」を参照してください。

エンドポイントに対する不正なテストをブロックするには、どのようなコントロールがありますか?

ペンテストのターゲット URLsとして指定されたエンドポイントには、所有権の尺度として DNS 検証または HTTP 検証が必要です。AWS セキュリティエージェントは、エンドポイントの DNS に TXT レコードを追加するか、所有権の証明として HTTP Route 戻り検証文字列を公開するようにお客様に求めます。所有権の証明を証明した後でのみ、ユーザーはペンテストに進むことができます。ターゲット外の URLs およびアクセス可能な URLsへのリクエストは、ネットワークによってブロックされます。

お客様は、侵入テスト活動の影響を受ける可能性のあるすべてのシステムをテストするための適切な認可があることを確認する責任があります。AWS セキュリティエージェントのすべての使用は、AWS 適正使用ポリシー (https://aws.amazon.com/aup/) に準拠する必要があります。

AWS セキュリティエージェントを使用して不正使用をブロックして報告するにはどうすればよいですか?

AWS セキュリティエージェントは、リクエストを継続的にモニタリングし、ターゲット URLsの外部にある URLs。AWS セキュリティエージェントを使用してサードパーティーエンドポイントで不正なテストを実行しようとするなど、不正使用が検出された場合、アカウントで進行中のペストはすべて終了します。お客様は、AWS サポートまたは AWS アカウントチームに連絡してサポートを受けることができます。

AWS セキュリティエージェントはペンテストワークフローを置き換えることができますか?

AWS セキュリティエージェントは専門的な侵入テストサービスではなく、AWS セキュリティエージェントをセキュリティレビューワークフローに統合することをお勧めします。AWS セキュリティエージェントは、ペンテストの専門家とやり取りするのが時期尚早、非実用的、または頻繁に再評価する必要がある場合に、ソフトウェアライフサイクルの開発段階で、ペネトレーションテストへのアクセシビリティをオンデマンドで提供できます。セキュリティプロフェッショナルは、AWS セキュリティエージェントからの検出結果を確認して、検証、説明、新しい新しい検出結果の拡張を行うことができます (存在する場合) 。

ユーザーは、異なるチームメンバーにロールベースのアクセスコントロール (RBAC) を設定できますか?

はい。AWS Security Agent は AWS IAM Identity Center と統合されているため、管理者は AWS Security Agent ウェブアプリケーションにアクセスできるチームメンバーを管理できます。これにより、ユーザーは設計レビューとペンテストを作成、管理、表示できます。

機能のテスト

AWS Security Agent はどのような種類の脆弱性を検出できますか?

AWS セキュリティエージェントは、ウェブアプリケーションの OWASP トップ 10 の脆弱性を検出します。AWS セキュリティエージェントは、以下に概説するテストに含めたり除外したりできる特定のリスクタイプを提供します。検出結果は、これらのリスクカテゴリ内、またはこれらのリスクカテゴリの組み合わせからのリードに従うことで検出された新しい検出結果から発生する可能性があります。

- 任意のファイルのアップロード
  - 任意ファイルアップロードは、アプリケーションとユーザーの安全を維持するために、アプリケーションが偽ファイルや悪意のあるファイルを回避できることを確認します。
- コードインジェクション
  - Code Injection は、アプリケーションによって解釈/実行されるコードの挿入で構成される攻撃タイプの一般的な用語です。
- コマンドインジェクション
  - コマンドインジェクションは、脆弱なアプリケーションを介してホストオペレーティングシステムで任意のコマンドを実行することを目標とする攻撃です。
- クロスサイトスクリプティング (XSS)

  - クロスサイトスクリプティング (XSS) 攻撃は、悪意のあるスクリプトが無害で信頼できるウェブサイトに挿入されるインジェクションの一種です。
- 安全でない直接オブジェクトのリファレンス
  - 安全でないダイレクトオブジェクトリファレンス (IDOR) は、ユーザーが指定した入力に基づいてアプリケーションがオブジェクトに直接アクセスする場合に発生します。
- JSON ウェブトークンの脆弱性
  - JWTsは、アプリケーションと基盤となるライブラリの両方で、脆弱性の一般的な原因です。
- ローカルファイルの包含
  - ファイル包含の脆弱性により、攻撃者はファイルを含めることができ、通常はターゲットアプリケーションに実装されている「動的ファイル包含」メカニズムを悪用します。
- パストラバーサル
  - パストラバーサル攻撃 (ディレクトリトラバーサルとも呼ばれます) は、ウェブルートフォルダの外部に保存されているファイルとディレクトリへのアクセスを目的としています。
- 特権エスカレーション
  - 特権エスカレーションは、ユーザーが通常許可されているよりも多くのリソースや機能にアクセスでき、そのような昇格や変更がアプリケーションで防止されるべきだった場合に発生します。
- サーバー側のリクエスト偽造 (SSRF)
  - サーバー側のリクエスト偽造 (SSRF) は、攻撃者がサーバー上の機能を悪用して内部リソースを読み取りまたは更新する可能性がある場合に発生します。
- サーバー側のテンプレートインジェクション
  - サーバー側のテンプレート挿入の脆弱性 (SSTI) は、ユーザー入力が安全でない方法でテンプレートに埋め込まれ、サーバーでリモートコードが実行された場合に発生します。
- SQL インジェクション
  - SQL インジェクション攻撃は、クライアントからアプリケーションへの入力データを介した SQL クエリの挿入または「挿入」で構成されます。
- XML 外部エンティティ
  - XML 外部エンティティ攻撃は、XML 入力を解析するアプリケーションに対する攻撃の一種です。この攻撃は、外部エンティティへの参照を含む XML 入力が弱い設定の XML パーサーによって処理された場合に発生します。

AWS セキュリティエージェントはどのような認証方法をサポートしていますか?

AWS セキュリティエージェントは、OAuth や JWT などの一般的な認証方法をサポートしています。詳細については、 のドキュメントを参照してください。

AWS セキュリティエージェントはレート制限とサービス拒否 (DOS) 防止をどのように処理しますか?

AWS セキュリティエージェントには、DOS を含むテスト対象のエンドポイントの中断や削除を防ぐためのガードレールがあります。予期しないトラフィックパターンを検出して処理するための内部速度制御があります。

AWS セキュリティエージェントは REST API と GraphQL APIs の両方をテストできますか?

はい、AWS セキュリティエージェントは API エンドポイントをテストできます。AWS セキュリティエージェントがテスト対象の各 API の形状と機能をより適切に把握できるように、API ドキュメントを追加の学習リソースとして提供することをお勧めします。

AWS セキュリティエージェントがすべての重要なアプリケーションロジックとエンドポイントをカバーしていることをユーザーが確認するにはどうすればよいですか?

AWS セキュリティエージェントは、ターゲットアプリケーション (複数可) を広範に最初に探索し、エクスプロイトを試みる前に正常に実行しようとします。これにより、実行時にアプリケーションに対する実用的な理解を構築し、重要なアプリケーションロジックとエンドポイントを検出できます。その確率的性質を考慮すると、AWS セキュリティエージェントは、ターゲットアプリケーションのすべての重要なアプリケーションとエンドポイントを検出してテストする保証はありません。AWS Security Agent ウェブアプリケーションは、検出されたすべてのエンドポイントと、侵入テストログで実行されたアクションを可視化します。

精度と信頼性

AWS セキュリティエージェントは、報告する前に検出結果をどのように検証しますか?

AWS セキュリティエージェントは、決定論的な検証機能を使用して、報告された結果を検証します。決定論的検証を使用できないリスクタイプでは、AWS セキュリティエージェントは検出結果の有効性を信頼するために検出結果ステップを個別にリプレイします。AWS セキュリティエージェントは、信頼度の高い検出結果または中程度の検出結果のみをレポートし、デフォルトで未検証の検出結果を非表示にします。

AWS Security Agent はカスタムアプリケーションロジックに適応できますか?

AWS セキュリティエージェントは、オプションでソースコード、脅威モデル、設計ドキュメント、API ドキュメントを追加の学習リソースとして受け入れ、ペンテストのライフサイクルで使用されるターゲットアプリケーションのユーザー主導のコンテキストを取得します。

ユーザーは実行前に AWS セキュリティエージェントのテスト方法を確認できますか?

現在、AWS セキュリティエージェントの一連のアクションをプレビューする方法はありません。AWS セキュリティエージェントプランは、ターゲットアプリケーションの探索に基づいて本質的に動的です。お客様は、侵入テストログを監視することで、AWS セキュリティエージェントをリアルタイムで監視できます。ログに無効または望ましくない軌跡が表示されている場合、お客様は進行中のペネテストの実行を停止できます。

統合とデプロイ

AWS セキュリティエージェントは、セキュリティツール (SIEM、脆弱性管理) または CI/CD パイプラインと統合されていますか?

AWS セキュリティエージェントは、既存のセキュリティツールや CI/CD パイプラインと統合されません。

AWS セキュリティエージェントは環境固有の設定をどのように処理しますか?

