ユーザーガイド

Amazon Inspector

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Inspector: ユーザーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

Amazon Inspector とは 1
- 機能 1
- Amazon Inspector へのアクセス 3

入門 5
- Amazon Inspector をアクティブ化する前に 5
- 入門チュートリアル: Amazon Inspector のアクティブ化 6

自動スキャン 8
- Amazon Inspector のスキャンタイプの概要 8
- スキャンタイプをアクティブ化する 9
  - スキャンのアクティブ化 11
- Amazon EC2 インスタンススキャン 12
  - エージェントベースのスキャン 12
  - エージェントレススキャン 17
  - スキャンモードの管理 19
  - Amazon Inspector スキャンからのインスタンスの除外 20
  - サポートされるオペレーティングシステム 20
  - Linux インスタンス向け詳細検査 21
  - Windows EC2 インスタンスのスキャン 25
- Amazon ECR コンテナイメージスキャン 28
  - Amazon ECR スキャンのスキャン動作 29
  - 実行中のコンテナへのコンテナイメージのマッピング 30
  - サポートされているオペレーティングシステムとメディアタイプ 31
  - Amazon ECR 再スキャン期間の設定 32
- Lambda 関数のスキャン 34
  - Lambda 関数スキャンのスキャン動作 35
  - サポートされているランタイムと関数 35
  - Amazon Inspector Lambda 標準スキャン 36
  - Amazon Inspector Lambda コードスキャン 37
- スキャンタイプの非アクティブ化 39
  - スキャンの非アクティブ化 40

CIS スキャン 42
- Amazon Inspector CIS スキャンに関する Amazon EC2 インスタンスの要件 43
  - プライベート Amazon EC2 インスタンスで CIS スキャンを実行するための Amazon Virtual Private Cloud エンドポイントの要件 44

CIS スキャンの実行 44
を使用した Amazon Inspector CIS スキャンの管理に関する考慮事項 AWS Organizations 45
Amazon Inspector CIS スキャンに使用される Amazon Inspector 所有の Amazon S3 バケット 46
CIS スキャン設定の作成 48
CIS スキャン結果の表示 49
CIS スキャン設定の編集 50
CIS スキャン結果のダウンロード 51
Amazon Inspector コードセキュリティ 53
前提条件 53
コードセキュリティのアクティブ化 53
にアクセスするためのカスタマーマネージドキーの作成 AWS KMS 53
統合の作成 56
の統合の作成 GitHub 57
の統合の作成 GitLab Self Managed 58
統合の表示 60
コードリポジトリの表示 61
統合の削除 62
スキャン設定の作成 63
スキャン設定の表示 65
スキャン設定の編集 66
スキャン設定の削除 67
オンデマンドスキャンの実行 67
サポートされている言語 68
コードセキュリティの無効化 69
検出結果について 70
検出結果タイプ 71
パッケージ脆弱性 71
コードの脆弱性 71
ネットワーク到達可能性 72
結果の表示 73
結果の詳細の表示 74
Amazon Inspector スコアの表示 78
Amazon Inspector スコア 78
脆弱性インテリジェンス 80
検出結果の重要度レベルについて 81

ソフトウェアパッケージの脆弱性の重要度 81
コード脆弱性の重要度 82
ネットワーク到達可能性の重要度 81
検出結果の管理 85
検出結果のフィルタリング 85
Amazon Inspector コンソールでフィルターを作成 85
調査結果を抑制する 86
抑制ルールを作成する 87
抑制された検出結果を表示する 87
抑制ルールの編集 88
抑制ルールを削除する 88
調査結果レポートのエクスポート 89
ステップ 1: アクセス許可を確認する 90
ステップ 2: S3 バケットを設定する 92
ステップ 3: AWS KMS keyを設定する 95
ステップ 4: 調査結果レポートを設定しエクスポートする 98
エラーのトラブルシューティング 101
EventBridge を使用して検出結果に対するレスポンスを自動化 102
イベントスキーマ 102
Amazon Inspector の検出結果を通知する EventBridge ルールの作成 105
Amazon Inspector マルチアカウント環境の EventBridge 109
ダッシュボード 110
ダッシュボードの表示 110
ダッシュボードのコンポーネントを理解する 111
脆弱性データベースの検索 114
脆弱性データベースの検索 114
CVE の詳細について 115
CVE の詳細 115
脆弱性インテリジェンス 115
リファレンス 115
SBOM のエクスポート 116
Amazon Inspector 形式 116
SBOM 用フィルター 121
SBOM の設定とエクスポート 122
EventBridge スキーマ 125
Amazon Inspector 用の Amazon EventBridge ベーススキーマ 125

Amazon Inspector 検出結果イベントスキーマの例 126
Amazon Inspector の初回スキャン完了イベントスキーマの例 138
Amazon Inspector カバレッジイベントスキーマの例 141
Amazon Inspector 自動有効化スキーマの例 142
SSM プラグイン 143
の Amazon Inspector SSM プラグイン Linux 143
Amazon Inspector SSM プラグインのアンインストール 143
の Amazon Inspector SSM プラグイン Windows 144
Amazon Inspector SSM プラグインのアンインストール 144
Amazon Inspector SBOM Generator 146
サポートされているパッケージタイプ 146
サポートされているコンテナイメージ設定チェック 146
Sbomgen のインストール 147
Sbomgen を使用する 148
コンテナイメージの SBOM の生成と結果の出力 148
ディレクトリとアーカイブからの SBOM の生成 150
Go または Rust のコンパイル済みバイナリからの SBOM の生成 150
マウントされたボリュームから SBOM を生成する 150
脆弱性特定のための SBOM の Amazon Inspector への送信 151
追加のスキャナーを使用して検出機能を強化する 153
スキャンする最大ファイルサイズを調整してコンテナスキャンを最適化する 154
進行状況インジケータの無効化 155
Sbomgen を使用したプライベートレジストリへの認証 155
キャッシュされた認証情報を使用した認証 (推奨) 155
インタラクティブメソッドを使用した認証 156
非インタラクティブなメソッドを使用した認証 156
Sbomgen からの出力例 157
以前のバージョン 159
オペレーティングシステムの収集 164
サポートされているオペレーティングシステムアーティファクト 165
APK ベースの OS パッケージコレクション 166
DPKG ベースの OS パッケージコレクション 167
RPM ベースの OS パッケージコレクション 168
Chainguard イメージパッケージコレクション 169
Distroless イメージパッケージコレクション 170
MinimOS パッケージコレクション 171

依存関係の収集 ... 172
Go 依存関係スキャン ... 173
Java 依存関係スキャン ... 176
JavaScript 依存関係スキャン ... 180
.NET 依存関係スキャン ... 187
PHP 依存関係スキャン ... 192
Python 依存関係スキャン ... 195
Ruby 依存関係スキャン ... 200
Rust 依存関係スキャン ... 203
サポートされていないアーティファクト ... 206
エコシステムコレクション ... 208
サポートされているエコシステム ... 208
Apache エコシステムコレクション ... 209
Java エコシステムコレクション ... 211
Google エコシステムコレクション ... 213
WordPress エコシステムコレクション ... 214
Node.JS ランタイムコレクション ... 217
OpenSSL エコシステムコレクション ... 218
ライセンスコレクション ... 219
ライセンス情報を収集する ... 219
サポートされているパッケージ ... 220
パッケージ URLs ... 226
PURL 構造 ... 226
バージョンリファレンス ... 228
レコメンデーション ... 229
Java ... 229
JavaScript ... 229
Python ... 230
CycloneDX 名前空間の使用 ... 230
`amazon:inspector:sbom_scanner` 名前空間の分類 ... 230
`amazon:inspector:sbom_generator` 名前空間の分類 ... 232
CI/CD の統合 ... 235
プラグインによる統合 ... 235
サポートされている CI/CD ソリューション ... 236
カスタム統合 ... 237
CI/CD 統合用のアカウントをセットアップする ... 237

にサインアップする AWS アカウント 238
管理アクセスを持つユーザーを作成する 238
CI/CD への統合のための IAM ロールを設定する 240
Amazon Inspector Dockerfile チェック 241
Sbomgen Dockerfile チェックの使用 241
サポートされている Dockerfile チェック 244
カスタム CI/CD 統合の作成 249
ステップ 1. の設定 AWS アカウント 250
ステップ 2. Sbomgen バイナリのインストール 250
ステップ 3. Sbomgen を使用する 250
ステップ 4. Amazon Inspector スキャン API の呼び出し 250
(オプション) ステップ 5. 1 つのコマンドでの SBOM の生成とスキャン 251
API 出力形式 251
Jenkins プラグイン 259
ステップ 1. のセットアップ AWS アカウント 260
ステップ 2. Amazon Inspector Jenkins プラグインのインストール 260
(オプション) ステップ 3. Jenkins への Docker 認証情報の追加 260
(オプション) ステップ 4. AWS 認証情報を追加する 261
ステップ 5. Jenkins スクリプトへの CSS サポートの追加 261
ステップ 6。Amazon Inspector スキャンのビルドへの追加 261
ステップ 7。Amazon Inspector の脆弱性レポートの確認 265
トラブルシューティング 266
TeamCity プラグイン 268
GitHub アクション 270
GitLab コンポーネント 270
CodeCatalyst アクションの使用 271
Amazon Inspector スキャンアクションの使用 271
カバレッジの評価 272
アカウントレベルのカバレッジを評価する 273
Amazon EC2 インスタンスのカバレッジを評価する 273
Amazon EC2 インスタンスのステータス値 274
Amazon ECR リポジトリのカバレッジを評価する 276
Amazon ECR リポジトリスキャンのステータス値 277
Amazon ECR コンテナイメージのカバレッジを評価する 278
Amazon ECR コンテナイメージのスキャンステータスの値 279
AWS Lambda 関数のカバレッジを評価する 280

Lambda 関数スキャンのステータス値 ... 281
複数のアカウントの管理 ... 282
委任管理者アカウントとメンバーアカウントについて ... 282
委任管理者のアクション ... 282
メンバーアカウントのアクション ... 283
管理者アカウントの指定 ... 284
考慮事項 ... 284
委任された管理者の指定に必要な許可 ... 285
委任された管理者を指定する ... 286
メンバーアカウントの Amazon Inspector スキャンをアクティブ化する ... 287
メンバーアカウントの関連付けを解除する ... 291
委任管理者の削除 ... 292
リソースのタグ付け ... 294
タグ付けの基本 ... 294
タグの追加 ... 295
Amazon Inspector リソースへのタグの追加 ... 295
タグの削除 ... 296
Amazon Inspector リソースからのタグの削除 ... 296
使用方法 ... 298
コンソール使用状況を使用する ... 298
Amazon Inspector での使用コストの計算方法について ... 300
Amazon Inspector の無料トライアルについて ... 300
セキュリティ ... 302
データ保護 ... 303
保管中の暗号化 ... 304
転送中の暗号化 ... 308
Identity and Access Management ... 308
対象者 ... 309
アイデンティティを使用した認証 ... 310
ポリシーを使用したアクセスの管理 ... 313
Amazon Inspector と IAM の連携 ... 316
アイデンティティベースのポリシーの例 ... 323
AWS マネージドポリシー ... 327
サービスにリンクされたロールの使用 ... 343
トラブルシューティング ... 359
Amazon Inspector のモニタリング ... 361

CloudTrail ログ ... 362
コンプライアンス検証 ... 365
耐障害性 ... 366
インフラストラクチャセキュリティ ... 367
インシデントへの対応 ... 367
AWS PrivateLink ... 368
考慮事項 ... 368
インターフェイスエンドポイントの作成 ... 368
統合 ... 370
Amazon Inspector と Amazon ECR の統合 ... 370
Amazon Inspector と Security Hub の統合 ... 370
Amazon ECR の統合 ... 370
統合をアクティブ化する ... 371
マルチアカウント環境との統合を使用する ... 371
セキュリティハブの統合 ... 371
での Amazon Inspector の検出結果の表示 AWS Security Hub ... 372
Amazon Inspector と Security Hub の統合のアクティブ化と設定 ... 376
統合先からの検出結果フローの無効化 ... 376
Security Hub での Amazon Inspector のセキュリティコントロールの表示 ... 376
サポートされているオペレーティングシステムとプログラミング言語 ... 377
サポートされるオペレーティングシステム ... 378
サポートされているオペレーティングシステム: Amazon EC2スキャン ... 378
サポートされているオペレーティングシステム: Amazon Inspector による Amazon ECR スキャン ... 382
サポートされているオペレーティングシステム: CIS スキャン ... 384
終了オペレーティングシステム ... 385
サポートされているプログラミング言語 ... 392
サポートされているプログラミング言語: Amazon EC2 エージェントレススキャン ... 392
サポートされているプログラミング言語: Amazon EC2 詳細検査 ... 393
サポートされているプログラミング言語: Amazon ECR スキャン ... 393
ランタイムのサポート ... 394
サポートされているランタイム: Amazon Inspector Lambda 標準スキャン ... 394
サポートされているランタイム: Amazon Inspector Lambda コードスキャン ... 396
Amazon Inspector の非アクティブ化 ... 397
Amazon Inspector を非アクティブ化する ... 398
クォータ ... 400