AWS セキュリティエージェントは、特定の IAM ロール、VPCs 内、お客様が指定したアプリケーション関連の認証情報、およびターゲットアプリケーションのソースコード参照として Github ソースリポジトリを使用して実行するように設定できます。

AWS セキュリティエージェントはエアギャップまたは隔離された環境で実行できますか?

AWS セキュリティエージェントは、アウトバウンドインターネットアクセスを持たないものを含め、VPCs への接続を持つように設定できます。

複数のチームメンバーが同時にテストを実行できますか?

AWS セキュリティエージェントは、テストを開始するユーザーに関係なく、アカウントごとに 5 回の同時ペテスト実行をサポートします。お客様は、最大 100 のエージェントスペースと 1,000 の Pentest プロジェクトを作成できます。

運用上の影響

テストされたシステムのパフォーマンスにはどのような影響がありますか?

AWS セキュリティエージェントには、テスト対象のエンドポイントの中断や削除を防ぐためのガードレールがあります。これには、AWS セキュリティエージェントがエンドポイントに対して実行できる呼び出しの数の速度制御が含まれます。システムまたはテスト対象のエンドポイントでは、ペンテストアクティビティが原因でトラフィックが増加し、潜在的なモニタリングアラートがトリガーされることを想定する必要があります。AWS セキュリティエージェントまたはペンテストアクティビティは、本番稼働前の環境でのみ実行することをお勧めします。

ユーザーは AWS セキュリティエージェントをスケジュールまたはスロットリングできますか?

AWS セキュリティエージェントには、パブリック APIsやペンテストの実行をスケジュールする機能はありません。また、AWS セキュリティエージェントは、ペンテスト実行の開始時にターゲットエンドポイントへのリクエストの同時実行制御を提供しません。AWS セキュリティエージェントがターゲットエンドポイントで問題を引き起こしている場合、お客様は進行中のペンテスト (複数可) を停止できます。

完全なセキュリティ評価の一般的な期間はどれくらいですか?

各ペンテストのランタイムは、ターゲットアプリケーションの幅と、評価するように設定されているリスクタイプによって異なります。ほとんどのペンテストの実行は 16 時間以内に完了します。

AWS セキュリティエージェントの AWS 管理ポリシー

AWS 管理ポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS 管理ポリシーは、多くの一般的なユースケースにアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS のお客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。AWS が AWS 管理ポリシーで定義されたアクセス許可を更新すると、その更新はポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。AWS は、新しい AWS サービスが起動されたとき、または既存のサービスで新しい API オペレーションが利用可能になったときに、AWS 管理ポリシーを更新する可能性が最も高いです。

詳細については、IAM ユーザーガイドの AWS 管理ポリシーを参照してください。

AWS 管理ポリシー: SecurityAgentWebAppAPIPolicy

Security Agent Web Application API を操作するアクセス許可を付与します。このポリシーにより、ユーザーは自動セキュリティ侵入テストの設定と実行、テスト実行の管理、セキュリティ検出結果の表示、Security Agent リソースへのアクセスを行うことができます。このポリシーは、レガシーエージェントインスタンスリソースタイプと特定のレガシー IAM アクションを参照します。

アクセス許可の詳細

このポリシーは、以下のセキュリティテストコントロールサービス (securityagent:*) とやり取りするアクセス許可を付与します。

- Pentest Management: ペネトレーションテストとその実行ジョブの作成、更新、削除、一覧表示
- セキュリティ検出結果: 関連するコンテンツやメタデータなど、完了したテストのセキュリティ検出結果を表示、説明、更新します。
- タスク管理: コードレビュータスクとドキュメントレビュータスクを一覧表示および取得する
- リソース検出: エージェントインスタンス、アーティファクト、統合、検出されたエンドポイントを一覧表示および表示する
- テスト実行: リアルタイムモニタリング機能を使用してペンテスト実行を開始および停止する
- コード修復: セキュリティ検出結果の自動コード修復を開始する

JSON ポリシードキュメントの最新バージョンを表示するには、AWS マネージドポリシーリファレンスガイドの SecurityAgentWebAppAPIPolicy を参照してください。

AWS 管理ポリシー: AWSSecurityAgentWebAppPolicy

Security Agent Web Application API を操作するアクセス許可を付与します。このポリシーにより、ユーザーは自動セキュリティ侵入テストの設定と実行、テスト実行の管理、セキュリティ検出結果の表示、Security Agent リソースへのアクセスを行うことができます。

アクセス許可の詳細

このポリシーは、以下のセキュリティテストコントロールサービス (securityagent:*) とやり取りするアクセス許可を付与します。

- ペネテスト管理: ペネトレーションテストとそのジョブの作成、更新、削除、一覧表示
- セキュリティ検出結果: 関連するコンテンツやメタデータなど、完了したテストのセキュリティ検出結果を表示、説明、更新します。

- タスク管理: コードレビューおよび設計レビュータスクを一覧表示および取得する
- リソース検出: エージェントスペース、アーティファクト、統合、検出されたエンドポイントを一覧表示および表示する
- テスト実行: リアルタイムモニタリング機能を使用してペンテストジョブを開始および停止する
- コード修復: セキュリティ検出結果の自動コード修復を開始する

JSON ポリシードキュメントの最新バージョンを表示するには、AWS AWSSecurityAgentWebAppPolicy」を参照してください。

AWS セキュリティエージェントの AWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始してからの AWS セキュリティエージェントの AWS 管理ポリシーの更新に関する詳細を表示します。

この特定のドキュメントページへのすべてのソースファイルの変更通知を受け取るには、RSS リーダーを使用して次の URL をサブスクライブできます。

変更	説明	日付
AWSSecurityAgentWebAppPolicy にアクセス許可を追加しました。	新しいDesignReviewFeedback リソースタイプの TargetDomain および リソースのアクセス許可を追加しました。	2026 年 3 月 31 日
新しい管理ポリシー AWSSecurityAgentWebAppPolicy を追加しました。	新しい AgentSpace リソースタイプの管理ポリシーと IAM アクション名の変更を追加AWSSecurityAgentWebAppPolicy しました。	2026 年 2 月 9 日
SecurityAgentWebApAPIPolicy にアクセス許可を追加しました。	セキュリティ検出結果の自動コード修復を開始securityagent:StartCodeRemediation できるように を追加しました。	2026 年 1 月 20 日

変更	説明	日付
SecurityAgentWebAppAPIPolicy にアクセス許可を追加しました。	コンソールで securityagent:BatchGetSecurityTestContentMetadata イメージを表示できるように を追加しました。	2025 年 12 月 5 日
AWS セキュリティエージェントが変更の追跡を開始しました。	AWS セキュリティエージェントは、AWS 管理ポリシーの変更の追跡を開始しました。	2025 年 12 月 2 日

AWS セキュリティエージェントのデータ保護

AWS 責任共有モデルは、AWS セキュリティエージェントのデータ保護に適用されます。このモデルで説明したように、AWS は、すべての AWS クラウドを実行するグローバルインフラストラクチャを保護します。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定および管理タスクについても責任を負います。欧州でのデータ保護の詳細については、AWS セキュリティブログの「AWS の責任共有モデルと GDPR」のブログ記事を参照してください。データ保護の目的で、AWS アカウントの認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- AWS CloudTrail を使用して API とユーザーアクティビティログを設定します。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、AWS CloudTrail ユーザーガイドの「CloudTrail 証跡の使用」を参照してください。 CloudTrail
- AWS 暗号化ソリューションを、AWS サービス内のすべてのデフォルトのセキュリティ管理と一緒に使用します。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。

- コマンドラインインターフェイスまたは API を介して AWS にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDKs。タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

AWS セキュリティエージェントは、デフォルトで AWS マネージド暗号化キーを使用して保管中のすべてのデータを暗号化します。これには、以下が含まれます。

- 設計ドキュメントとコード – セキュリティレビューのために提供するすべての設計ドキュメント、コードリポジトリ、およびアプリケーションアーティファクトは、AES-256 暗号化を使用して暗号化されます。
- セキュリティ検出結果 – すべてのセキュリティ検出結果、脆弱性レポート、および修復の推奨事項は、保管時に暗号化されます。
- 設定データ – セキュリティ要件、カスタムポリシー、サービス設定は暗号化されます。
- 監査ログ – すべてのサービスアクティビティログと監査証跡は暗号化されます。

AWS セキュリティエージェントは、AWS Key Management Service (AWS KMS) を使用して暗号化キーを管理します。必要に応じて、カスタマーマネージドキーを使用してデータを暗号化し、リソースを保護する暗号化キーを完全に制御できます。詳細については、「the section called “カスタマーマネージドキー”」を参照してください。

転送中の暗号化

AWS セキュリティエージェントは、Transport Layer Security (TLS) 1.2 以降を使用して、転送中のすべてのデータを暗号化します。これは、以下に適用されます。

- API 通信 – アプリケーションと AWS セキュリティエージェント間のすべての API コールは、TLS 暗号化で HTTPS を使用します。

- コンソールアクセス – AWS セキュリティエージェントコンソールには HTTPS 経由でアクセスします。
- リポジトリ接続 – GitHub およびその他のコードリポジトリへの接続では、暗号化されたプロトコルが使用されます。
- エージェント通信 — AWS セキュリティエージェントサービスと侵入テストエージェント間のすべての通信は、暗号化されたチャネルを使用します。

キー管理

AWS セキュリティエージェントは、AWS Key Management Service (AWS KMS) を使用して暗号化キーを管理します。デフォルトでは、データは AWS マネージドキーを使用して暗号化されます。エージェントスペースや統合などのリソースを作成するときに、オプションでカスタマーマネージドキーを指定できます。詳細については、「the section called “カスタマーマネージドキー”」を参照してください。

ネットワーク間のトラフィックのプライバシー

AWS セキュリティエージェントは、パブリックインターネットを使用して、クラウドホスト型ソースコントロールプロバイダー (GitHub、GitLab、Bitbucket) および Confluence Cloud と通信します。

セルフホスト型プロバイダー (GitLab Self-Managed、GitHub Enterprise Server) の場合、Amazon VPC Lattice を使用してプライベート接続を設定して、すべてのトラフィックを AWS ネットワーク内に保持できます。詳細については、「the section called “プライベートにホストされたソースコントロールに接続する”」を参照してください。

デフォルト設定では、AWS セキュリティエージェントはパブリックインターネットを使用してアプリケーションに到達し、侵入テストを行います。オプションで、VPC を使用してアプリケーションにアクセスするように侵入テストを設定できます。詳細については、「the section called “エージェントをプライベート VPC リソースに接続する”」を参照してください。

クロスリージョンデータ処理

AWS セキュリティエージェントは、クロスリージョン推論を使用して、利用可能なコンピューティングリソースとモデルの可用性を最適化します。リクエストの送信元のリージョンによっては、入力プロンプトと出力結果を別のリージョンで処理することがあります。

- 米国東部 (バージニア北部) – `us-east-1`、米国西部 (オレゴン) – `us-west-2`、アジアパシフィック (シドニー) – `ap-southeast-2`、アジアパシフィック (東京) – `ap-northeast-1`、欧州 (フラ

ンクフルト) – eu-central-1、欧州 (アイルランド) – eu-west-1では、AWS セキュリティエージェントは地理的クロスリージョン推論を使用します。ほとんどの機能では、データ処理はリクエスト元の地理的境界 (米国、欧州、オーストラリア、日本など) 内にとどまります。コード修復の場合、オーストラリアと日本からのリクエストは欧州連合で処理されます。機能固有のルーティングの詳細については、クロスリージョン推論表を参照してください。

- アジアパシフィック (ムンバイ) – ap-south-1、アジアパシフィック (シンガポール) – ap-southeast-1、南米 (サンパウロ) – ではsa-east-1、AWS セキュリティエージェントはグローバルクロスリージョン推論を使用します。入力プロンプトと出力結果は、商用 AWS リージョンで処理される場合があります。

いずれの場合も、データはリクエストが発生したリージョンにのみ保存されます。クロスリージョンオペレーション中に送信されるすべてのデータは AWS ネットワークに残り、パブリックインターネットを経由しません。AWS リージョン間で転送中のデータは暗号化されます。

クロスリージョン推論は常に有効になっており、オプトアウトすることはできません。各機能のリクエストを処理するリージョンの詳細については、「」の「クロスリージョン推論」セクションを参照してくださいthe section called “セキュリティのベストプラクティス”。

データの削除

AWS セキュリティエージェントからデータを削除する場合:

- データは削除対象としてマークされ、サービス経由でアクセスできなくなります。
- データは 30 日以内にすべての AWS セキュリティエージェントシステムから削除されます。

データを削除するには

1. AWS コンソールで、AWS セキュリティエージェントに移動します。
2. 削除するデータ (セキュリティレビュー、検出結果、またはカスタム要件) を選択します。
3. [削除] を選択し、削除を確定します。

AWS セキュリティエージェントの Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS resources へのアクセスを安全に制御 AWS のサービス するのに役立つ です。 IAM 管理者は、誰を認証 (サインイン) し、誰に AWS セキュリティエージェントリソースの使用を許可する (アクセス許可を付与 AWS のサービス する) かを制御します。 IAM は、追加料金なしで使用できる です。

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、AWS セキュリティエージェントで行う作業によって異なります。

サービスユーザー – AWS セキュリティエージェントサービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が提供されます。さらに多くの AWS セキュリティエージェントの機能を使用して作業を行う場合は、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。

AWS セキュリティエージェントの機能にアクセスできない場合は、「」を参照してくださいthe section called “AWS セキュリティエージェントの ID とアクセスのトラブルシューティング”。

サービス管理者 - 社内の AWS セキュリティエージェントのリソースを担当している場合は、通常、AWS セキュリティエージェントへのフルアクセスがあります。サービスユーザーがどの AWS セキュリティエージェントの機能とリソースにアクセスするかを決めるのは管理者の仕事です。その後、 IAM 管理者にリクエストを送信して、サービスユーザーのアクセス許可を変更する必要があります。このページの情報を確認して、 の基本概念を理解します IAM。AWS セキュリティエージェント IAM で を使用する方法の詳細については、「」を参照してくださいthe section called “AWS セキュリティエージェントが と連携する方法 IAM”。

IAM 管理者 - IAM 管理者は、AWS セキュリティエージェントへのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。で使用できる AWS セキュリティエージェントのアイデンティティベースのポリシーの例を表示するには IAM、「」を参照してくださいthe section called “AWS セキュリティエージェントのアイデンティティベースのポリシーの例”。

アイデンティティを使用した認証

認証は、ID 認証情報 AWS を使用して にサインインする方法です。AWS アカウントのルートユーザー、、または IAM ロールを引き受けることで IAM ユーザー、認証 (にサインイン AWS) される必要があります。AWS では、IAM ロールを使用し、ルートユーザーを直接使用しないことをお勧めします。

ID ソースを通じて提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM アイデンティティセンター (IAM アイデンティティセンター) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインすると、管理者は以前に IAM ロールを使用して ID フェデレーションをセットアップしていました。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS マネジメントコンソール または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、AWS サインインユーザーガイドの「AWS アカウントにサインインする方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。 AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。推奨される方法を使用してリクエストに署名する方法の詳細については、AWS 全般のリファレンスの署名バージョン 4 の署名プロセスを参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供を要求される場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM アイデンティティセンター (AWS Single Sign-On) ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウント のルートユーザー

を初めて作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの完全なアクセス権を持つシングルサインインアイデンティティから始めます。このアイデンティティは root ユーザーと呼ばれ、AWS アカウントの作成に使用したメールアドレスとパスワードでのサインインによりアクセスされます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、ルートユーザーのみが実行できるタスク実行に使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「 アカウント管理リファレンスガイド」の「ルートユーザーの認証情報を必要とするタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、、AWS Directory Serviceアイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグルー

プのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「IAM Identity Center とは」を参照してください。 AWS IAM Identity Center (AWS Single Sign-On の後継) ユーザーガイドの「」を参照してください。

IAM ユーザー および グループ

IAM ユーザー は、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を持つ IAM ユーザー ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、 で長期的な認証情報を必要とする特定のユースケースがある場合は IAM ユーザー、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、 のコレクションを指定する ID です IAM ユーザー。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が簡単になります。たとえば、IAMAdmins という名前のグループがあり、そのグループに IAM リソースを管理するアクセス許可を付与できます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、IAM ユーザーガイドの IAM ユーザー 「 (ロールではなく) を作成するタイミング」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは に似ていますが IAM ユーザー、特定のユーザーに関連付けられていません。 IAM ロールを切り替える AWS マネジメントコンソール ことで、 で ロールを一時的に引き受けることができます。ロールを引き受けるには、 AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールの使用方法の詳細については、IAM ユーザーガイドの IAM「ロールの使用」を参照してください。

IAM 一時的な認証情報を持つ ロールは、以下の状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーテッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーテッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティーアイデンティティプロバイダー向けロールの

作成」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。IDが認証後にアクセスできるものをコントロールするために、IAM アイデンティティセンター は権限セットを IAMのロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center (AWS Single Sign-On の後継) ユーザーガイド」の「アクセス許可セット」を参照してください。

- 一時的な IAM ユーザー アクセス許可 – は、 IAM ロールを引き受けて、特定のタスクに対して異なるアクセス許可を一時的に引き受け IAM ユーザー ることができます。
- クロスアカウントアクセス – IAM ロールを使用して、別のアカウントのユーザー (信頼されたプリンシパル) が自分のアカウントのリソースにアクセスすることを許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、IAM ユーザーガイドの IAM 「ロールとリソースベースのポリシーの違い」を参照してください。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。たとえば、サービスで呼び出しを行うと、そのサービスが でアプリケーションを実行 Amazon EC2 したり、オブジェクトを保存したりするのが一般的です Amazon S3。サービスは、呼び出し元のプリンシパルの許可、サービスロール、サービスリンクロールを使用してこれを行う場合があります。
  - プリンシパルアクセス許可 – IAM ユーザー または ロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。ポリシーによって、プリンシパルに許可が付与されます。一部のサービスを使用する際に、アクションを実行することで、別サービスの別アクションがトリガーされることがあります。この場合、両方のアクションを実行するためのアクセス許可が必要です。
  - サービスロール – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受ける IAM ロールです。 IAM 管理者は、内部からサービスロールを作成、変更、削除できます IAM。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。 IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。
- で Amazon EC2 実行されているアプリケーション – IAM ロールを使用して、 Amazon EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、 Amazon EC2 インスタンス内にアクセスキーを保存するよ