リージョンとエンドポイント 402
Amazon Inspector のサービスエンドポイント 402
Amazon Inspector スキャン API のエンドポイント 402
リージョン固有機能の可用性 417
ドキュメント履歴 422
AWS 用語集 445
..... cdxlvi

Amazon Inspector とは

Amazon Inspector は、ソフトウェアの脆弱性や意図しないネットワークの露出についてワークロードを自動的に検出し、継続的にスキャンする脆弱性管理サービスです。Amazon Inspector は、Amazon EC2 インスタンス、Amazon ECR のコンテナイメージ、および Lambda 関数を検出してスキャンします。Amazon Inspector がソフトウェアの脆弱性または意図しないネットワークへの露出を検出すると、問題に関する詳細なレポートである検出結果が作成されます。Amazon Inspector コンソールまたは API で検出結果を管理できます。

トピック

- Amazon Inspector の特徴
- Amazon Inspector へのアクセス

Amazon Inspector の特徴

複数の Amazon Inspector アカウントを一元管理

AWS 環境に複数のアカウントがある場合は、 AWS Organizations を使用して 1 つのアカウントで環境を一元管理できます。この方法を使用することで、Amazon Inspector の委任された管理者アカウントとしてアカウントを指定できます。

Amazon Inspector は、ワンクリックで組織全体にアクティブ化できます。さらに、今後新たなメンバーが組織に入るたびに、自動的にサービスをアクティブ化することもできます。Amazon Inspector の委任された管理者アカウントは、組織のメンバーの検出結果データと特定の設定を管理できます。これには、すべてのメンバーアカウントの集計結果の詳細の表示、メンバーアカウントのスキャンのアクティブ化または非アクティブ化、 AWS 組織内のスキャンされたリソースの確認が含まれます。

環境を継続的にスキャンして、脆弱性やネットワークの露出がないかを確認します。

Amazon Inspector を使用すれば、評価スキャンを手動でスケジュールまたは設定する必要はありません。Amazon Inspector は、対象となるリソースを自動的に検出し、スキャンを開始します。Amazon Inspector は、EC2 インスタンスへの新しいパッケージのインストール、パッチのインストール、リソースに影響を与える新しい共通脆弱性識別子 (CVE) が発行された場合など、新しい脆弱性をもたらす可能性のある変更に対応してリソースを自動的に再スキャンすることで、リソースのライフサイクル全体を通じて引き続き環境を評価します。従来のセキュリティスキャンソフトウェ

アとは異なり、Amazon Inspector はお客様のフリートのパフォーマンスへの影響を最小限に抑えます。

脆弱性またはオープンネットワークパスが特定されると、Amazon Inspector は調査可能な検出結果を生成します。検出結果には、脆弱性、影響を受けるリソース、および修復に関する推奨事項に関する包括的な詳細が含まれます。検出結果を適切に修復すると、Amazon Inspector は自動的に修正を検出し、検出結果を終了します。

Amazon Inspector のリスクスコアを使用して脆弱性を正確に評価

Amazon Inspector はスキャンを通じて環境に関する情報を収集し、その環境に合わせて特別に調整された重要度スコアを提供します。Amazon Inspector は、脆弱性の全国脆弱性データベース (NVD) の基本スコアを構成するセキュリティメトリクスを調べ、コンピューティング環境に応じて調整します。たとえば、脆弱性がネットワーク上で悪用可能であるが、インスタンスからインターネットへのオープンネットワークパスが利用できない場合、サービスは Amazon EC2 インスタンスの検出結果の Amazon Inspector スコアを下げる可能性があります。このスコアは CVSS 形式で、NVD が提供する共通脆弱性スコアリングシステム (CVSS) の基本スコアを修正したものです。

Amazon Inspector ダッシュボードを使用して影響の大きい検出結果を特定する

Amazon Inspector ダッシュボードには、環境全体から得られた検出結果の概要が表示されます。ダッシュボードから、検出結果の詳細にアクセスできます。ダッシュボードには、環境内のスキャン範囲、最も緊急の検出結果、および最も多くの検出結果が得られたリソースに関する効率的な情報が表示されます。Amazon Inspector ダッシュボードのリスクベースの修復パネルには、最も多くのインスタンスとイメージに影響する検出結果が表示されます。このパネルでは、環境に最も大きな影響を与える検出結果の特定、検出結果の詳細確認、および推奨される解決策の確認がより容易になります。

カスタマイズ可能なビューを使用して検出結果を管理

ダッシュボードに加えて、Amazon Inspector コンソールには検出結果ビューがあります。このページでは、環境に関する検出結果をリスト化し、個別の検出結果の詳細を提供します。検出結果は、カテゴリまたは脆弱性タイプ別にグループ化して表示できます。各ビューでは、フィルターを使用して結果をさらにカスタマイズできます。フィルターを使用して、不要な検出結果をビューから隠す非表示ルールを作成することもできます。

フィルターと抑制ルールを使用して、すべての検出結果またはカスタマイズされた結果の選択を表示する結果レポートを生成できます。レポートは CSV 形式または JSON 形式で生成できます。

他のサービスおよびシステムを用いた検出結果のモニタリングと処理

他のサービスやシステムとの統合をサポートするために、Amazon Inspector は検出結果イベントとして Amazon EventBridge に検出結果を発行します。EventBridge は、検出結果を AWS Lambda 関数や Amazon Simple Notification Service (Amazon SNS) トピックなどのターゲットにルーティングできるサーバーレスイベントバスサービスです。EventBridge を使用すると、既存のセキュリティおよびコンプライアンスワークフローの一部として、ほぼリアルタイムで検出結果をモニタリングおよび処理できます。

AWS Security Hub をアクティブ化すると、Amazon Inspector は検出結果を Security Hub にも公開します。Security Hub は、 AWS 環境全体のセキュリティ体制を包括的に把握し、セキュリティ業界標準とベストプラクティスに照らして環境をチェックするのに役立つサービスです。Security Hub を使用すると、 AWS内の組織のセキュリティ体制の広範な分析の一部として、検出結果をより簡単にモニタリングおよび処理できます。

Amazon Inspector へのアクセス

Amazon Inspector はほとんどの で利用できます AWS リージョン。Amazon Inspector が現在利用可能な リージョンの一覧については、Amazon Web Services 全般リファレンス の「Amazon Inspector のエンドポイントとクォータ」を参照してください。 AWS リージョンの詳細については、「Amazon Web Services General Reference」の「Managing AWS リージョン」を参照してください。各リージョンで、次のいずれかの方法で Amazon Inspector を使用できます。

「AWS マネジメントコンソール」

AWS Management Console は、 リソースの作成と管理 AWS に使用できるブラウザベースのインターフェイスです。そのコンソールの一部として、Amazon Inspector コンソールは Amazon Inspector アカウントとリソースへのアクセスを提供します。Amazon Inspector タスクは、Amazon Inspector コンソールから実行できます。

AWS コマンドラインツール

AWS コマンドラインツールを使用すると、システムのコマンドラインでコマンドを発行して Amazon Inspector タスクを実行できます。コマンドラインを使用すると、コンソールを使用するよりも高速で便利になります。コマンドラインツールは、タスクを実行するスクリプトを作成する場合にも便利です。

AWS には、 AWS Command Line Interface （ AWS CLI) と の 2 セットのコマンドラインツールが用意されています AWS Tools for PowerShell。のインストールと使用の詳細については AWS CLI、AWS 「 コマンドラインインターフェイスユーザーガイド」を参照してください。Tools for

PowerShell のインストールおよび使用の方法については、AWS Tools for PowerShell ユーザーガイドを参照してください。

AWS SDK

AWS はSDKs を提供します。SDK は、Amazon Inspector および他の AWS のサービスへの便利なプログラムによるアクセスを提供します。SDK は、暗号署名によるリクエスト、エラーの管理、リクエストの自動再試行などのタスクも処理します。 AWS SDKs」を参照してください。 AWS

Amazon Inspector REST API

Amazon Inspector REST API は、Amazon Inspector アカウントとリソースへの包括的なプログラムによるアクセスを提供します。この API を使用すると、HTTPS リクエストを Amazon Inspector に直接送信できます。ただし、 AWS コマンドラインツールや SDKs とは異なり、この API を使用するには、アプリケーションがリクエストに署名するためのハッシュの生成などの低レベルの詳細を処理する必要があります。

Amazon Inspector の開始方法

このセクションでは、Amazon Inspector をアクティブ化する前に考慮すべき情報と、Amazon Inspector をアクティブ化して Amazon Inspector コンソールと Amazon Inspector API で検出結果を表示する方法について説明する入門チュートリアルを提供します。

トピック

- Amazon Inspector をアクティブ化する前に
- 入門チュートリアル: Amazon Inspector のアクティブ化

Amazon Inspector をアクティブ化する前に

Amazon Inspector をアクティブ化する前に、次の点に注意してください。

Amazon Inspector はリージョナルサービスです

データは、Amazon Inspector をアクティブ化 AWS リージョン する に保存されます。Amazon Inspector を使用する予定のすべての AWS リージョン について、入門チュートリアルの最初の部分の手順を繰り返します。

Amazon Inspector は、サービスにリンクされたロール AWSServiceRoleForAmazonInspector2 と AWSServiceRoleForAmazonInspector2Agentless を作成します。

サービスにリンクされたロールは、 サービスにリンクされた AWS Identity and Access Management (IAM) AWS のロールです。AWSServiceRoleForAmazonInspector2 および AWSServiceRoleForAmazonInspector2Agentless を使用すると、Amazon Inspector はセキュリティ評価の実行 AWS のサービス に必要な にアクセスできます。

管理者アクセス許可を持つ IAM ユーザーアイデンティティは、Amazon Inspector を有効にできます。

IAM または AWS IAM Identity Center でユーザーを作成して、認証情報を保護します。これにより、ユーザーは Amazon Inspector の管理に必要なアクセス許可のみを持つようになります。詳細については、「AWS 管理ポリシー: AmazonInspectorFullAccess」を参照してください。

ハイブリッドスキャンは自動的に有効になります

ハイブリッドスキャンには、エージェントベースのスキャンとエージェントレススキャンが含まれます。デフォルトでは、Amazon Inspector は、対象となるすべての Amazon EC2 インスタンスでこ

れらのスキャン方式を使用します。詳細については、「Amazon Inspector による Amazon EC2 のスキャン」を参照してください。

Amazon ECR スキャンと Lambda 関数スキャンには SSM Agent は必要ありません

エージェントベースのスキャンでは、SSM Agent を使用してソフトウェアインベントリを収集します。エージェントレススキャンでは、Amazon EBS スナップショットを使用してソフトウェアインベントリを収集します。

> **Note**
>
> デフォルトでは、SSM Agent は、Amazon マシンイメージに基づいて Amazon EC2 インスタンスにインストール済みです。ただし、場合によっては SSM Agent を手動でアクティブ化しなければならない場合があります。詳細については、「AWS Systems Manager ユーザーガイド」の「SSM Agent の使用」を参照してください。

毎月のコストは、スキャンされたワークロードに基づいています。

詳細については、「Amazon Inspector の料金」を参照してください。

入門チュートリアル: Amazon Inspector のアクティブ化

このトピックでは、スタンドアロンアカウント環境 (メンバーアカウント) とマルチアカウント環境 (委任管理者アカウント) で Amazon Inspector をアクティブ化する方法について説明します。Amazon Inspector をアクティブ化すると、ワークロードの検出と、ソフトウェア脆弱性や意図しないネットワーク露出のスキャンが自動的に開始されます。

Standalone account environment

次の手順では、メンバーアカウントのコンソールで Amazon Inspector をアクティブ化する方法について説明します。Amazon Inspector をプログラムでアクティブ化するには、inspector2-enablement-with-cli を使用します。

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. [今すぐ始める] を選択します。
3. [Amazon Inspector をアクティブ化] を選択します。

スタンドアロンアカウントで Amazon Inspector をアクティブ化すると、すべてのスキャンタイプがデフォルトでアクティブ化されます。メンバーアカウントの詳細については、Amazon Inspector の委任管理者アカウントとメンバーアカウントについて」を参照してください。

Multi-account environment

次の手順では、委任管理者アカウントのコンソールで Amazon Inspector をアクティブ化する方法について説明します。複数のアカウントに対して Amazon Inspector をプログラムでアクティブ化するには、Amazon Inspector Inspector Inspector2-enablement-with-cli シェルスクリプトを使用します。

> **Note**
>
> この手順を完了するには、 AWS Organizations 管理アカウントを使用する必要があります。 AWS Organizations 管理アカウントのみが委任管理者を指定できます。委任された管理者を指定するには、アクセス許可が必要になる場合があります。詳細については、「委任された管理者の指定に必要な許可」を参照してください。

Amazon Inspector を初めてアクティブ化すると、Amazon Inspector は`AWSServiceRoleForAmazonInspector`アカウントのサービスにリンクされたロールを作成します。Amazon Inspector がサービスにリンクされたロールを使用する方法については、「」を参照してくださいAmazon Inspector でのサービスにリンクされたロールの使用。

Amazon Inspector 用の委任された管理者の指定

1. AWS Organizations 管理アカウントにサインインし、https://console.aws.amazon.com/inspector/v2/home://www.com で Amazon Inspector コンソールを開きます。
2. [開始する] を選択します。
3. 委任管理者 に、委任管理者として AWS アカウント 指定する の 12 桁の ID を入力します。
4. Delegate を選択し、もう一度 Delegate を選択します。
5. （オプション) AWS Organizations 管理アカウントの Amazon Inspector をアクティブ化する場合は、サービスアクセス許可で Amazon Inspector をアクティブ化を選択します。

委任された管理者を指定すると、デフォルトですべてのスキャンタイプがアカウントに対して有効になります。委任管理者アカウントの詳細については、Amazon Inspector の委任管理者アカウントとメンバーアカウントについて」を参照してください。

Amazon Inspector の自動スキャンタイプ

Amazon Inspector は、ソフトウェア脆弱性や意図しないネットワーク露出についてリソースをモニタリングする専用のスキャンエンジンを使用します。Amazon Inspector がソフトウェア脆弱性または意図しないネットワーク露出を検出すると、検出結果が作成されます。Amazon Inspector を初めてアクティブ化すると、アカウントは Amazon EC2 スキャン、Amazon ECR スキャン、Lambda 標準スキャンを含むすべてのスキャンタイプに自動的に登録されます。

> **Note**
>
> Lambda コードスキャンは Lambda 関数スキャンのオプションレイヤーで、いつでもアクティブ化できます。

トピック

- Amazon Inspector のスキャンタイプの概要
- スキャンタイプをアクティブ化する
- Amazon Inspector による Amazon EC2 インスタンスのスキャン
- Amazon Inspector による Amazon Elastic Container Registry コンテナイメージのスキャン
- Amazon Inspector を使用した AWS Lambda 関数のスキャン
- Amazon Inspector でのスキャンタイプの非アクティブ化

Amazon Inspector のスキャンタイプの概要

Amazon Inspector には、 AWS 環境内の特定のリソースタイプに焦点を当てたさまざまなスキャンタイプが用意されています。

Amazon EC2 スキャン

Amazon EC2 スキャンを有効にすると、Amazon Inspector は EC2 インスタンスをスキャンして、一般的な脆弱性と露出 (CVEs)、ネットワーク露出の問題、ネットワーク到達可能性の問題、オペレーティングシステムとプログラミング言語パッケージの脆弱性を検出します。Amazon Inspector は、インスタンスにインストールされている SSM Agent を使用するか、インスタンスの Amazon EBS スナップショットを通じてスキャンを実行します。詳細については、「Amazon Inspector による Amazon EC2 インスタンスのスキャン」を参照してください。デフォルトで

は、Amazon EC2 スキャンをアクティブ化すると、ハイブリッドスキャンモードが自動的に有効になります。詳細については、「エージェントレススキャン」を参照してください。

Amazon ECR スキャン

Amazon ECR スキャンを有効にすると、Amazon Inspector はプライベートレジストリ内のすべてのリポジトリを基本的なスキャンコンテナリポジトリから拡張スキャンリポジトリに変換します。この設定を包含ルールで設定して、オンプッシュのみをスキャンしたり、選択したリポジトリをスキャンしたりできます。Amazon Inspector は、過去 30 日以内にプッシュされた、または過去 90 日以内にプルされたすべてのイメージをスキャンします。Amazon Inspector はデフォルトで 90 日間イメージをモニタリングし続けます。この設定はいつでも変更できます。詳細については、「Amazon Inspector による Amazon Elastic Container Registry コンテナイメージのスキャン」を参照してください。

Lambda 標準スキャン

Lambda 標準スキャンをアクティブ化すると、Amazon Inspector はアカウント内のすべての Lambda 関数を検出し、脆弱性を直ちにスキャンします。Amazon Inspector は、デプロイ時に新しい Lambda 関数とレイヤーをスキャンします。Amazon Inspector は、更新時または新しい CVEsが公開されたときに再スキャンします。スキャンの詳細については、「」を参照してください Amazon Inspector を使用した AWS Lambda 関数のスキャン。

Lambda 標準スキャン + Lambda コードスキャン

Lambda コードスキャンをアクティブ化すると、Amazon Inspector はアカウント内の Lambda 関数とレイヤーを検出し、コードの脆弱性をスキャンします。このタイプのスキャンでは、CVEs の Lambda 関数で使用されるアプリケーションパッケージの依存関係を評価します。このスキャンタイプをアクティブ化すると、Lambda 標準スキャンもアクティブ化されます。詳細については、「Amazon Inspector を使用した AWS Lambda 関数のスキャン」を参照してください。

Amazon Inspector のコードセキュリティ

このスキャンタイプは、Amazon Q Developer スキャンエンジンを活用して、ファーストパーティアプリケーションコード、サードパーティーアプリケーションの依存関係、Infrastructure as Code の脆弱性をスキャンします。詳細については、Amazon Inspector のコードセキュリティ」を参照してください。

スキャンタイプをアクティブ化する

スキャンタイプはいつでもアクティブ化できます。スキャンタイプをアクティブ化すると、Amazon Inspector はスキャンタイプの対象となるリソースのスキャンを開始します。

Amazon EC2 スキャン

このスキャンタイプは、Amazon EC2 インスタンスからメタデータを抽出してから、メタデータをセキュリティアドバイザリから収集されたルールと比較します。このスキャンタイプをアクティブ化すると、Amazon Inspector はアカウント内のすべての適格な Amazon EC2 インスタンスをスキャンして、パッケージの脆弱性とネットワーク到達可能性の問題を確認します。このスキャンタイプをアクティブ化すると、スキャンされているインスタンスの数をインスタンスタブで表示できます。

Amazon ECR スキャン

このスキャンタイプは、Amazon ECR のコンテナイメージとコンテナリポジトリをスキャンします。このスキャンタイプをアクティブ化すると、プライベートレジストリのスキャン設定が基本スキャンから拡張スキャンに変更されます。Amazon ECR スキャンをアクティブ化すると、コンテナイメージとコンテナリポジトリタブでスキャンされているイメージとリポジトリの数を表示できます。

Lambda 標準スキャン + Lambda コードスキャン

Lambda 標準スキャンはデフォルトの Lambda スキャンタイプです。Lambda 標準スキャンをアクティブ化すると、過去 90 日間に呼び出または更新された限り、すべての Lambda 関数がソフトウェアの脆弱性についてスキャンされます。Lambda 標準スキャンをアクティブ化すると、Lambda 関数タブにスキャンされている Lambda 関数の数が表示されます。

Lambda コードスキャンは、Lambda 関数でカスタムアプリケーションコードをスキャンします。Lambda コードスキャンをアクティブ化すると、Lambda 関数が過去 90 日間に呼び出または更新された限り、すべての Lambda 関数にコードの脆弱性がスキャンされます。Lambda 標準スキャンをアクティブ化すると、Lambda 関数タブでコードの脆弱性についてスキャンされている Lambda 関数の数を表示できます。

> **Note**
>
> Lambda 標準スキャンと Lambda コードスキャンは、個別に、または一緒にアクティブ化できます。

Amazon Inspector コードセキュリティ

このスキャンタイプは、ファーストパーティアプリケーションコード、サードパーティーアプリケーションの依存関係、Infrastructure as Code の脆弱性をスキャンします。Code Security を有効にする

と、Amazon Inspector はスキャン設定に基づいてコードリポジトリのコード脆弱性のスキャンを開始します。Amazon Inspector Code Security をアクティブ化すると、Code repositories タブでスキャンされているコードリポジトリの数を表示できます。

スキャンのアクティブ化

次の手順では、Amazon Inspector でスキャンタイプをアクティブ化する方法について説明します。

> **Note**
>
> AWS 組織の委任管理者である場合は、シェルスクリプトを使用して、複数のリージョンの複数のアカウントの Amazon Inspector スキャンタイプを有効にできます。詳細については、GitHub の inspector2-enablement-with-cli を参照してください。それ以外の場合は、Amazon Inspector の委任管理者としてサインインしながら、次の手順を実行します。

Console

スキャンをアクティブ化するには

1. https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. ページの右上隅にある AWS リージョン セレクターを使用して、新しいスキャンタイプをアクティブ化するリージョンを選択します。
3. ナビゲーションペインで、[アカウント管理] を選択します。
4. [アカウント管理] ページで、スキャンタイプをアクティブ化するアカウントを選択します。
5. [アクティブ化] を選択し、アクティブ化するスキャンタイプを選択します。
6. （推奨) スキャンタイプをアクティブ化する各 AWS リージョン で、これらのステップを繰り返します。

API

有効化 APL オペレーションを実行します。リクエストには、スキャンをアクティブ化するアカウント ID、冪等性トークン、およびそのタイプのスキャンをアクティブ化するための、`EC2`、`ECR`、`LAMBDA`、または `resourceTypes` の `LAMBDA_CODE` を 1 つ以上指定します。

Amazon Inspector による Amazon EC2 インスタンスのスキャン

Amazon Inspector EC2 スキャンは、EC2 インスタンスからメタデータを抽出し、このメタデータをセキュリティアドバイザリから収集されたルールと比較します。Amazon Inspector はインスタンスをスキャンして、パッケージの脆弱性とネットワーク到達可能性の問題について調べ、検出結果を生成します。Amazon Inspector は、12 時間に 1 回ネットワーク到達可能性スキャンを実行し、EC2 インスタンスに関連付けられたスキャン方法に依存する可変頻度で脆弱性スキャンをパッケージ化します。

パッケージの脆弱性スキャンは、エージェントベースまたはエージェントレスのスキャン方式を使用して実行できます。これらのスキャン方法はいずれも、Amazon Inspector がパッケージ脆弱性スキャンのために EC2 インスタンスからソフトウェアインベントリを収集する方法とタイミングを決定します。エージェントベースのスキャンは SSM Agent を使用してソフトウェアインベントリを収集し、エージェントレススキャンは Amazon EBS スナップショットを使用してソフトウェアインベントリを収集します。

Amazon Inspector は、アカウントに対してアクティブ化したスキャン方式を使用します。初めて Amazon Inspector をアクティブ化すると、アカウントは両方のスキャン方式を使用するハイブリッドスキャンに自動的に登録されます。ただし、いつでもこの設定を変更できます。スキャンタイプをアクティブ化する方法の詳細については、「スキャンタイプをアクティブ化する」を参照してください。このセクションでは、Amazon EC2 スキャンについて説明します。

> **Note**
>
> Amazon EC2 スキャンは、ディープインスペクションによってプロビジョニングされていても、仮想環境に関連するファイルシステムディレクトリをスキャンしません。たとえば、パス`/var/lib/docker/`はコンテナの実行時間に一般的に使用されるため、スキャンされません。

エージェントベースのスキャン

エージェントベースのスキャンは、対象となるすべてのインスタンスで SSM エージェントを使用して継続的に実行されます。エージェントベースのスキャンの場合、Amazon Inspector は SSM 関連付けと、これらの関連付けを通じてインストールされたプラグインを使用して、インスタンスからソフトウェアインベントリを収集します。Amazon Inspector のエージェントベースのスキャンでは、オペレーティングシステムパッケージに対するパッケージの脆弱性スキャンに加えて、Linux ベース

のインスタンス内のアプリケーションプログラミング言語パッケージに対するパッケージの脆弱性も Linux ベースの Amazon EC2 インスタンス向け Amazon Inspector 詳細検査 によって検出できます。

以下のプロセスでは、Amazon Inspector が SSM を使用してインベントリを収集し、エージェントベースのスキャンを実行する方法について説明します。

1. Amazon Inspector は、インスタンスからインベントリを収集するために、アカウントに SSM 関連付けを作成します。一部のインスタンスタイプ (Windows および Linux) では、これらの関連付けによって、インベントリが収集するために個々のインスタンスにプラグインがインストールされます。
2. Amazon Inspector は SSM を使用してインスタンスからパケージインベントリを抽出します。
3. Amazon Inspector は抽出されたインベントリを評価し、検出された脆弱性について検出結果を生成します。

> **Note**
>
> エージェントベースのスキャンの場合、Amazon EC2 インスタンスは同じ で SSM によって管理される必要があります AWS アカウント。

対象インスタンス

Amazon Inspector は、インスタンスが以下の条件を満たす場合、エージェントベース方式でスキャンします。

- サポートされている OS がインスタンスに備わっている。サポートされている OS のリストについては、「the section called “サポートされているオペレーティングシステム: Amazon EC2スキャン”」の「エージェントベースのスキャンのサポート」コラムを参照してください。
- インスタンスが Amazon Inspector の EC2 除外タグによってスキャンから除外されていない。
- SSM マネージドインスタンスである。エージェントを確認および設定する手順については、「SSM Agent の設定」を参照してください。