りも優先されます。 AWS ロールを Amazon EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルには ロールが含まれており、 Amazon EC2 インスタンスで実行されているプログラムが一時的な認証情報を取得できるようにします。詳細については、IAM ユーザーガイドの「 IAM ロールを使用して Amazon EC2 インスタンスで実行されているアプリケーションにアクセス許可を付与する」を参照してください。

IAM ロールを使用する方法については、IAM ユーザーガイドの (ユーザーではなく) IAM ロールを作成するタイミングを参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

すべての IAM エンティティ (ユーザーまたはロール) は、アクセス許可なしで始まります。デフォルトでは、ユーザーは何もできず、自分のパスワードを変更することすらできません。何かを実行する許可をユーザーに付与するには、管理者がユーザーに許可ポリシーをアタッチする必要があります。また、管理者は、必要な許可があるグループにユーザーを追加できます。管理者がグループに許可を付与すると、そのグループ内のすべてのユーザーにこれらの許可が付与されます。

IAM ポリシーは、オペレーションの実行に使用するメソッドに関係なく、アクションのアクセス許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS マネジメントコンソール、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

ID ベースのポリシーは、、 IAM ユーザーロール、グループなどの ID にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクショ

ン、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドのIAM 「ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、IAM ユーザーガイドのマネージドポリシーとインラインポリシーの比較を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、 Amazon S3 バケットなどのリソースにアタッチする JSON ポリシードキュメントです。サービス管理者は、これらのポリシーを使用して、特定のプリンシパル (アカウントメンバー、ユーザー、またはロール) がそのリソースに対して実行する条件およびアクションを定義することができます。リソースベースのポリシーはインラインポリシーです。マネージド型のリソースベースのポリシーはありません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、ロール) がリソースにアクセスする許可を持つかについて管理するポリシーのタイプです。ACLs はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式を使用しません。Amazon S3、 AWS WAF、 Amazon VPC は ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 – アクセス許可の境界は、アイデンティティベースのポリシーが IAM エンティティ (IAM ユーザー またはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として許可される範囲は、エンティティのアイデンティティベースのポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイドのIAM 「エンティティのアクセス許可の境界」を参照してください。

- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 AWS アカウントルートユーザーを含めてメンバーアカウントのエンティティに対するアクセス許可を制限します。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの SCP の仕組みを参照してください。
- セッションポリシー – セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果として得られるセッションの許可は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分です。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

AWS セキュリティエージェントが と連携する方法 IAM

IAM を使用して AWS セキュリティエージェントへのアクセスを管理する前に、AWS セキュリティエージェントで使用できる IAM 機能を確認してください。

IAM 機能	AWS セキュリティエージェントのサポート
the section called “AWS セキュリティエージェントのアイデンティティベースのポリシー”	はい
the section called “AWS セキュリティエージェント内のリソースベースのポリシー”	いいえ
the section called “AWS セキュリティエージェントのポリシーアクション”	はい

IAM 機能	AWS セキュリティエージェントのサポート
the section called “AWS セキュリティエージェントのポリシーリソース”	部分的
the section called “AWS セキュリティエージェントのポリシー条件キー”	はい
the section called “AWS セキュリティエージェントのアクセスコントロールリスト (ACLs)”	いいえ
the section called “AWS セキュリティエージェントによる属性ベースのアクセスコントロール (ABAC)”	なし
the section called “AWS セキュリティエージェントでの一時的な認証情報の使用”	はい
the section called “AWS セキュリティエージェントの転送アクセスセッション”	はい
the section called “AWS セキュリティエージェントのサービスロール”	いいえ
the section called “AWS セキュリティエージェントのサービスにリンクされたロール”	はい

AWS Security Agent およびその他の が と AWS のサービス 連携する方法の概要を把握するには IAM、IAM ユーザーガイドの「 AWS のサービス と連携する IAM」を参照してください。

AWS セキュリティエージェントのアイデンティティベースのポリシー

ID ベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否される条件を指定できます。プリンシパルはアタッチされているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用するすべての要素については、IAM ユーザーガイドのIAM 「JSON ポリシー要素リファレンス」を参照してください。

AWS セキュリティエージェントのアイデンティティベースのポリシーの例

AWS セキュリティエージェントのアイデンティティベースのポリシーの例を表示するには、「」を参照してくださいthe section called “AWS セキュリティエージェントのアイデンティティベースのポリシーの例”。

AWS セキュリティエージェント内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または AWS サービスを含めることができます。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウントにある場合、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、ID ベースのポリシーをエンティティにアタッチすることで許可を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

AWS セキュリティエージェントのポリシーアクション

アクションをサポート はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

IAM アイデンティティベースのポリシーの Action要素は、ポリシーによって許可または拒否される特定のアクションを記述します。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。このアクションは、関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS セキュリティエージェントのポリシーアクションは、アクションの前にプレフィックス を使用しますsecurityagent:。たとえば、AWS Security Agent CreateEnvironment API オペレーションを使用して環境を作成するアクセス許可を付与するには、ポリシーに securityagent:CreateEnvironmentアクションを含めます。ポリシーステートメントにはAction または NotAction 要素を含める必要があります。AWS セキュリティエージェントは、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには次のようにコンマで区切ります。

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

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、List という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

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

AWS セキュリティエージェントのポリシーリソース

ポリシーリソースのサポート: 一部

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントにはResource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これはリソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの許可をサポートしないアクションの場合はステートメントがすべてのリソースに適用されることを表示するワイルドカード (*) を使用します。

```
"Resource": "*"
```

一部の AWS Security Agent API アクションは、複数のリソースをサポートしています。たとえば、ListEnvironments API アクションを呼び出すときに複数の環境を参照できます。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

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

たとえば、AWS Security Agent 環境リソースには次の ARN があります。

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

ステートメントmy-environment-2で環境my-environment-1と を指定するには、次の ARNs の例を使用します。

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

特定のアカウントに属するすべての環境を指定するには、ワイルドカード (*) を使用します。

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

AWS セキュリティエージェントのポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (またはConditionブロック) を使用すると、ステートメントが有効になる条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントのアクセス許可が付与される前に、すべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。たとえば、リソースにその IAM ユーザー 名前がタグ付けされている場合にのみ、リソースへのアクセス IAM ユーザー 許可を付与できます。詳細については、IAM ユーザーガイドのIAM 「ポリシー要素: 変数とタグ」を参照してください。

AWS セキュリティエージェントは、独自の条件キーのセットを定義し、いくつかのグローバル条件キーの使用もサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

AWS セキュリティエージェントのアクセスコントロールリスト (ACLs)

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

AWS セキュリティエージェントによる属性ベースのアクセスコントロール (ABAC)

ABAC (ポリシー内のタグ) のサポート: なし

AWS セキュリティエージェントでの一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報を使用してサインインすると、一部の AWS サービスは機能しません。一時的な認証情報を使用する AWS サービスなどの詳細については、IAM ユーザーガイドの「IAM を使用する AWS のサービス」を参照してください。

ユーザー名とパスワード以外の方法で AWS マネジメントコンソールにサインインする場合は、一時認証情報を使用していることになります。例えば、会社の Single Sign-On (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時認証情報は、AWS CLI または AWS API を使用して手動で作成できます。作成後、これらの一時的な認証情報を使用して AWS にアクセスできます。AWS では、長期のアクセスキーを使用する代わりに、一時認証情報を動的に生成することを推奨しています。詳細については、の「IAM の一時的なセキュリティ認証情報」を参照してください。

AWS セキュリティエージェントの転送アクセスセッション

転送アクセスセッション (FAS) のサポート: あり

IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS サービスを呼び出すプリンシパルのアクセス許可を、リクエスト元の AWS サービスと組み合わせて使用して、ダウンストリームサービスにリクエストを行います。FAS リクエストは、サービスが他の AWS のサービスまたはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AWS セキュリティエージェントのサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの「AWS サービスにアクセス許可を委任するロールを作成する」を参照してください。

AWS セキュリティエージェントのサービスにリンクされたロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、AWS サービスにリンクされたサービスロールの一種です。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは AWS アカウントに表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

AWS セキュリティエージェントのアイデンティティベースのポリシーの例

デフォルトでは、 IAM ユーザー および ロールには AWS Security Agent リソースを作成または変更するアクセス許可はありません。また、 AWS マネジメントコンソール、 AWS CLI、または AWS API を使用してタスクを実行することはできません。 IAM 管理者は、必要な指定されたリソースに

対して特定の API オペレーションを実行するアクセス許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。管理者は、これらのアクセス許可を必要とする IAM ユーザー または グループにこれらのポリシーをアタッチする必要があります。

これらのサンプル JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの「JSON エディタを使用したポリシーの作成」を参照してください。