エージェントベーススキャンの動作

エージェントベースのスキャン方式を使用する場合、Amazon Inspector は、以下の状況において EC2 インスタンスの脆弱性スキャンを新たに開始します。

- 新しい EC2 インスタンスを起動するとき。
- 既存の EC2 インスタンス (Linux および Mac) に新しいソフトウェアをインストールするとき。
- Amazon Inspector がデータベースに新しい共通脆弱性識別子 (CVE) 項目を追加し、その CVE が EC2 インスタンス (Linux と Mac) に関連する場合。

Amazon Inspector は、初回スキャンが完了すると EC2 インスタンスの [最終スキャン日] フィールドを更新します。この後、Amazon Inspector が SSM インベントリを評価したとき (デフォルトでは 30 分ごと)、またはインスタンスに影響を与える CVE が Amazon Inspector データベースに新たに追加されたためにインスタンスが再スキャンされたときに、[最終スキャン日] フィールドが更新されます。

EC2 インスタンスの脆弱性の最終スキャン日は、[アカウント管理] ページの [インスタンス] タブから、または ListCoverage コマンドを使用して確認できます。

SSM Agent の設定

Amazon Inspector がエージェントベースのスキャン方式を使用して Amazon EC2 インスタンスのソフトウェア脆弱性を検出するには、そのインスタンスが Amazon EC2 Systems Manager (SSM) のマネージドインスタンスである必要があります。SSM マネージドインスタンスには SSM Agent がインストールされ実行されており、SSM にはインスタンスを管理するアクセス許可があります。すでに SSM を使用してインスタンスを管理している場合、エージェントベースのスキャンのために必要な他の手順はありません。

SSM Agent は、一部の Amazon マシンイメージ (AMI) から作成された EC2 インスタンスにデフォルトでインストールされます。詳細については、「AWS Systems Manager ユーザーガイド」の「SSM Agent について」を参照してください。ただし、SSM Agent がインストールされている場合でも、それを手動でアクティブ化し、SSM にインスタンスを管理するためのアクセス許可の付与が必要の場合があります。

以下の手順では、IAM インスタンスプロファイルを使用して Amazon EC2 インスタンスをマネージドインスタンスとして設定する方法について説明します。この手順では、「AWS Systems Manager ユーザーガイド」にあるより詳細な情報のリンクも提供しています。

AmazonSSMManagedInstanceCore は、インスタンスプロファイルをアタッチするときに使用する推奨ポリシーです。このポリシーには、Amazon Inspector EC2 スキャンに必要なすべてのアクセス許可が含まれています。

> **Note**
>
> SSM デフォルトのホスト管理設定を使用すると、IAM インスタンスプロファイルを使用しなくても、すべての EC2 インスタンスの SSM 管理を自動化できます。詳細については、「デフォルトのホスト管理設定」を参照してください。

Amazon EC2 インスタンス用に SSM を設定する方法

1. オペレーティングシステムベンダーがまだ SSM Agent をインストールしていない場合は、SSM Agent をインストールします。詳細については、「SSM Agent の使用」を参照してください。
2. を使用して AWS CLI 、SSM エージェントが実行中であることを確認します。詳細については、「SSM Agent ステータスの確認とエージェントの起動」を参照してください。
3. SSM にインスタンスを管理するアクセス許可を付与します。IAM インスタンスプロファイルを作成してインスタンスにアタッチすることで、アクセス許可を付与できます。AmazonSSMManagedInstanceCore ポリシーを使用することをお勧めします。このポリシーには、Amazon Inspector がスキャンに必要な SSM ディストリビューター、SSM インベントリ、および SSM ステートマネージャーのアクセス許可があるためです。これらの許可を持つインスタンスプロファイルを作成してインスタンスにアタッチする手順については、「Configure instance permissions for Systems Manager」を参照してください。
4. (オプション) SSM Agent の自動アップデートをアクティブ化にします。詳細については、「SSM Agent への更新の自動化」を参照してください。
5. (オプション) Amazon Virtual Private Cloud (Amazon VPC) エンドポイントを使用するように Systems Manager を設定します。詳細については、「Create Amazon VPC endpoints」を参照してください。

> **Important**
>
> Amazon Inspector では、ソフトウェアアプリケーションインベントリを収集するために、お客様のアカウントに Systems Manager ステートマネージャーの関連付けが必要です。まだ存在しない場合は、Amazon Inspector によって `InspectorInventoryCollection-do-not-delete` という名前の関連付けが自動的に作成されます。
> Amazon Inspector ではリソースデータの同期も必要で、まだ存在しない場合は自動的に `InspectorResourceDataSync-do-not-delete` という名前で作成されます。詳細については、「AWS Systems Manager ユーザーガイド」の「インベントリのリソースデータの同期の設定」を参照してください。各アカウントは、リージョンごとに設定された数のリ

ソースデータを同期することができます。詳細については、「SSM エンドポイントとクォータでのリソースデータ同期の最大数 (リージョン AWS アカウント あたり) 」を参照してください。

スキャン用に作成された SSM リソース

Amazon Inspector では、Amazon EC2 スキャンを実行するためにアカウントに多数の SSM リソースが必要です。Amazon Inspector EC2 スキャンを初めてアクティブ化すると、以下のリソースが作成されます。

> **Note**
>
> アカウントで Amazon Inspector Amazon EC2 スキャンがアクティブ化されている間にこれらの SSM リソースのいずれかが削除された場合、Amazon Inspector は次のスキャン間隔でそれらのリソースの再作成を試みます。

`InspectorInventoryCollection-do-not-delete`

これは Amazon Inspector が Amazon EC2 インスタンスからソフトウェアアプリケーションインベントリを収集するために使用するSystems Manager ステートマネージャー (SSM) 関連付けです。`InstanceIds*` からインベントリを収集するための SSM 関連付けがすでにアカウントに存在する場合、Amazon Inspector は独自のものを作成する代わりにその関連付けを使用します。

`InspectorResourceDataSync-do-not-delete`

これは Amazon Inspector が、Amazon EC2 インスタンスから収集したインベントリデータを所有する Amazon S3 バケットに送信するために使用するリソースデータ同期です。詳細については、「AWS Systems Manager ユーザーガイド」の「インベントリのリソースデータの同期の設定」を参照してください。

`InspectorDistributor-do-not-delete`

これは Amazon Inspector が Windows インスタンスのスキャンに使用する SSM 関連付けです。この関連付けにより、Windows インスタンスに Amazon Inspector SSM プラグインがインストールされます。プラグインファイルが誤って削除された場合、この関連付けは次の関連付け間隔でプラグインを再インストールします。

`InvokeInspectorSsmPlugin-do-not-delete`

これは Amazon Inspector が Windows インスタンスのスキャンに使用する SSM 関連付けです。この関連付けにより、Amazon Inspector はプラグインを使用してスキャンを開始できます。また、これを使用して Windows インスタンスのスキャンのカスタム間隔を設定することもできます。詳細については、「Windows インスタンススキャンのカスタムスケジュールの設定」を参照してください。

`InspectorLinuxDistributor-do-not-delete`

これは Amazon Inspector が Amazon EC2 Linux 詳細検査に使用する SSM 関連付けです。この関連付けにより、Amazon Inspector SSM プラグインが Linux インスタンスにインストールされます。

`InvokeInspectorLinuxSsmPlugin-do-not-delete`

これは Amazon Inspector が Amazon EC2 Linux 詳細検査に使用する SSM 関連付けです。この関連付けにより、Amazon Inspector はプラグインを使用してスキャンを開始できます。

> **Note**
>
> Amazon Inspector Amazon EC2 のスキャンまたは詳細検査を非アクティブ化すると、SSM リソース `InvokeInspectorLinuxSsmPlugin-do-not-delete` は呼び出されなくなります。

エージェントレススキャン

アカウントがハイブリッドスキャンモードの場合、Amazon Inspector は対象となるインスタンスに対してエージェントレススキャン方式を使用します。ハイブリッドスキャンモードには、エージェントベースのスキャンとエージェントレススキャンが含まれ、Amazon EC2 スキャンをアクティブ化すると自動的に有効になります。

エージェントレススキャンの場合、Amazon Inspector は EBS スナップショットを使用してインスタンスからソフトウェアインベントリを収集します。エージェントレススキャンでは、オペレーティングシステムとアプリケーションプログラミング言語パッケージの脆弱性についてインスタンスがスキャンされます。

> **Note**
>
> Linux インスタンスのアプリケーションプログラミング言語パッケージの脆弱性をスキャンする場合、エージェントレス方式では使用可能なすべてのパスがスキャンされます。一方、エージェントベーススキャンでは、デフォルトパスと、Linux ベースの Amazon EC2 インスタンス向け Amazon Inspector 詳細検査 の一部として指定した追加パスのみがスキャンされます。これにより、同じインスタンスでも、エージェントベース方式を使用してスキャンされたか、エージェントレス方式を使用してスキャンされたかによって、異なる検出結果が得られる可能性があります。

以下のプロセスでは、Amazon Inspector が EBS スナップショットを使用してインベントリを収集し、エージェントレススキャンを実行する方法について説明します。

1. Amazon Inspector は、インスタンスにアタッチされたすべてのボリュームの EBS スナップショットを作成します。Amazon Inspector が使用している間、スナップショットはアカウントに保存され、タグキーとして `InspectorScan`、タグ値として一意のスキャン ID でタグ付けされます。
2. Amazon Inspector は、EBS ダイレクト API を使用してスナップショットからデータを取得し、脆弱性がないか評価します。検出された脆弱性に対して検出結果が生成されます。
3. Amazon Inspector は、アカウント内に作成した EBS スナップショットを削除します。

対象インスタンス

Amazon Inspector は、インスタンスが以下の条件を満たす場合、エージェントレス方式でスキャンします。

- サポートされている OS がインスタンスに備わっている。詳細については、「the section called “サポートされているオペレーティングシステム: Amazon EC2スキャン”」の「エージェントベースのスキャンサポート」列を参照してください。
- インスタンスのステータスが `Unmanaged EC2 instance`、`Stale inventory`、または `No inventory` である。
- インスタンスが Amazon EBS によってバックアップされ、以下のいずれかのファイルシステム形式である。
  - `ext3`
  - `ext4`

- xfs
- インスタンスが Amazon EC2 除外タグを通じてスキャンから除外されていない。
- インスタンスにアタッチされたボリュームの数が 8 未満で、合計サイズが 1200 GB 以下である。

エージェントレススキャンの動作

アカウントが [ハイブリッドスキャン] 用に設定されている場合、Amazon Inspector は 24 時間ごとに対象インスタンスに対してエージェントレススキャンを実行します。Amazon Inspector は、新たに対象となるインスタンスを 1 時間ごとに検出してスキャンします。これには、SSM エージェントのない新しいインスタンス、またはステータスが SSM_UNMANAGED に変わった既存のインスタンスが含まれます。

Amazon Inspector は、エージェントレススキャンの後にインスタンスから抽出されたスナップショットをスキャンするたびに、Amazon EC2 インスタンスの [最終スキャン日] フィールドを更新します。

EC2 インスタンスの脆弱性の最終スキャン日は、[アカウント管理] ページの [インスタンス] タブから、または ListCoverage コマンドを使用して確認できます。

スキャンモードの管理

アカウントで EC2 スキャンを実行するときに Amazon Inspector がどのスキャン方式を使用するかは、EC2 スキャンモードで決まります。アカウントのスキャンモードは、[全般設定] の [EC2 スキャン設定] ページで確認できます。スタンドアロンアカウントまたは Amazon Inspector の委任管理者は、スキャンモードを変更できます。Amazon Inspector の委任管理者としてスキャンモードを設定すると、そのスキャンモードが組織内のすべてのメンバーアカウントに設定されます。Amazon Inspector には以下のスキャンモードがあります。

エージェントベースのスキャン — このスキャンモードでは、Amazon Inspector はパッケージの脆弱性をスキャンする際にエージェントベースのスキャン方式のみを使用します。このスキャンモードは、アカウント内の SSM マネージドインスタンスのみをスキャンしますが、新しい CVE やインスタンスへの変更に応じて継続的にスキャンできるという利点があります。エージェントベースのスキャンでは、対象となるインスタンスに対して Amazon Inspector の詳細検査を行うこともできます。これは、新しくアクティブ化されたアカウントではデフォルトのスキャンモードです。

ハイブリッドスキャン — このスキャンモードでは、Amazon Inspector はエージェントベースの方式とエージェントレス方式の両方を組み合わせてパッケージの脆弱性をスキャンします。SSM エージェントがインストールおよび設定されている適格な EC2 インスタンスでは、Amazon Inspector

はエージェントベースの方式を使用します。SSM マネージドではない対象インスタンスの場合、Amazon Inspector は対象の EBS-backed インスタンスに対してエージェントレス方式を使用します。

スキャンモードを変更するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ページの右上隅にある AWS リージョン セレクターを使用して、EC2 スキャンモードを変更するリージョンを選択します。
3. サイドナビゲーションパネルの [全般設定] から、[EC2 スキャン設定] を選択します。
4. [スキャンモード] で [編集] を選択します。
5. スキャンモードを選択し、[変更を保存] を選択します。

Amazon Inspector スキャンからのインスタンスの除外

Linux および Windows インスタンスを Amazon Inspector スキャンから除外するには、これらのインスタンスに `InspectorEc2Exclusion` キーをタグ付けします。タグ値を含めることはオプションです。タグの追加の詳細については、「Amazon EC2 リソースのタグ付け」を参照してください。

Amazon Inspector スキャンから除外するようにインスタンスにタグ付けすると、Amazon Inspector はインスタンスを除外済みとしてマークし、その検出結果を作成しません。ただし、Amazon Inspector SSM プラグインは引き続き呼び出されます。プラグインが呼び出されないようにするには、インスタンスメタデータのタグへのアクセスを許可する必要があります。

> **Note**
>
> 除外されたインスタンスには課金されません。

さらに、そのボリュームの暗号化に使用される AWS KMS キーに タグを付けることで、暗号化された EBS ボリュームをエージェントレススキャンから除外できます`InspectorEc2Exclusion`。詳細については、「キーのタグ付け」を参照してください。

サポートされるオペレーティングシステム

Amazon Inspector は、サポートされている Mac、Windows、Linux の EC2 インスタンスをスキャンして、オペレーティングシステムパッケージの脆弱性を検出します。Linux インスタンスの場

合、Amazon Inspector では Linux ベースの Amazon EC2 インスタンス向け Amazon Inspector 詳細検査 を使用してアプリケーションプログラミング言語パッケージの結果を生成できます。Mac および Windows インスタンスの場合、オペレーティングシステムパッケージのみがスキャンされます。

SSM エージェントなしでスキャンできるオペレーティングシステムなど、サポートされているオペレーティングシステムについては、「Amazon EC2 インスタンスのステータス値」を参照してください。

Linux ベースの Amazon EC2 インスタンス向け Amazon Inspector 詳細検査

Amazon Inspector は Amazon EC2 スキャンの対象範囲を拡大し、詳細検査を含めました。詳細検査により、Amazon Inspector は Linux ベースの Amazon EC2 インスタンス内のアプリケーションプログラミング言語パッケージのパッケージ脆弱性を検出します。Amazon Inspector は、プログラミング言語パッケージライブラリのデフォルトパスをスキャンします。ただし、Amazon Inspector がデフォルトでスキャンするパスに加えて、カスタムパスを設定できます。

> **Note**
>
> 詳細検査は、デフォルトのホスト管理設定で使用できます。ただし、`ssm:PutInventory` および `ssm:GetParameter` アクセス許可で設定されたロールを作成または使用する必要があります。

Amazon Inspector は、Linux ベースの Amazon EC2 インスタンスの詳細検査スキャンを実行するため、Amazon Inspector SSM プラグインで収集されたデータを使用します。Amazon Inspector SSM プラグインを管理し、Linux 用の詳細検査を実行するために、Amazon Inspector はアカウントに SSM 関連付け `InvokeInspectorLinuxSsmPlugin-do-not-delete` を自動的に作成します。Amazon Inspector は、6 時間ごとに Linux ベースの Amazon EC2 インスタンスから更新されたアプリケーションインベントリを収集します。

> **Note**
>
> Windows または Mac インスタンスでは詳細検査はサポートされていません。

このセクションでは、Amazon Inspector がスキャンするカスタムパスを設定する方法など、Amazon EC2 インスタンスの Amazon Inspector 詳細検査を管理する方法について説明します。

トピック

- 詳細検査へのアクセスまたは無効化
- Amazon Inspector 詳細検査のカスタムパス
- Amazon Inspector 詳細検査のカスタムスケジュール
- サポートされているプログラミング言語

詳細検査へのアクセスまたは無効化

> **Note**
>
> 2023 年 4 月 17 日以降に Amazon Inspector をアクティブ化したアカウントの場合、Amazon EC2 スキャンの一環として詳細検査が自動的にアクティブ化されます。

詳細検査を管理するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから全般設定を選択し、Amazon EC2 スキャン設定を選択します。
3. [Amazon EC2 インスタンスの詳細検査] で、組織または独自のアカウントのカスタムパスを設定できます。

GetEc2DeepInspectionConfiguration API を使用して、アクティブ化のステータスをプログラムで 1 つのアカウントに対して確認できます。BatchGetMemberEc2DeepInspectionStatus API を使用して、複数のアカウントでアクティブ化のステータスをプログラムで確認できます。

2023 年 4 月 17 日より前に Amazon Inspector をアクティベートした場合は、コンソールバナーまたは UpdateEc2DeepInspectionConfiguration API を使用して詳細検査をアクティベートできます。組織において Amazon Inspector の委任管理者である場合は、BatchUpdateMemberEc2DeepInspectionStatus API を使用して、自分とメンバーアカウントに対して詳細検査をアクティブ化できます。

UpdateEc2DeepInspectionConfiguration API を使用して詳細検査を非アクティブ化できます。組織のメンバーアカウントは詳細検査を非アクティブ化することはできません。その代わりに、委任管理者が BatchUpdateMemberEc2DeepInspectionStatus API を使用してメンバーアカウントを非アクティブ化する必要があります。

Amazon Inspector 詳細検査のカスタムパス

Amazon Inspector が Linux Amazon EC2 インスタンスの詳細検査を実行するときにスキャンするカスタムパスを設定できます。カスタムパスを設定すると、Amazon Inspector はそのディレクトリとその中のすべてのサブディレクトリにあるパッケージをスキャンします。

すべてのアカウントで、最大 5 個のカスタムパスを定義できます。組織の委任管理者は、10 個のカスタムパスを定義できます。

Amazon Inspector は、すべてのアカウントで Amazon Inspector がスキャンする以下のデフォルトパスに加えて、すべてのカスタムパスをスキャンします。

- `/usr/lib`
- `/usr/lib64`
- `/usr/local/lib`
- `/usr/local/lib64`

> **Note**
>
> カスタムパスはローカルパスである必要があります。Amazon Inspector は、ネットワークファイルシステムマウントや Amazon S3 ファイルシステムマウントなどのマッピングされたネットワークパスをスキャンしません。

カスタムパスのフォーマット

カスタムパスは 256 文字より長くすることはできません。以下は、カスタムパスがどのように表示されるかの例です。

パスの例

```
/home/usr1/project01
```

> **Note**
>
> インスタンスあたりのパッケージ制限は 5,000 です。パッケージインベントリの最大収集時間は 15 分です。これらの制限を避けるため、Amazon Inspector ではカスタムパスを選択することをお勧めします。

Amazon Inspector コンソールと Amazon Inspector API でのカスタムパスの設定

次の手順は、Amazon Inspector コンソールと Amazon Inspector API で Amazon Inspector 詳細検査のカスタムパスを設定する方法を示しています。カスタムパスを設定すると、Amazon Inspector は次の詳細検査にパスを含めます。

Console

1. 委任管理者 AWS Management Console として にサインインし、https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. AWS リージョン セレクターを使用して、Lambda 標準スキャンをアクティブ化するリージョンを選択します。
3. ナビゲーションペインから、[全般設定]、[EC2 スキャン設定] の順に選択します。
4. [独自のアカウントのカスタムパス] で、[編集] を選択します。
5. テキストボックスにカスタムパスを入力します。
6. [保存] を選択します。

API

UpdateEc2DeepInspectionConfiguration コマンドを実行します。`packagePaths` には、スキャンするパスの配列を指定します。

Amazon Inspector 詳細検査のカスタムスケジュール

デフォルトでは、Amazon Inspector は 6 時間ごとに Amazon EC2 インスタンスからアプリケーションインベントリを収集します。ただし、次のコマンドを実行して、Amazon Inspector がこれを実行する頻度を制御できます。

コマンド例 1: 関連付けを一覧表示して関連付け ID と現在の間隔を表示する

次のコマンドは、関連付け `InvokeInspectorLinuxSsmPlugin-do-not-delete` の関連付け ID を示しています。

```
aws ssm list-associations \
--association-filter-list "key=AssociationName,value=InvokeInspectorLinuxSsmPlugin-do-not-delete" \
--region your-Region
```

コマンド例 2: 関連付けを更新して新しい間隔を含める

次のコマンドは、関連付け `InvokeInspectorLinuxSsmPlugin-do-not-delete` の関連付け ID を使用します。`schedule-expression` のレートは、6 時間から新しい間隔 (12 時間など) に設定できます。

```
aws ssm update-association \
--association-id "your-association-ID" \
--association-name "InvokeInspectorLinuxSsmPlugin-do-not-delete" \
--schedule-expression "rate(6 hours)" \
--region your-Region
```

> **Note**
>
> ユースケースに応じて、`schedule-expression` のレートを 6 時間から 30 分などの間隔に設定すると、毎日の ssm インベントリ制限を超える可能性があります。これにより結果が遅延し、部分的なエラーステータスを持つ Amazon EC2 インスタンスが発生する可能性があります。

サポートされているプログラミング言語

Linux インスタンスの場合、Amazon Inspector の詳細検査により、アプリケーションプログラミング言語パッケージとオペレーティングシステムパッケージの検出結果を生成できます。

Mac および Windows インスタンスの場合、Amazon Inspector の詳細検査はオペレーティングシステムパッケージに対してのみ検出結果を生成できます。

サポートされているプログラミング言語の詳細については、「サポートされているプログラミング言語: Amazon EC2 詳細検査」を参照してください。

Amazon Inspector による Windows EC2 インスタンスのスキャン

Amazon Inspector は、サポートされているすべて Windows インスタンスを自動的に検出し、追加のアクションなしで継続的スキャンにそれらを含めます。サポートされているインスタンスの詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。Amazon Inspector は定期的に Windows スキャンを実行します。Windows インスタンスは検出時にスキャンされ、その後 6 時間ごとにスキャンされます。ただし、最初のスキャン後にデフォルトのスキャン間隔を調整できます。

Amazon EC2 スキャンがアクティブ化されると、Amazon Inspector は Windows リソースに対して以下の SSM 関連付けを作成します: `InspectorDistributor-do-not-delete`、`InspectorInventoryCollection-do-not-delete` および `InvokeInspectorSsmPlugin-do-not-delete`。Windows インスタンスに Amazon Inspector SSM プラグインをインストールするため、`InspectorDistributor-do-not-delete` SSM 関連付けは、`AWS-ConfigureAWSPackage` SSM ドキュメントと `AmazonInspector2-InspectorSsmPlugin` SSM Distributor パッケージを使用します。詳細については、「 の Amazon Inspector SSM プラグインWindows」を参照してください。インスタンスデータを収集し、Amazon Inspector の検出結果を生成するため、`InvokeInspectorSsmPlugin-do-not-delete` SSM 関連付けは Amazon Inspector SSM プラグインを 6 時間の間隔で実行します。ただし、cron 式または rate 式を設定して、この設定をカスタマイズできます。

> **Note**
>
> Amazon Inspector は、更新されたオープン脆弱性評価言語 (OVAL) 定義ファイルを S3 バケット `inspector2-oval-prod-`*`your-AWS-Region`* にステージングします。Amazon S3 バケットには、スキャンで使用される OVAL 定義が含まれています。これらの OVAL 定義は変更しないでください。それ以外の場合、Amazon Inspector はリリース時に新しい CVE をスキャンしません。

Windows インスタンの Amazon Inspector スキャン要件

Windows インスタンスをスキャンするには、Amazon Inspector ではインスタンスが次の基準を満たす必要があります。

- インスタンスは SSM マネージドインスタンスです。インスタンスをスキャン用に設定する手順については、「SSM Agent の設定」を参照してください。
- インスタンスオペレーティングシステムは、サポートされている Windows オペレーティングシステムの 1 つです。サポートされているオペレーティングシステムの完全なリストについては、「Amazon EC2 インスタンスのステータス値」を参照してください。
- インスタンスには Amazon Inspector SSM プラグインがインストールされています。Amazon Inspector は、マネージドインスタンスの検出時に Amazon Inspector SSM プラグインを自動的にインストールします。プラグインの詳細については、次のトピックを参照してください。

> **Note**
>
> ホストがインターネットにアクセスできない Amazon VPC で動作している場合、Windows スキャンでは、ホストがリージョンの Amazon S3 エンドポイントにアクセスできる必要があります。Amazon S3 Amazon VPC エンドポイントの設定方法については、「Amazon Virtual Private Cloud User Guide」の「Create a gateway endpoint」を参照してください。Amazon VPC エンドポイントポリシーが外部 S3 バケットへのアクセスを制限している場合は、インスタンスの評価に使用される OVAL 定義 AWS リージョン を保存する で Amazon Inspector が管理するバケットへのアクセスを特に許可する必要があります。このバケットは次の形式になります。`inspector2-oval-prod-REGION`