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内で誰かが AWS Security Agent リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「 AWS 管理ポリシー」または「職務機能用の AWS 管理ポリシー」を参照してください。
- 最小特権のアクセス許可を適用する – IAM ポリシーでアクセス許可を設定するときは、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用してアクセス許可を適用する方法の詳細については、IAM ユーザーガイドの「 のポリシーとアクセス許可 IAM」を参照してください。
- IAM ポリシーの条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションとリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイドのIAM「JSON ポリシー要素: 条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保する – ポリシーがポリシー IAM 言語 (JSON) と IAM ベストプラクティスに準拠するように、新規および既存のポリシー IAM Access Analyzer を検証します。 は、安全で機能的なポリシーの作成に役立つ 100 を超えるポリシーチェックと実用的な推奨事項 IAM Access Analyzer を提供します。詳細については、「IAM ユーザーガイド」のIAM Access Analyzer 「ポリシーの検証」を参照してください。

- 多要素認証 (MFA) を要求する – アカウントに IAM ユーザー または ルートユーザーを必要とするシナリオがある場合は、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA 保護 API アクセスの設定 を参照してください。

AWS セキュリティエージェントコンソールの使用

AWS セキュリティエージェントコンソールにアクセスするには、IAM プリンシパルに最小限のアクセス許可のセットが必要です。これらのアクセス許可により、プリンシパルは 内の AWS セキュリティエージェントリソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限されたアイデンティティベースのポリシーを作成すると、そのポリシーをアタッチしたプリンシパルに対してはコンソールが意図したとおりに機能しません。

IAM プリンシパルが引き続き AWS セキュリティエージェントコンソールを使用できるようにするには、独自の一意の名前でポリシーを作成します。ポリシーをプリンシパルにアタッチします。詳細については、IAM ユーザーガイド」の「ユーザーへの許可の追加」を参照してください。

ポリシーの詳細については、「」を参照してくださいthe section called “AWS 管理ポリシー: SecurityAgentWebAppAPIPolicy”。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行する API オペレーションと一致するアクションのみへのアクセスを許可します。

AWS セキュリティエージェントの ID とアクセスのトラブルシューティング

次の情報は、AWS セキュリティエージェント および の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます IAM。

AccessDeniedException

AWS API オペレーションを呼び出すAccessDeniedExceptionときに を受け取った場合、使用している IAM プリンシパル認証情報には、その呼び出しを行うために必要なアクセス許可がありません。

```
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
```

前のメッセージ例では、ユーザーには AWS Security Agent `CreateEnvironment` API オペレーションを呼び出すアクセス許可がありません。AWS セキュリティエージェントの管理者権限を IAM プリンシパルに付与するには、「」を参照してくださいthe section called “AWS セキュリティエージェントのアイデンティティベースのポリシーの例”。

IAM の詳細については、IAM ユーザーガイドの「ポリシーを使用して AWS リソースへのアクセスを制御する」を参照してください。

自分の 以外のユーザーに AWS セキュリティエージェントのリソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- AWS セキュリティエージェントがこれらの機能をサポートしているかどうかを確認するには、「」を参照してくださいthe section called “AWS セキュリティエージェントが と連携する方法 IAM”。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の IAM ユーザー の へのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにロールとリソースベースのポリシーを使用する方法の違いについては、IAM ユーザーガイドの IAM 「ロールとリソースベースのポリシーの違い」を参照してください。

インシデントへの対応

AWS セキュリティエージェントは、脆弱性が悪用される前に脆弱性を特定して防止するために設計されたプロアクティブセキュリティテストサービスです。このサービスは、事後対応型インシデント

の検出や対応ではなく、設計レビュー、コード分析、侵入テストによる予防的セキュリティ検証に焦点を当てています。

インシデント検知

AWS セキュリティエージェントは、インシデント検出機能を提供しません。このサービスは、開発中およびデプロイ前にアプリケーションの脆弱性を検証するプロアクティブなセキュリティテストツールとして機能します。ランタイムセキュリティモニタリングとインシデント検出には、Amazon GuardDuty、AWS Security Hub、Amazon CloudWatch などのサービスを使用します。

インシデントアラート

AWS セキュリティエージェントは、セキュリティインシデントのリアルタイムアラートを生成しません。このサービスは、設計レビュー、コード分析、または侵入テストエンゲージメントを完了した後、AWS コンソールを通じてセキュリティ検出結果を提供します。これらの検出結果は、アクティブなセキュリティインシデントではなく、テスト中に検出された潜在的な脆弱性を示しています。

インシデント修復

AWS セキュリティエージェントは、自動インシデント修復を提供しません。このサービスは、セキュリティの脆弱性を特定し、以下を含む修復ガイダンスを提供します。

- 特定された脆弱性の詳細な説明
- 検証済みの検出結果の再現可能なエクスプロイトパス
- 特定のコード修正と実装ガイダンス
- 検出された問題の影響分析

開発チームとセキュリティチームは、このガイダンスを使用して、本番環境に到達する前に脆弱性に手動で対処します。

インシデント対応アクティビティのサポート

AWS セキュリティエージェントはインシデント対応用に設計されていませんが、セキュリティチームは サービスを使用してインシデント後のアクティビティをサポートできます。

脆弱性の検証

セキュリティインシデント後、AWS セキュリティエージェントを使用して、他のアプリケーションや環境に同様の脆弱性が存在するかどうかをテストします。

セキュリティ体制の評価

インシデント修復の一環として実装されたセキュリティ改善を検証するために、侵入テストを実施します。

根本原因の分析

コードセキュリティレビュー機能を使用して、他のコードベースに同様の脆弱性が存在する可能性を特定します。

AWS セキュリティエージェントのコンプライアンス検証

AWS のサービスが特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスプログラムによる対象範囲内の AWS のサービス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

サードパーティーの監査報告書は、AWS Artifact を使用してダウンロードすることができます。詳細については、AWS Artifact のレポートのダウンロードを参照してください。

AWS のサービスを使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性、貴社のコンプライアンス目的、適用可能な法律および規制によって決まります。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービスが HIPAA の対象となるわけではありません。
- AWS コンプライアンスのリソース - このワークブックおよびガイド群には、貴社の所在地や属する業界に適用可能なものが含まれています。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、AWS のサービスを保護するためのベストプラクティスをまとめ、ガイダンスを複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティコントロールにマッピングします。

- AWS Config デベロッパーガイドの「AWS Config ルールでのリソースの評価」 – AWS Config サービスは、リソースの設定が内部規定、業界のガイドライン、規制にどの程度適合しているかを評価します。
- AWS Security Hub – この AWS サービスは、AWS 内のセキュリティ状態を包括的に表示します。Security Hub は、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – この AWS サービスは、疑わしいアクティビティや悪意のあるアクティビティがないか環境をモニタリングすることで、AWS アカウント、ワークロード、コンテナ、データに対する潜在的な脅威を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検出要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager – この AWS のサービスでは、AWS の使用を継続的に監査することで、リスクとコンプライアンスの管理方法を簡素化し、規制や業界標準への準拠を支援します。

AWS セキュリティエージェントの耐障害性

> **Note**
>
> AWS セキュリティエージェントは、米国東部 (バージニア北部) – us-east-1、米国西部 (オレゴン) – us-west-2、アジアパシフィック (ムンバイ) – ap-south-1、アジアパシフィック (シンガポール) – ap-southeast-1、アジアパシフィック (シドニー) – ap-southeast-2、アジアパシフィック (東京) – ap-northeast-1、欧州 (フランクフルト) – eu-central-1、欧州 (アイルランド) – eu-west-1、南米 (サンパウロ) – の各リージョンで利用できますsa-east-1。クロスリージョン推論を使用します。アジアパシフィック (ムンバイ) 、アジアパシフィック (シンガポール) 、南米 (サンパウロ) では、AWS セキュリティエージェントはグローバルクロスリージョン推論を使用して推論リクエストを商用 AWS リージョンにルーティングします。他のすべてのリージョンでは、地理的なクロスリージョン推論を使用して、データ処理を特定の地理的境界内に保ちます。AWS セキュリティエージェントはアプリケーションの開発中に使用されるツールであり、重要または顧客向けのインフラストラクチャとしてデプロイしないでください。

AWS の グローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心として構築されます。AWS リージョンには、低レイテンシー、高いスループット、そして高度の冗

長ネットワークで接続されている、複数の物理的に独立し隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS グローバルインフラストラクチャを参照してください。

AWS セキュリティエージェントのインフラストラクチャセキュリティ

マネージドサービスである AWS セキュリティエージェントは、AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、AWS Well-Architected フレームワークの「セキュリティ」を参照してください。

ネットワークの隔離

AWS セキュリティエージェントは、AWS コンソールと AWS セキュリティエージェントウェブアプリケーションを介してアクセスされるフルマネージドサービスです。サービスへのアクセスは、ID プロバイダーと統合できる AWS Identity and Access Management (IAM) または AWS IAM Identity Center を通じて制御されます。

AWS セキュリティエージェントは、AWS PrivateLink を搭載したインターフェイス VPC エンドポイントをサポートしているため、パブリックインターネットを経由することなく、VPC からサービス APIs にプライベートにアクセスできます。詳細については、「the section called “インターフェイス VPC エンドポイント”」を参照してください。

AWS セキュリティエージェントでは、ターゲットアプリケーションおよびコントロールプレーンオペレーションでペネトレーションテストを実行するためにインターネットアクセスが必要です。このサービスはテストに AWS マネージドインフラストラクチャを使用し、EC2 インスタンスやその他のコンピューティングリソースをアカウントにプロビジョニングしません。ペネトレーションテスト用に VPC アクセスを設定すると、セキュリティエージェントはサブネットに ENI を作成しますが、この ENI にはパブリック IP アドレスがありません。詳細については、「the section called “エージェントをプライベート VPC リソースに接続する”」を参照してください。

マルチテナンシーとリソース分離

AWS セキュリティエージェントはマルチテナントサービスです。セキュリティレビュー、検出結果、顧客データは、個々の AWS アカウントに分離され、保管時に暗号化されます。AWS は、ある顧客のセキュリティテストアクティビティが別の顧客のパフォーマンスや機密性に影響を与えないように、標準のインフラストラクチャ分離コントロールを適用します。

AWS セキュリティエージェントとインターフェイス VPC エンドポイント (AWS PrivateLink)

AWS PrivateLink を使用して、VPC と AWS セキュリティエージェントの間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのようにセキュリティエージェントにアクセスできます。VPC 内のインスタンスは、セキュリティエージェントにアクセスするためにパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、 AWS PrivateLink を搭載したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、セキュリティエージェント宛てのトラフィックのエントリポイントとして機能するリクエスタ管理のネットワークインターフェイスです。

詳細については、「 AWS PrivateLink ガイド」の「PrivateLink 経由で AWS サービスにアクセスする」を参照してください。 AWS PrivateLink

セキュリティエージェントの考慮事項

セキュリティエージェントのインターフェイスエンドポイントを設定する前に、 AWS 「 PrivateLink ガイド」の「考慮事項」を参照してください。

セキュリティエージェントは、エージェントスペースの管理オペレーション、侵入テスト、セキュリティ検出結果など、VPC からのすべての API アクションの呼び出しをサポートします。

エンドポイントの作成時に、IPv4、IPv6、またはデュアルスタックを選択できます。

VPC エンドポイントポリシーは、 セキュリティエージェントでサポートされています。デフォルトでは、Security Agent へのフルアクセスはインターフェイスエンドポイントを介して許可されます。エンドポイントポリシーをインターフェイスエンドポイントにアタッチするか、セキュリティグループをエンドポイントネットワークインターフェイスに関連付けることで、アクセスを制御できます。

セキュリティエージェントへのすべての API コールは、VPC エンドポイントを介したコールを含め、TLS 1.2 以降を使用して暗号化されます。データ保護の詳細については、「the section called

“データ保護”」を参照してください。セキュリティエージェントの API コールは AWS CloudTrail に記録されます。詳細については、「the section called “例: AWS セキュリティエージェントのログファイルエントリ”」を参照してください。

Security Agent のインターフェイスエンドポイントを作成する

セキュリティエージェントのインターフェイスエンドポイントは、Amazon VPC コンソールまたは コマンドラインインターフェイス (AWS CLI) AWS を使用して作成できます。詳細については、AWS 「 PrivateLink ガイド」の「インターフェイスエンドポイントの作成」を参照してください。

次のサービス名を使用して、Security Agent のインターフェイスエンドポイントを作成します。

```
com.amazonaws.<region>.securityagent
```

AWS CLI を使用してインターフェイスエンドポイントを作成するには、次のコマンドを実行します。リージョン、VPC ID、サブネット IDs、セキュリティグループ ID を独自の値に置き換えます。

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.<region>.securityagent \
    --subnet-ids subnet-1a2b3c4d subnet-5e6f7a8b \
    --security-group-ids sg-1a2b3c4d \
    --private-dns-enabled
```

> **Important**
>
> インターフェイスエンドポイントに対してプライベート DNS を有効にする必要があります。セキュリティエージェントでは、デフォルトのリージョン DNS 名 (などsecurityagent.us-east-1.api.aws) を使用して、エンドポイント経由で API リクエストを行う必要があります。エンドポイント固有の VPCE URL を直接呼び出すことはサポートされていません。

プライベート DNS を使用するには、VPC trueの 属性enableDnsSupportと enableDnsHostnames 属性の両方を に設定する必要があります。詳細については、「Amazon VPC ユーザーガイド」の「VPC の DNS 属性」を参照してください。

インターフェイスエンドポイントのセキュリティグループを設定する

インターフェイスエンドポイントを作成するときに、セキュリティグループをエンドポイントネットワークインターフェイスに関連付けて、Security Agent へのトラフィックを制御できます。セキュリティエージェントと通信する必要がある VPC 内のリソースからのインバウンド HTTPS トラフィック (ポート 443) を許可するセキュリティグループを作成します。

セキュリティグループには、次のインバウンドルールを含める必要があります。

- タイプ: HTTPS
- [Protocol]: TCP
- ポート範囲: 443
- ソース: VPC CIDR ブロック (など`10.0.0.0/16`)、または Security Agent へのアクセスを必要とするリソースのセキュリティグループ IDs

詳細については、 AWS 「 PrivateLink ガイド」の「セキュリティグループ」を参照してください。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイントを介したセキュリティエージェントへのフルアクセスが許可されます。VPC から Security Agent に許可されるアクセスを制御するには、カスタムエンドポイントポリシーをインターフェイスエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール) 。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、 AWS 「 PrivateLink ガイド」の「エンドポイントポリシーを使用してサービスへのアクセスを制御する」を参照してください。

例: AWS セキュリティエージェントアクションの VPC エンドポイントポリシー

以下は、 AWS セキュリティエージェントのエンドポイントポリシーの例です。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リストされている AWS セキュリティエージェントアクションへのアクセスを許可します。

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": [
            "securityagent:CreatePentest",
            "securityagent:ListPentests",
            "securityagent:BatchGetPentests",
            "securityagent:StartPentestExecution",
            "securityagent:StopPentestExecution",
            "securityagent:ListFindings",
            "securityagent:BatchGetFindings"
         ],
         "Resource": "*"
      }
   ]
}
```

例: 指定された AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、エンドポイントを使用したリソースへの123456789012すべてのアクセスを AWS アカウントで拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "securityagent:*",
         "Resource": "*"
      },
      {
         "Effect": "Deny",
         "Principal": {
            "AWS": "123456789012"
         },
```

```
            "Action": "securityagent:*",
            "Resource": "*"
        }
    ]
}
```

トラブルシューティング

Security Agent API を呼び出すと接続がタイムアウトする

- VPC エンドポイントのステータスが available であることを確認します。

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-1a2b3c4d \
    --query "VpcEndpoints[].State"
```

- エンドポイントに関連付けられたセキュリティグループが、VPC CIDR またはソースセキュリティグループからのポート 443 でのインバウンド TCP トラフィックを許可していることを確認します。
- VPC ルートテーブルが、エンドポイントではなくインターネットゲートウェイまたは NAT ゲートウェイにセキュリティエージェントのトラフィックをルーティングしていないことを確認します。

の DNS 解決が失敗する **securityagent.<region>.api.aws**

- エンドポイントでプライベート DNS が有効になっていることを確認します。

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-1a2b3c4d \
    --query "VpcEndpoints[].PrivateDnsEnabled"
```

- VPC で enableDnsSupport と の両方enableDnsHostnames が に設定されていることを確認します。