Windows インスタンススキャンのカスタムスケジュールの設定

SSM を使用して Windows 関連付けの cron 式または rate 式を設定することで、`InvokeInspectorSsmPlugin-do-not-delete` Amazon EC2 インスタンススキャンの間隔をカスタマイズできます。詳細については、「AWS Systems Manager ユーザーガイド」の「リファレンス: Systems Manager の Cron 式および rate 式」を参照するか、次の手順を使用してください。

次のコード例から選択し、rate 式または cron 式を使用して Windows インスタンスのスキャン間隔をデフォルトの 6 時間から 12 時間に変更します。

次の例では、`InvokeInspectorSsmPlugin-do-not-delete` という名前の関連付けに AssociationId を使用する必要があります。以下の AWS CLI コマンドを実行して AssociationId を取得できます。

```
$ aws ssm list-associations --association-filter-list
 "key=AssociationName,value=InvokeInspectorSsmPlugin-do-not-delete" --region us-east-1
```

> **Note**
>
> AssociationId はリージョン別であるため、まずそれぞれの一意の ID を取得する必要があります AWS リージョン。その後、コマンドを実行して、Windows インスタンスのカスタムスキャンスケジュールを設定したい各リージョンのスキャン頻度を変更できます。

Example rate expression

```
$ aws ssm update-association \
```

```
--association-id "YourAssociationId" \
--association-name "InvokeInspectorSsmPlugin-do-not-delete" \
--schedule-expression "rate(12 hours)"
```

Example cron expression

```
$ aws ssm update-association \
--association-id "YourAssociationId" \
--association-name "InvokeInspectorSsmPlugin-do-not-delete" \
--schedule-expression "cron(0 0/12 * * ? *)"
```

Amazon Inspector による Amazon Elastic Container Registry コンテナイメージのスキャン

Amazon Inspector は、Amazon Elastic Container Registry に保存されているコンテナイメージをスキャンしてソフトウェアの脆弱性がないかを調べ、パッケージ脆弱性の検出結果を生成します。Amazon ECR のスキャンをアクティブ化するときは、Amazon Inspector をプライベートレジストリの優先スキャンサービスとして設定します。

> **Note**
>
> Amazon ECR はレジストリポリシーを使用して、 AWS プリンシパルにアクセス許可を付与します。このプリンシパルには、スキャンのために Amazon Inspector APIsを呼び出すために必要なアクセス許可があります。レジストリポリシーの範囲を設定するときは、`ecr:*` `PutRegistryScanningConfiguration`アクションまたは を に追加しないでください`deny`。これにより、Amazon ECR のスキャンを有効または無効にすると、レジストリレベルでエラーが発生します。

基本スキャンでは、プッシュ時にスキャンするか、手動スキャンを実行するようにレポジトリを設定できます。拡張スキャンでは、オペレーティングシステムとプログラミング言語パッケージの脆弱性をレジストリレベルでスキャンします。基本スキャンと拡張スキャンの違いを並べて比較するには、「Amazon Inspector に関するよくある質問」を参照してください。

> **Note**
>
> 基本スキャンは Amazon ECR を通じて提供され、請求されます。詳細については、「Amazon Elastic Container Registry の料金」を参照してください。拡張スキャンは

Amazon Inspector を通じて提供され、請求されます。詳細については、「Amazon Inspector の料金」を参照してください。

Amazon ECR スキャンをアクティブ化する方法については、「スキャンタイプをアクティブ化する」を参照してください。検出結果を表示する方法の詳細については、「Amazon Inspector での検出結果の管理」を参照してください。検出結果をイメージレベルで表示する方法の詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「イメージスキャン」を参照してください。また、 AWS Security Hub や Amazon EventBridge など、基本スキャンに AWS のサービス 使用できない の検出結果を管理することもできます。

このセクションでは、Amazon ECR スキャンに関する情報と、Amazon ECR リポジトリの拡張スキャンを設定する方法について説明します。

Amazon ECR スキャンのスキャン動作

Amazon ECR スキャンを初めてアクティブ化すると、Amazon Inspector は過去 14 日以内にプッシュされたイメージを検出します。次にAmazon Inspector はイメージをスキャンし、スキャンステータスを に設定しますactive。連続スキャンが有効になっている場合、Amazon Inspector は 14 日以内にプッシュされたイメージ (デフォルトでは)、last-in-use日が 14 日以内 (デフォルトでは)、またはイメージが設定された再スキャン期間内にスキャンされる限り、イメージをモニタリングします。2025 年 5 月 16 日より前に作成された Amazon Inspector アカウントの場合、デフォルト設定では、過去 90 日以内にプッシュまたはプルされたイメージを再スキャンしてモニタリングします。詳細については、「Amazon ECR 再スキャン期間の設定」を参照してください。

継続的スキャンの場合、Amazon Inspector は、以下の状況でコンテナイメージの新しい脆弱性スキャンを開始します。

- 新しいコンテナイメージがプッシュされる場合。
- Amazon Inspector がデータベースに新しい共通脆弱性識別子 (CVE) 項目を追加し、その CVE がそのコンテナイメージに関連する場合 (継続的スキャンのみ)。

リポジトリをプッシュスキャン用に設定した場合、イメージはプッシュ時にのみスキャンされます。

コンテナイメージの脆弱性の最終確認日は、[アカウント管理] ページの [コンテナイメージ] タブから、または ListCoverage API を使用して確認できます。Amazon Inspector は、以下のイベントに応じて Amazon ECR イメージの [最終スキャン日] フィールドを更新します。

- Amazon Inspector がコンテナイメージの初回スキャンを完了した場合。

- Amazon Inspector がコンテナイメージを再スキャンしたとき。これは、そのコンテナイメージに影響を与える新しい共通脆弱性識別子 (CVE) 項目が Amazon Inspector データベースに追加されたためです。

実行中のコンテナへのコンテナイメージのマッピング

Amazon Inspector は、コンテナイメージを Amazon Elastic Container Service (Amazon ECS) と Amazon Elastic Kubernetes Service (Amazon EKS) 全体の実行中のコンテナにマッピングすることで、包括的なコンテナセキュリティ管理を提供します。これらのマッピングは、実行中のコンテナ上のイメージの脆弱性に関するインサイトを提供します。

> **Note**
>
> マネージドポリシー`AWSReadOnlyAccess`だけでは、Amazon ECR イメージと実行中のコンテナ間のマッピングを表示するための十分なアクセス許可は提供されません。コンテナイメージマッピング情報を表示するには、 `AWSReadOnlyAccess`と`AWSInspector2ReadOnlyAccess`の両方の管理ポリシーが必要です。

この機能を使用すると、運用リスクに基づいて修復作業に優先順位を付け、コンテナエコシステム全体のセキュリティカバレッジを維持できます。現在使用中のコンテナイメージと、過去 24 時間以内に Amazon ECS または Amazon EKS クラスターでコンテナイメージが最後に使用された日時をモニタリングできます。新しいイメージまたはアカウントの場合、データが利用可能になるまでに最大 36 時間かかることがあります。その後、このデータは 24 時間に 1 回更新されます。この情報は、コンテナイメージの検出結果の詳細画面の Amazon Inspector コンソールを介して検出結果で利用でき、 `ecrImageInUseCount` および `ecrImageLastInUseAt`フィルターを介して Amazon Inspector API で利用できます。

> **Note**
>
> このデータは、Amazon ECR スキャンをアクティブ化し、継続的スキャン用にリポジトリを設定すると、Amazon ECR の検出結果に自動的に送信されます。継続的スキャンは、Amazon ECR リポジトリレベルで設定する必要があります。詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「拡張スキャン」を参照してください。

また、last-in-use日に基づいてクラスターからコンテナイメージを再スキャンすることもできます。

この機能は Amazon ECS Amazon EKS Fargate でもサポートされています。

サポートされているオペレーティングシステムとメディアタイプ

サポートされるオペレーティングシステムの詳細については、「サポートされているオペレーティングシステム: Amazon Inspector による Amazon ECR スキャン」を参照してください。

Amazon ECR リポジトリの Amazon Inspector スキャンは、サポートされている以下のメディアタイプを対象としています。

イメージマニフェスト

- `"application/vnd.oci.image.manifest.v1+json"`
- `"application/vnd.docker.distribution.manifest.v2+json"`

イメージ設定

- `"application/vnd.docker.container.image.v1+json"`
- `"application/vnd.oci.image.config.v1+json"`

イメージレイヤー

- `"application/vnd.docker.image.rootfs.diff.tar"`
- `"application/vnd.docker.image.rootfs.diff.tar.gzip"`
- `"application/vnd.docker.image.rootfs.foreign.diff.tar.gzip"`
- `"application/vnd.oci.image.layer.v1.tar"`
- `"application/vnd.oci.image.layer.v1.tar+gzip"`
- `"application/vnd.oci.image.layer.v1.tar+zstd"`
- `"application/vnd.oci.image.layer.nondistributable.v1.tar"`
- `"application/vnd.oci.image.layer.nondistributable.v1.tar+gzip"`

> **Note**
>
> Amazon Inspector は、Amazon ECR リポジトリのスキャン用の`"application/vnd.docker.distribution.manifest.list.v2+json"`メディアタイプをサポートしていません。

Amazon ECR 再スキャン期間の設定

Amazon ECR 再スキャン期間の設定により、Amazon Inspector がリポジトリのコンテナイメージを継続的にモニタリングする期間が決定します。イメージlast-in-use日、最終プル日、プッシュ日の再スキャン期間を設定します。ベストプラクティスとして、環境に最適な再スキャン期間を設定します。

イメージを頻繁に構築する場合は、短いスキャン期間を選択します。長期間使用される画像の場合は、長いスキャン期間を選択します。組織に追加された新しいアカウントを含む新しいアカウントのデフォルトのスキャン期間は 14 日間です。

Amazon Inspector は、クラスターで最後に使用されているか、14 日以内にプッシュされている限り (デフォルトでは) 、イメージのモニタリングと再スキャンを継続します。設定されたプッシュ日と最終使用日内に、実行中のコンテナでイメージがプッシュまたは最終使用されていない場合、Amazon Inspector はイメージのモニタリングを停止します。必要に応じて、最終使用日ではなく、最終プル日でイメージをモニタリングするように設定を変更することもできます。Amazon Inspector がイメージのモニタリングを停止すると、イメージスキャンステータスコードを非アクティブに設定し、理由コードを期限切れに設定します。次に、Amazon Inspector は、関連するすべてのイメージ検出結果を終了するようスケジュールします。

プッシュ日の期間を延ばすと、Amazon Inspector は、継続スキャン用に設定されたリポジトリでアクティブにスキャンされているすべてのイメージにその変更を適用します。ただし、非アクティブなイメージは、新しい期間内にプッシュした場合でも非アクティブのままになります。

> **Note**
>
> 委任管理者アカウントから再スキャン期間を設定すると、その設定は組織内のすべてのメンバーアカウントに適用されます。委任管理者アカウントが Amazon ECR スキャンを有効にしていない場合、API イメージのクラスターを表示することはできません。

> **Note**
>
> 2025 年 5 月 16 日より前に設定されたすべての再スキャン期間設定は変更されません。以前に設定したデフォルト設定を引き続き使用できます。

イメージの再スキャン期間

イメージの再スキャン期間は、Amazon Inspector がイメージをモニタリングする期間を決定します。イメージの再スキャン期間には、最終使用日 (デフォルト) または最終プル日の 2 つのモードが含まれます。Amazon ECS/Amazon EKS クラスターアクティビティから最終使用日を使用する場合は、最終使用日 (デフォルト) を選択します。Amazon ECR イメージの最後のプル日を使用してイメージを再スキャンする場合は、Last pull date を選択します。再スキャン期間として、以下のオプションが利用できます。

- 14 日間 (デフォルト)
- 30 日間
- 60 日
- 90 日間
- 180 日間

イメージプッシュ日の期間

イメージのプッシュ日の期間は、Amazon Inspector がリポジトリにプッシュされた後にイメージを継続的にモニタリングする期間を決定します。再スキャン期間として、以下のオプションが利用できます。

- 14 日間 (デフォルト)
- 30 日間
- 60 日
- 90 日間
- 180 日間
- 有効期間

Amazon ECR の再スキャン期間を設定するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. Amazon ECR の再スキャン期間 AWS リージョン を設定する を選択します。
3. ナビゲーションペインから、[全般設定]、[ECR スキャン設定] の順に選択します。
4. ECR 再スキャン期間で、イメージの再スキャンモードを選択し、対応する期間を選択します。
5. イメージプッシュ日 で、イメージプッシュ日を選択します。
6. [保存] を選択します。

Amazon Inspector を使用した AWS Lambda 関数のスキャン

Amazon Inspector の AWS Lambda 関数とレイヤーのサポートにより、継続的な自動化されたセキュリティ脆弱性評価が提供されます。Amazon Inspector では、2 種類の Lambda 関数スキャンを提供しています。

Amazon Inspector Lambda 標準スキャン

これはデフォルトの Lambda スキャンタイプです。Lambda 標準スキャンでは、Lambda 関数とレイヤー内のアプリケーションの依存関係をスキャンして、パッケージの脆弱性について調べます。

Amazon Inspector Lambda コードスキャン

このスキャンタイプでは、Lambda 関数やレイヤー内のカスタムアプリケーションコードをスキャンして、コードの脆弱性について調べます。Lambda 標準スキャンをアクティブ化することも、Lambda コードスキャンと同時に Lambda 標準スキャンをアクティブ化することもできます。

Lambda 関数スキャンをアクティブ化すると、Amazon Inspector はアカウントに以下の AWS CloudTrail サービスにリンクされたチャネルを作成します: `cloudtrail:CreateServiceLinkedChannel` および `cloudtrail:DeleteServiceLinkedChannel`。Amazon Inspector はこれらのチャネルを管理し、それらを使用して CloudTrail イベントのスキャンをモニタリングします。これらのチャネルにより、CloudTrail で証跡があったかのように、アカウントの CloudTrail イベントを表示できます。CloudTrail で独自の証跡を作成して、アカウントのイベントを管理することをお勧めします。

Lambda 関数スキャンをアクティブ化する方法については、「スキャンタイプをアクティブ化する」を参照してください。このセクションでは、Lambda 関数のスキャンについて説明します。

Lambda 関数スキャンのスキャン動作

Amazon Inspector はアクティベーション時に、アカウント内で過去 90 日間に呼び出された、または更新されたすべての Lambda 関数をスキャンします。Amazon Inspector は、次のような状況で Lambda 関数の脆弱性スキャンを開始します。

- Amazon Inspector が既存の Lambda 関数を検出した時。
- 新しい Lambda 関数が Lambda サービスにデプロイされた場合。
- 既存の Lambda 関数またはそのレイヤーのアプリケーションコードまたは依存関係に更新がデプロイされた場合。
- Amazon Inspector がデータベースに新しい共通脆弱性識別子 (CVE) 項目を追加し、その CVE が関数に関連している場合。

Amazon Inspector は、各 Lambda 関数が削除されるかスキャンから除外されるまで、そのライフタイム期間を通じてモニタリングします。

Lambda 関数の脆弱性の最終確認日は、[アカウント管理] ページの [Lambda 関数] タブから、または ListCoverage API を使用して確認できます。Amazon Inspector は、以下のイベントに応じて Lambda 関数の [最終スキャン日] フィールドを更新します。

- Amazon Inspector が Lambda 関数の初回スキャンを完了した時。
- Lambda 関数の更新時。
- Amazon Inspector が Lambda 関数を再スキャンしたとき。これは、その関数に影響する新しい CVE 項目が Amazon Inspector データベースに追加されたためです。

サポートされているランタイムと対象となる関数

Amazon Inspector は、Lambda 標準スキャンと Lambda コードスキャンのさまざまなランタイムをサポートしています。各スキャンタイプでサポートされているランタイムのリストについては、「サポートされているランタイム: Amazon Inspector Lambda 標準スキャン」と「サポートされているランタイム: Amazon Inspector Lambda コードスキャン」を参照してください。

Lambda 関数が Amazon Inspector スキャンの対象となるには、ランタイムがサポートされていることに加えて、以下の基準を満たす必要があります。

- この関数は過去 90 日間に呼び出されたか、または更新されています。

- この関数は $LATEST にマークされています。
- この関数はタグによるスキャンから除外されていません。

> **Note**
>
> 過去 90 日間に呼び出されたり変更されたりしていない Lambda 関数は、自動的にスキャンから除外されます。Amazon Inspector は、再度呼び出されたり、Lambda 関数コードに変更が加えられたりする場合に、自動的に除外された関数のスキャンを再開します。

Amazon Inspector Lambda 標準スキャン

Amazon Inspector Lambda 標準スキャンでは、Lambda 関数のコードとレイヤーに追加したアプリケーションパッケージの依存関係内にあるソフトウェアの脆弱性を特定します。たとえば、Lambda 関数が既知の脆弱性があるバージョンの `python-jwt` パッケージを使用している場合、Lambda 標準スキャンはその関数の検出結果を生成します。

Amazon Inspector が Lambda 関数のアプリケーションパッケージの依存関係に脆弱性を検出すると、Amazon Inspector は詳細なパッケージ 脆弱性タイプの検出結果を生成します。

スキャンタイプをアクティブ化する手順については、「スキャンタイプをアクティブ化する」を参照してください。

> **Note**
>
> Lambda 標準スキャンは、Lambda ランタイム環境にデフォルトでインストールされている AWS SDK 依存関係をスキャンしません。Amazon Inspector は、関数コードとともにアップロードされた依存関係、またはレイヤーから継承された依存関係のみをスキャンします。

> **Note**
>
> Amazon Inspector Lambda 標準スキャンを非アクティブ化すると、Amazon Inspector Lambda コードスキャンも非アクティブ化になります。

Lambda 標準スキャンから関数の除外

Lambda 関数にタグを追加して、Amazon Inspector Lambda 標準スキャンから除外することができます。スキャンから関数を除外すると、実行不可能なアラートを防ぐことができます。除外する関数にタグを付ける場合、タグには次のキーと値のペアが必要です。

- キー: `InspectorExclusion`
- 値: `LambdaStandardScanning`

このトピックでは、スキャンから除外する関数にタグを付ける方法について説明します。Lambda へのタグの追加について詳しくは、「Lambda 関数でのタグの使用」を参照してください。

スキャンから関数を除外するには

1. 認証情報を使用してサインインし、Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. ナビゲーションペインで、[関数] を選択します。
3. Amazon Inspector Lambda 標準スキャンから除外する関数の名前を選択します。
4. [Configuration] (設定)、[Tags] (タグ) の順に選択します。
5. [タグを管理]、[新しいタグを追加] の順に選択します。
   a. [Key] (キー) に「`InspectorExclusion`」と入力します。
   b. [Value] (値) に「`LambdaStandardScanning`」と入力します。
6. [Save] を選択します。

Amazon Inspector Lambda コードスキャン

> **Important**
> この機能では、Lambda 関数のスニペットをキャプチャして、検出された脆弱性をハイライトします。これらのスニペットでは、ハードコーディングされた認証情報やその他の機密の内容が表示される場合があります。

この機能を使用すると、Amazon Inspector は Lambda 関数内のアプリケーションコードをスキャンして、 AWS セキュリティのベストプラクティスに基づいてコードの脆弱性を検出し、データ漏

洩、インジェクションの欠陥、暗号化の欠落、暗号化の弱さを検出します。Amazon Inspector は、自動推論と機械学習を使用して Lambda 関数アプリケーションコードを評価します。また、Amazon CodeGuru と共同で開発された内部ディテクターを使用して、ポリシー違反と脆弱性を識別します。詳細については、「CodeGuru Detector Library」を参照してください。

Amazon Inspector は、Lambda 関数アプリケーションコードで脆弱性を検出すると、コードの脆弱性を生成します。この検出結果タイプには、問題と、コード内のその問題を見つけることができる場所を示すコードスニペットが含まれています。また、問題を修正する方法も提案します。提案には、脆弱なコード行を置き換えるために使用できるプラグアンドプレイのコードブロックが含まれます。これらのコード修正は、この検出結果タイプに対する一般的なコード修正ガイダンスに加えて提供されます。

コード修正案は、自動推論と生成 AI サービスを利用しています。一部のコード修復提案は、意図したとおりに機能しない場合があります。採用するコード修正案に対する責任はユーザーにあります。採用する前に、必ず修正案を確認してください。コードが意図したとおりに動作するように、編集が必要になる場合があります。詳細については、「責任ある AI ポリシー」を参照してください。

Lambda コードスキャンは、単独で、または Lambda 標準スキャンと同時にアクティブ化できます。詳細については、「スキャンタイプをアクティブ化する」を参照してください。がこの機能 AWS リージョン をサポートする方法については、「」を参照してくださいリージョン固有機能の可用性。

コードの脆弱性検出結果におけるコードの暗号化

CodeGuru は、Lambda コードスキャンを使用したコード脆弱性の検出結果に関連して検出されたコードスニペットを保存します。デフォルトでは、CodeGuru はコードの暗号化に使用される AWS 所有キーを制御します。ただし、Amazon Inspector API を通じた暗号化のために、独自のカスタマー管理キーを使用できます。詳細については、「検出結果のコードの保管時の暗号化」を参照してください

Lambda コードスキャンから関数の除外

Lambda 関数にタグを追加して、Amazon Inspector Lambda コードスキャンから除外することができます。スキャンから関数を除外すると、実行不可能なアラートを防ぐことができます。除外する関数にタグを付ける場合、タグには次のキーと値のペアが必要です。

- キー – `InspectorCodeExclusion`
- 値 – `LambdaCodeScanning`

このトピックでは、コードスキャンから除外する関数にタグを付ける方法について説明します。Lambda へのタグの追加について詳しくは、「Lambda 関数でのタグの使用」を参照してください。

コードスキャンから関数を除外するには

1. 認証情報を使用してサインインし、Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. ナビゲーションペインで、[関数] を選択します。
3. Amazon Inspector Lambda コードスキャンから除外する関数の名前を選択します。
4. [Configuration] (設定)、[Tags] (タグ) の順に選択します。
5. [タグを管理]、[新しいタグを追加] の順に選択します。
   a. [Key] (キー) に「`InspectorCodeExclusion`」と入力します。
   b. [Value] (値) に「`LambdaCodeScanning`」と入力します。
6. [Save] を選択します。

Amazon Inspector でのスキャンタイプの非アクティブ化

スキャンタイプを非アクティブ化すると、そのスキャンタイプによって生成された検出結果にはアクセスできなくなります。スキャンタイプを再アクティブ化すると、Amazon Inspector はすべての対象リソースをスキャンして新しい検出結果を生成します。検出結果の記録を保持する場合は、それを検出結果レポートとして Amazon Simple Storage Service (Amazon S3) バケットにエクスポートできます。詳細については、「Amazon Inspector 検出結果レポートのエクスポート」を参照してください。スキャンタイプを非アクティブ化すると、スキャンタイプを非アクティブ化した AWS アカウントで次の変更が発生することがあります。

Amazon EC2 スキャン

アカウントの Amazon Inspector Amazon EC2 スキャンを非アクティブ化すると、次の SSM 関連付けが削除されます。

- `InspectorDistributor-do-not-delete`
- `InspectorInventoryCollection-do-not-delete`
- `InspectorLinuxDistributor-do-not-delete`
- `InvokeInspectorLinuxSsmPlugin-do-not-delete`

- `InvokeInspectorSsmPlugin-do-not-delete`.

さらに、Amazon Inspector SSM プラグインはすべてのWindowsホストから削除されます。詳細については、「Windows EC2 インスタンスのスキャン」を参照してください。

Amazon ECR スキャン

アカウントの Amazon ECR スキャンを非アクティブ化すると、Amazon ECR スキャンタイプのアカウントが Amazon Inspector による拡張スキャンから Amazon ECR による基本スキャンに変わります。

Lambda 標準スキャン

アカウントの Lambda 標準スキャンを非アクティブ化する場合、スキャンタイプがアクティブになっていれば、Lambda コードスキャンは非アクティブ化されます。また、Lambda 標準スキャンをアクティブ化するときに Amazon Inspector が作成する CloudTrail サービスにリンクされたチャネルも削除します。

Amazon Inspector コードセキュリティ

アカウントのコードセキュリティを無効にすると、それに関連付けられたすべての統合、プロジェクト、スキャン設定が削除されます。アカウントが組織の委任管理者である場合は、アカウントの Code Security のみを非アクティブ化し、memeber アカウントはスタンドアロンアカウントになります。

スキャンの非アクティブ化

アカウントのすべてのスキャンタイプを非アクティブ化すると、 AWS リージョン内のそのアカウントの Amazon Inspector も非アクティブ化されます 詳細については、「Amazon Inspector の非アクティブ化」を参照してください。

マルチアカウント環境でこの手順を完了するには、Amazon Inspector の委任された管理者としてサインインした状態で以下の手順を行います。

Console

スキャンを非アクティブ化するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。

2. ページの右上隅にある AWS リージョン セレクターを使用して、スキャンを非アクティブ化するリージョンを選択します。
3. ナビゲーションペインで、[アカウント管理] を選択します。
4. [アカウント] タブを選択すると、アカウントのスキャンステータスが表示されます。
5. スキャンを非アクティブ化するアカウントのチェックボックスを選択します。
6. [アクション] を選択し、[非アクティブ化] オプションから、非アクティブ化するスキャンタイプを選択します。
7. （推奨) スキャンタイプを非アクティブ化する各 AWS リージョン で、これらのステップを繰り返します。