```
aws ec2 describe-vpc-attribute --vpc-id vpc-1a2b3c4d --attribute enableDnsSupport
aws ec2 describe-vpc-attribute --vpc-id vpc-1a2b3c4d --attribute enableDnsHostnames
```

Access denied Security Agent APIsエラー

- VPC エンドポイントポリシーをチェックして、実行しようとしているアクションが許可されていることを確認します。
- IAM ID ポリシーが必要なsecurityagent:アクセス許可を付与していることを確認します。

- カスタムエンドポイントポリシーを使用する場合は、エンドポイントポリシーと IAM ポリシーの両方がリクエストを許可していることを確認します。

AWS セキュリティエージェントの設定と脆弱性の分析

設定および IT 管理は、AWS とお客様の間で共有される責任です。詳細については、AWS 責任共有モデルを参照してください。

サービス間の混乱した代理の防止

混乱した代理問題とは、あるアクションを実行する許可を持たないエンティティが、より多くの特権を持つエンティティにアクションの実行を強制できることで生じるセキュリティ上の問題です。AWS では、サービス間でのなりすましが、混乱した代理問題の原因となる可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自身のアクセス許可が使用されて、本来アクセス許可が付与されるべきではない方法により、別の顧客のリソースに影響を与える可能性があります。これを防ぐために、AWS では、アカウントのリソースへのアクセス許可が付与されたサービスプリンシパルを持つすべてのサービスのデータ保護に役立つツールを提供しています。詳細については、AWS Identity and Access Management ユーザーガイドの「サービス間の混乱した代理の防止」を参照してください。

AWS セキュリティエージェントのセキュリティのベストプラクティス

AWS セキュリティエージェントには、独自のセキュリティポリシーを開発および実装する際に考慮すべきセキュリティ機能が多数用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお客様の環境に適切ではないか、十分ではない場合があるため、これらは指示ではなく、有用な考慮事項と見なしてください。

ペネトレーションテストに非本番環境を使用する

AWS セキュリティエージェントは、Kali Linux ディストリビューションの包括的なペネトテストツールスイートを使用します。これらのツールは、セキュリティの脆弱性を特定するように設計されており、アプリケーションの状態、データ、またはシステム設定を変更するアクションを実行する場合があります。

ベストプラクティス: 本番稼働用セットアップを反映する非本番稼働用環境に対してペネトレーションテストを実施します。これらのテスト環境は次の条件を満たす必要があります。

- ライブの顧客データや機密性の高い本番稼働情報が含まれていない
- 本稼働システムから分離する
- 本番環境と同様の設定とセキュリティコントロールがある
- 本番稼働用システムへのアクセスに認証情報を使用しない

本番環境でテストすると、次の結果になる可能性があります。

- データの変更または削除
- サービスの中断またはパフォーマンスの低下
- 意図しない状態の変更
- セキュリティアラートまたはインシデント対応手順のトリガー

AI が生成したセキュリティ検出結果を検証する

AWS セキュリティエージェントは、AI エージェントを使用してセキュリティ分析を実行します。AI システムは非決定的であるため、ペネトレーションテストの実行では、実行ごとに結果が異なる場合があります。

ベストプラクティス: 修復アクションを実行する前に、セキュリティの検出結果を検証します。

- 検出結果ごとに AWS Security Agent によって生成された検証スクリプトを確認する
- テスト環境で検証スクリプトを実行して脆弱性を確認する
- 包括的なカバレッジを確保するために、複数のペネトレーションテストの実行を検討する
- 専門的なセキュリティ判断を適用して、検出結果の重要度と悪用可能性を評価する

特定されたすべての問題が、特定のデプロイコンテキストで悪用可能な脆弱性を示しているわけではありません。

生成された修復コードを確認してテストする

AWS セキュリティエージェントは、特定された脆弱性に対するコード修正とセキュリティの改善を生成できます。これらの AI 生成の修正には、デプロイ前に検証が必要です。

ベストプラクティス: 生成されたすべてのコード変更を確認します。

- 提案された修正の完全性と正確性を検証する

- 非本番環境で修正を徹底的にテストする
- 修正によって新しい脆弱性や機能破損が発生しないことを確認する
- AWS セキュリティエージェントを使用して修正を適用した後に再テストするか、提供された検証スクリプトを実行します。
- 組織のコードレビューと承認プロセスに従う

コードリポジトリへのアクセス

AWS セキュリティエージェントは、プルリクエストコメントとコードレビュー統合を通じて、コード変更に関するセキュリティガイダンスを提供できます。機密性の高いセキュリティ情報を保護するために、この機能は特定の制約の下で動作します。

制限: コードセキュリティガイダンスはプライベートリポジトリのみに制限されています。これにより、以下が保証されます。

- セキュリティの検出結果は組織内で機密に保たれます。
- 潜在的な脆弱性は修復前に公開されない
- 生成された修正レコメンデーションは、悪用の詳細を公開しません

AWS セキュリティエージェントは、パブリックリポジトリのコードセキュリティガイダンスを提供しません。セキュリティ検出結果が公開されるパブリックリポジトリやオープンソースプロジェクトにはコメントしません。

AWS セキュリティエージェントのペネトレーションテストでは、ペンテスト用に設定したプライベートリポジトリとパブリックリポジトリを確認して修正できます。リポジトリがパブリックの場合、修復コードはプルリクエストの代わりにダウンロード可能な差分ファイルとして提供されます。

アクセス可能な URLs

アクセス可能な URLsペネトレーションテスト環境がテスト中にアクセスできる追加のエンドポイントを指定します。これらは、アプリケーションがサードパーティーの認証プロバイダーや CDNs などの外部サービスに依存する場合に必要です。テストに必要なすべてのネットワーク依存関係は、ターゲット URLsまたはアクセス可能な URLsとして指定する必要があります。ネットワークは、指定されていないエンドポイントへのアクセスをブロックします。

セキュリティへの影響: AWS セキュリティエージェントは、アクセス可能な URLs でセキュリティテストを実行するように指示されていません。アクセス可能な URLsを指定することで、これらの依

存関係に対する信頼を示します。認証情報を含む侵入テストデータは、テスト中にこれらのアクセス可能な URL エンドポイントに送信される場合があります。

クロスリージョン推論

AWS セキュリティエージェントは、推論リクエストを処理する最適なリージョンを自動的に選択します。これにより、利用可能なコンピューティングリソース、モデルの可用性を最大化し、最高のカスタマーエクスペリエンスを実現します。データはリクエストが発生したリージョンにのみ保存されますが、入力プロンプトと出力結果はそのリージョン外で処理される場合があります。AWS ネットワーク全体で暗号化されたすべてのデータを送信します。

AWS セキュリティエージェントは、リージョンに応じて 2 種類のクロスリージョン推論を使用します。

- 地理的クロスリージョン推論 – ほとんどの機能について、データ処理を特定の地理的境界 (米国、欧州、オーストラリア、日本など) 内に保持します。コード修復の場合、オーストラリアと日本からのリクエストは欧州連合で処理されます。米国東部 (バージニア北部) – us-east-1、米国西部 (オレゴン) – us-west-2、アジアパシフィック (シドニー) – ap-southeast-2、アジアパシフィック (東京) – ap-northeast-1、欧州 (フランクフルト) – eu-central-1、欧州 (アイルランド) – で使用されますeu-west-1。
- グローバルクロスリージョン推論 – 推論リクエストを商用 AWS リージョンにルーティングし、利用可能なリソースを最適化して、モデルスループットを向上させます。アジアパシフィック (ムンバイ) – ap-south-1、アジアパシフィック (シンガポール) – ap-southeast-1、南米 (サンパウロ) – で使用されますsa-east-1。

グローバルクロスリージョン推論を使用するリージョンでは、入力プロンプトと出力結果が任意の商用 AWS リージョンで処理される場合があります。クロスリージョンオペレーション中に送信されるすべてのデータは AWS ネットワークに残り、パブリックインターネットを経由しません。AWS リージョン間で転送中のデータは暗号化されます。

次の表は、リクエストが発生したリージョンと使用された機能に基づいて、推論リクエストが処理される場所を示しています。

リクエストオリジン	コード修復を除くすべての機能	コード修復
米国 — 米国東部 (バージニア北部) – us-east-1 、米国西部 (オレゴン) – us-west-2	アメリカ	アメリカ
欧州連合 — 欧州 (アイルランド) – eu-west-1 、欧州 (フランクフルト) – eu-central-1	欧州連合	欧州連合
オーストラリア — アジアパシフィック (シドニー) – ap-southeast-2	オーストラリア	欧州連合
日本 — アジアパシフィック (東京) – ap-northeast-1	日本	欧州連合
南米 — 南米 (サンパウロ) – sa-east-1	商用 AWS リージョン	商用 AWS リージョン
インド — アジアパシフィック (ムンバイ) – ap-south-1	商用 AWS リージョン	商用 AWS リージョン
東南アジア — アジアパシフィック (シンガポール) – ap-southeast-1	商用 AWS リージョン	商用 AWS リージョン

クロスリージョン推論は常に有効になっており、オプトアウトすることはできません。クロスリージョン推論は、顧客コンテンツを特定のリージョンに制限するサービスコントロールポリシー (SCPsまたは AWS Control Tower の顧客ポリシーの影響を受けません。AWS セキュリティエージェントがクロスリージョン処理中にデータを保護する方法の詳細については、「クロスリージョンデータ処理」を参照してください。

IAM アクションとリソースの移行

AWS セキュリティエージェントは、すべての環境の開発ライフサイクルを通じてアプリケーションをプロアクティブに保護するフロンティアエージェントです。2026 年 2 月 9 日より前に AWS セキュリティエージェントにオンボーディングした場合、2026 年 3 月 9 日の既存のエージェントインスタンスリソースと AWS セキュリティエージェントの IAM アクションに対する今後の変更の影響を受けます。パブリック API/SDK サポートをリリースする準備として、エージェントインスタンスリソースの名前がエージェントスペースに変更され、特定の IAM アクションの名前が変更されています。これらの変更は、2026 年 3 月 9 日より前に作成したアプリケーションまたはエージェントインスタンス IAM ロールに影響します。2026 年 3 月 9 日以降に認証の問題が発生しないようにするには、「移行の準備」のステップに従う必要があります。

> **Note**
>
> 2026 年 2 月 9 日以降に新しいエージェントインスタンスを作成すると、新しいエージェントインスタンスはエージェントスペースとして作成され、移行ステップは必要ありません。

計画された変更

AWS セキュリティエージェントは、エージェントインスタンスリソースの名前をエージェントスペース`arn:aws:securityagent:us-east-1:{{accountId}}:agent-instance/*`に変更しています。名前は に変更されました`arn:aws:securityagent:us-east-1:{{accountId}}:agent-space/*`。さらに、次の IAM アクションの名前が変更されています。

- `securityagent:ListAgentInstances` の名前を `securityagent:ListAgentSpaces` に変更
- `securityagent:ListControls` の名前を `securityagent:ListSecurityRequirements` に変更
- `securityagent:BatchGetAgentInstances` の名前を `securityagent:BatchGetAgentSpaces` に変更
- `securityagent:BatchGetSecurityTestContentMetadata` の名前を `securityagent:BatchGetPentestJobContentMetadata` に変更
- `securityagent:BatchGetTasks` の名前を `securityagent:BatchGetPentestJobTasks` に変更
- `securityagent:CreateDocumentReview` の名前を `securityagent:CreateDesignReview` に変更

- securityagent:GetDocumentReview の名前を securityagent:GetDesignReview に変更
- securityagent:GetDocumentReviewArtifact の名前を securityagent:GetDesignReviewArtifact に変更
- securityagent:ListDocumentReviews の名前を securityagent:ListDesignReviews に変更
- securityagent:ListDocumentReviewComments の名前を securityagent:ListDesignReviewComments に変更
- securityagent:ListTasks の名前を securityagent:ListPentestJobTasks に変更
- securityagent:StartPentestExecution の名前を securityagent:StartPentestJob に変更
- securityagent:StopPentestExecution の名前を securityagent:StopPentestJob に変更
- securityagent:DeleteDocumentReview の名前を securityagent:DeleteDesignReview に変更

移行の準備

2026 年 3 月 9 日より前に AWS セキュリティエージェントを使用し続けながら 2026 年 3 月 9 日以降に問題が発生しないようにするには、2026 年 3 月 9 日まで IAM ロール/ポリシーの新規および古いリソースと IAM アクションの両方を信頼する必要があります。次の手順では、新しいリソースおよびアクション形式に移行するためのガイドを提供します。

1. AWS アカウントにログインし、AWS セキュリティエージェントコンソールに移動します。
2. 左側のパネルで、設定を選択し、サービスロールでロールを選択します。
3. 関連付けられたロールの IAM コンソールで、アクセス許可の追加とポリシーのアタッチを選択します。
4. AWSSecurityAgentWebAppPolicy を選択し、アクセス許可の追加を選択します。
   a. 重要な注意: AWSSecurityAgentWebAppPolicy SecurityAgentWebAppAPIPolicyを新しいポリシーとして選択したことを確認します。
5. アクセス許可ポリシーで IAM ロールに AWSSecurityAgentWebAppPolicy と SecurityAgentWebAppAPIPolicy の両方があることを確認します。
6. 同じ IAM ロールコンソールで、信頼関係を選択し、信頼ポリシーを編集します
7. 信頼ポリシーを次の形式に更新し、{{accountId}} を AWS アカウント ID に置き換えます。

```
{
  "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. AWS セキュリティエージェントコンソールに戻ります。左側のパネルから、エージェントスペースを選択します。
9. ペネトレーションテストを有効にしたエージェントスペースごとに、次の手順を実行します。
   a. エージェントスペースに移動し、侵入テストを選択する
   b. サービスアクセスで、サービスロール名でロールを選択します。
   c. 関連付けられたロールの IAM コンソールで、信頼関係を選択し、信頼ポリシーを編集します。
   d. 信頼ポリシーを次の形式に更新し、{{accountId}} を AWS アカウント ID に置き換えます。

```
{
  "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/*"
          ]
        }
      }
    }
  ]
}
```

AWS CloudTrail を使用した AWS Security Agent API コールのログ記録

AWS Security Agent は、 AWS Security Agent のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである AWS CloudTrail と統合されています。CloudTrail は、 AWS セキュリティエージェントのすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、 AWS セキュリティエージェントコンソールからの呼び出しと AWS 、セキュリティエージェント API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、 AWS セキュリティエージェントのイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールのイベント履歴で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、 AWS セキュリティエージェントに対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail に関する詳細は、「AWS CloudTrail ユーザーガイド」を参照してください。

AWS CloudTrail のセキュリティエージェント情報

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。 AWS Security Agent でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

AWS Security Agent のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての AWS セキュリティエージェントのアクションは CloudTrail によって ログに記録されます。たとえば、CreatePentest、BatchGetPentestJobs、ListFindings の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

例: AWS セキュリティエージェントのログファイルエントリ

AWS Security Agent ログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、公開 API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、CreatePentest アクションを示す CloudTrail ログエントリの例です。

```
{
    "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"
}
```


サービスクォータ

AWS Security Agent には、作成できるリソースの数とオペレーションの実行レートを制限するクォータがあります。調整可能とマークされたクォータは、AWS サポートを通じてリクエストを送信することで増やすことができます。詳細については、サポートケースとケース管理の作成を参照してください。

オペレーションクォータ

運用クォータは、サービス容量の管理に役立つように、セキュリティテストとレビュー機能の毎月の使用を制限します。

[リソース]	スコープ	クォータ	引き上げ可能
設計レビュー	1 か月、1 アカウント、1 リージョンあたり	200	はい

[リソース]	スコープ	クォータ	引き上げ可能
PR コードのレビュー	1 か月、1 アカウント、1 リージョンあたり	1,000	はい

設定クォータ

設定クォータは、AWS Security Agent 環境で設定できるリソースと設定の数を制限します。

[リソース]	スコープ	クォータ	引き上げ可能
エージェントスペース	アカウントごと、リージョンごと	100	はい
統合	アカウントごと、リージョンごと	20	いいえ
統合あたりの統合リソース	統合ごと	50	いいえ
カスタムセキュリティ要件	アカウントごと、リージョンごと	20	いいえ
ペンテストプロジェクト	アカウントごと、リージョンごと	1,000	はい
ペンテストの同時実行	アカウントごと、リージョンごと	5	はい
コードレビュープロジェクト	アカウントごと、リージョンごと	1,000	はい
コードレビューの同時実行	アカウントごと、リージョンごと	5	はい

ドキュメント履歴

次の表は、AWS セキュリティエージェントユーザーガイドの主な更新と新機能の一部を示しています。

変更	説明	日付
プライベート接続 (プレビュー)	プライベート接続に関するドキュメントを追加しました。この新機能により、AWS セキュリティエージェントは、システムをパブリックインターネットに公開することなく、Amazon VPC Lattice を使用してプライベートネットワークで実行されているソース管理システムに接続できます。	2026 年 6 月 16 日
脅威モデリング (プレビュー)	脅威モデリングに関するドキュメントを追加しました。この新しい機能は、設計ドキュメント (スコープドキュメント) 、ソースコード (ソース) 、またはその両方からアプリケーションの脅威モデルを構築します。スコープドキュメントは、分析の焦点を定義する機能設計ドキュメントであり、ソースコードは既存のシステムに関するコンテキストを提供します。スコープドキュメントを指定しない場合、エージェントはソースコードのみから脅威モデルを生成します。各実行では、システム概要と、重要度評価と	2026 年 6 月 8 日

	推奨事項を含む STRIDE カテゴリ別に分類された一連の脅威が生成されます。	
完全なリポジトリコードレビュー (プレビュー)	完全なリポジトリコードレビューに関するドキュメントを追加しました。この新機能は、コードベース全体のコンテキスト対応セキュリティ分析を実行し、検出結果のコード修復を生成します。	2026 年 5 月 12 日
検出結果の修復のためにリージョンの可用性を更新	AWS セキュリティエージェントウェブアプリでのペネトレーションテストの検出結果修復のリージョンの可用性が更新されました。	2026 年 4 月 16 日
検出結果の修正を有効にするためにリージョンの可用性を更新	AWS マネジメントコンソールで侵入テストの検出結果の修正を有効にするためのリージョンの可用性が更新されました。	2026 年 4 月 16 日
カスタマーマネージドキーのサポート	カスタマーマネージドキー (CMK) サポートに関するドキュメントを追加しました。エージェントスペースと統合を作成するときに、カスタマーマネージド KMS キーを指定して、制御するキーでデータを暗号化できるようになりました。	2026 年 3 月 31 日

AWS マネージドポリシーの更新	新しい TargetDomain リソースタイプと DesignReviewFeedback リソースタイプの AWSSecurityAgentWebAppPolicy 管理ポリシーを追加しました。	2026 年 3 月 31 日
AWS マネージドポリシーの更新	新しい AgentSpace リソースタイプと IAM アクション名の変更に AWSSecurityAgentWebAppPolicy 管理ポリシーを追加しました。	2026 年 2 月 9 日
AWS マネージドポリシーの更新	SecurityAgentWebAppAPIPolicy を更新して、お客様が設計レビューを削除できるようにしました。	2026 年 1 月 28 日
AWS マネージドポリシーの更新	SecurityAgentWebAppAPIPolicy を更新して、お客様がセキュリティ検出結果の自動コード修復を開始できるようにしました。	2026 年 1 月 20 日
AWS マネージドポリシーの更新	SecurityAgentWebAppAPIPolicy に更新され、お客様が コンソールでイメージを表示できるようになりました。	2025 年 12 月 5 日
AWS セキュリティエージェントの初回リリース	サービス開始時の初版ドキュメント	2025 年 12 月 2 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。