API

無効化 APL オペレーションを実行します。リクエストには、スキャンを非アクティブ化するアカウント ID を指定し、`resourceTypes` についてはスキャンを非アクティブ化するために、`EC2`、`ECR`、`LAMBDA`、または `LAMBDA_CODE` の 1 つ以上を指定します。

Amazon EC2 インスタンスオペレーティングシステムの Center for Internet Security (CIS) スキャン

Amazon Inspector CIS スキャン (CIS スキャン) はAmazon EC2 インスタンスオペレーティングシステムをベンチマークして、Center for Internet Security によって確立されたベストプラクティスの推奨事項に従って設定されていることを確認します。CIS Security Benchmarks は、システムを安全に設定するための業界標準の設定ベースラインとベストプラクティスを提供しています。アカウントの Amazon Inspector EC2 スキャンを有効にした後、CIS スキャンを実行またはスケジュールできます。Amazon EC2 スキャンをアクティブ化する方法については、「スキャンタイプをアクティブ化する」を参照してください。

> **Note**
>
> CIS 標準は x86_64 オペレーティングシステムを対象としています。一部のチェックは評価されないか、ARM ベースのリソースで無効な修復手順が返される場合があります。

Amazon Inspector は、インスタンスタグと定義されたスキャンスケジュールに基づいて、ターゲット Amazon EC2 インスタンスに対して CIS スキャンを実行します。Amazon Inspector は、ターゲットとなる各インスタンスに対して一連のインスタンスチェックを実行します。各チェックでは、システム設定が特定の CIS Benchmark 推奨事項を満たしているかどうかを評価します。各チェックには CIS チェック ID とタイトルがあり、そのプラットフォームの CIS Benchmark 推奨事項に対応します。CIS スキャンが完了すると、結果を表示して、そのシステムのどのインスタンスチェックが合格した、スキップされた、または不合格になったかを確認できます。

> **Note**
>
> CIS スキャンを実行またはスケジュールするには、安全なインターネット接続が必要です。ただし、プライベートインスタンスで CIS スキャンを実行する場合は、VPC エンドポイントを使用する必要があります。

トピック

- Amazon Inspector CIS スキャンに関する Amazon EC2 インスタンスの要件
- CIS スキャンの実行
- を使用した Amazon Inspector CIS スキャンの管理に関する考慮事項 AWS Organizations

- Amazon Inspector CIS スキャンに使用される Amazon Inspector 所有の Amazon S3 バケット
- CIS スキャン設定の作成
- CIS スキャン結果の表示
- CIS スキャン設定の編集
- CIS スキャン結果のダウンロード

Amazon Inspector CIS スキャンに関する Amazon EC2 インスタンスの要件

Amazon EC2 インスタンスで CIS スキャンを実行するには、Amazon EC2 インスタンスが次の基準を満たしている必要があります。

- インスタンスオペレーティングシステムは、CIS スキャンに対してサポートされているオペレーティングシステムの 1 つです。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。
- インスタンスは Amazon EC2 Systems Manager インスタンスです。詳細については、「AWS Systems Manager ユーザーガイド」の「SSM Agent の使用」を参照してください。
- インスタンスには Amazon Inspector SSM プラグインがインストールされています。Amazon Inspector は、このプラグインをマネジードインスタンスに自動的にインストールします。
- インスタンスには、SSM がインスタンスを管理するためのアクセス許可と、Amazon Inspector がそのインスタンスの CIS スキャンを実行するためのアクセス許可を付与するインスタンスプロファイルがあります。これらのアクセス許可を付与するには、AmazonSSMManagedInstanceCore ポリシーと AmazonInspector2ManagedCisPolicy ポリシーを IAM ロールにアタッチします。次に、IAM ロールをインスタンスプロファイルとしてインスタンスにアタッチします。インスタンスプロファイルを作成してアタッチする手順については、「Amazon EC2 ユーザーガイド」の「IAM ロールの使用」を参照してください。

> **Note**
>
> Amazon EC2 インスタンスで CIS スキャンを実行する前に、Amazon Inspector 詳細検査を有効にする必要はありません。Amazon Inspector 詳細検査を無効にすると、Amazon Inspector は自動的に SSM Agent をインストールしますが、SSM Agent は詳細検査を実行するために呼び出されなくなります。ただし、その結果、`InspectorLinuxDistributor-do-not-delete` の関連付けはアカウントにあります。

プライベート Amazon EC2 インスタンスで CIS スキャンを実行するための Amazon Virtual Private Cloud エンドポイントの要件

CIS スキャンは、Amazon ネットワーク経由で Amazon EC2 インスタンスで実行できます。ただし、プライベート Amazon EC2 インスタンスで CIS スキャンを実行する場合は、Amazon VPC エンドポイントを作成する必要があります。Systems Manager 用の Amazon VPC エンドポイントを作成するときは、以下のエンドポイントが必要です。

- `com.amazonaws.region.ec2messages`
- `com.amazonaws.region.inspector2`
- `com.amazonaws.region.s3`
- `com.amazonaws.region.ssm`
- `com.amazonaws.region.ssmmessages`

詳細については、「AWS Systems Manager ユーザーガイド」の「Systems Manager の VPC エンドポイントを作成する」を参照してください。

> **Note**
>
> 現在、一部の AWS リージョン は`amazonaws.com.region.inspector2`エンドポイントをサポートしていません。

CIS スキャンの実行

CIS スキャンは、オンデマンドで 1 回実行するか、スケジュールされた定期スキャンとして実行できます。スキャンを実行するには、まずスキャン設定を作成します。

スキャン設定を作成するときは、ターゲットインスタンスに使用するタグキーと値のペアを指定します。組織の Amazon Inspector 委任管理者である場合は、スキャン設定で複数のアカウントを指定できます。Amazon Inspector は、それらのアカウントごとに指定されたタグを持つインスタンスを検索します。スキャンの CIS ベンチマークレベルを選択します。各ベンチマークについて、CIS はレベル 1 およびレベル 2 プロファイルをサポートしており、さまざまな環境が必要とする異なるレベルのセキュリティのベースラインを提供するように設計されています。

- レベル 1 – あらゆるシステムで設定できる基本的なセキュリティ設定を推奨します。これらの設定を実装しても、サービスの中断はほとんどまたはまったく発生しません。これらの推奨事項の目的

は、システムへのエントリポイントの数を減らし、全体的なサイバーセキュリティリスクを減らすことです。

- レベル 2 – セキュリティの高い環境に対して、より高度なセキュリティ設定を推奨します。これらの設定を実装するには、ビジネスへの影響のリスクを最小限に抑えるための計画と調整が必要です。これらの推奨事項の目的は、規制の順守達成を支援することです。

レベル 2 はレベル 1 を拡張します。レベル 2 を選択すると、Amazon Inspector はレベル 1 とレベル 2 に推奨されるすべての設定をチェックします。

スキャンのパラメータを定義したら、設定完了後に実行する 1 回限りのスキャンとして実行するか、定期スキャンとして実行するかを選択できます。定期スキャンは、毎日、毎週、または毎月、任意のタイミングで実行できます。

> **Tip**
> スキャンの実行中にシステムに影響を与える可能性が最も低い日時を選択することをお勧めします。

を使用した Amazon Inspector CIS スキャンの管理に関する考慮事項 AWS Organizations

組織で CIS スキャンを実行すると、Amazon Inspector の委任管理者とメンバーアカウントは CIS スキャン設定とスキャン結果を異なる方法で操作します。

Amazon Inspector の委任管理者が CIS スキャン設定とスキャン結果を操作する方法

委任管理者が、すべてのアカウントまたは特定のメンバーアカウントに対してスキャン設定を作成すると、組織はその設定を所有します。組織が所有するスキャン設定には、組織 ID を所有者として指定する ARN があります。

```
arn:aws:inspector2:Region:111122223333:owner/OrganizationId/cis-configuration/scanId
```

委任管理者は、別のアカウントが作成した場合でも、組織が所有しているスキャン設定を管理できます。

委任管理者は、組織内の任意のアカウントのスキャン結果を表示できます。

委任管理者がスキャン設定を作成し、ターゲットアカウントとして SELF を指定すると、委任管理者は組織を離れてもスキャン設定を所有します。ただし、委任管理者は、SELF をターゲットとするスキャン設定のターゲットを変更することはできません。

> **Note**
>
> 委任管理者は、組織が所有している CIS スキャン設定にタグを追加することはできません。

Amazon Inspector メンバーアカウントが CIS スキャン設定とスキャン結果を操作する方法

メンバーアカウントが CIS スキャン設定を作成すると、その設定を所有します。ただし、委任管理者は設定を表示できます。メンバーアカウントが組織を離れると、委任管理者は設定を表示できなくなります。

> **Note**
>
> 委任管理者は、メンバーアカウントが作成したスキャン設定を編集することはできません。

メンバーアカウント、SELF をターゲットとする委任管理者、およびスタンドアロンアカウントはすべて、作成するスキャン設定を所有します。これらのスキャン設定には、所有者としてアカウント ID を示す ARN があります。

```
arn:aws:inspector2:Region:111122223333:owner/111122223333/cis-
configuration/scanId
```

メンバーアカウントは、アカウントのスキャン結果を表示できます。これには、委任管理者がスケジュールした CIS スキャンの結果が含まれます。

Amazon Inspector CIS スキャンに使用される Amazon Inspector 所有の Amazon S3 バケット

Open Vulnerability and Assessment Language (OVAL) は、コンピュータシステムのマシンの状態を評価および報告する方法を標準化する情報セキュリティの取り組みです。次の表は、CIS スキャンに使用される OVAL 定義を持つ Amazon Inspector 所有の Amazon S3 バケットをすべて示しています。Amazon Inspector は、CIS スキャンに必要な OVAL 定義ファイルをステージングしま

す。Amazon Inspector が所有する Amazon S3 バケットは、必要に応じて VPC で許可リストに登録する必要があります。

> **Note**
>
> 以下の Amazon Inspector 所有の Amazon S3 バケットの詳細は変更の対象ではありません。ただし、新しくサポートされた AWS リージョンを反映するようにテーブルが更新される場合があります。Amazon Inspector が所有する Amazon S3 バケットを他の Amazon S3 オペレーションや独自の Amazon S3 バケットで使用することはできません。

CIS バケット	AWS リージョン
`cis-datasets-prod-arn-5908f6f`	欧州 (ストックホルム)
`cis-datasets-prod-bah-8f88801`	中東 (バーレーン)
`cis-datasets-prod-bjs-0f40506`	中国 (北京)
`cis-datasets-prod-bom-435a167`	アジアパシフィック (ムンバイ)
`cis-datasets-prod-cdg-f3a9c58`	欧州 (パリ)
`cis-datasets-prod-cgk-09eb12f`	アジアパシフィック (ジャカルタ)
`cis-datasets-prod-cmh-63030b9`	米国東部 (オハイオ)
`cis-datasets-prod-cpt-02c5c6f`	アフリカ (ケープタウン)
`cis-datasets-prod-dub-984936f`	欧州 (アイルランド)
`cis-datasets-prod-fra-6eb96eb`	欧州 (フランクフルト)
`cis-datasets-prod-gru-de69f99`	南米 (サンパウロ)
`cis-datasets-prod-hkg-8e30800`	アジアパシフィック (香港)
`cis-datasets-prod-iad-8438411`	米国東部 (バージニア北部)
`cis-datasets-prod-icn-f4eff1c`	アジアパシフィック (ソウル)

CIS バケット	AWS リージョン
cis-datasets-prod-kix-5743b21	アジアパシフィック (大阪)
cis-datasets-prod-lhr-8b1fbd0	欧州 (ロンドン)
cis-datasets-prod-mxp-7b1bbce	欧州 (ミラノ)
cis-datasets-prod-nrt-464f684	アジアパシフィック (東京)
cis-datasets-prod-osu-5bead6f	AWS GovCloud (米国東部)
cis-datasets-prod-pdt-adadf9c	AWS GovCloud (米国西部)
cis-datasets-prod-pdx-acfb052	米国西部 (オレゴン)
cis-datasets-prod-sfo-1515ba8	米国西部 (北カリフォルニア)
cis-datasets-prod-sin-309725b	アジアパシフィック (シンガポール)
cis-datasets-prod-syd-f349107	アジアパシフィック (シドニー)
cis-datasets-prod-yul-5e0c95e	カナダ (中部)
cis-datasets-prod-zhy-5a8eacb	中国 (寧夏)
cis-datasets-prod-zrh-67e0e3d	欧州 (チューリッヒ)

CIS スキャン設定の作成

このトピックでは、CIS スキャン設定を作成する方法について説明します。

CIS スキャンを実行するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. AWS リージョン ドロップダウンを使用して、CIS スキャンを実行する AWS リージョン を選択します。
3. ナビゲーションペインから [オンデマンドスキャン] を選択し、[CIS スキャン] を選択します。

4. [新しいスキャンを作成] を選択します。
5. [スキャン設定名] で、[スキャン設定名] を入力します。
6. [ターゲットリソースのタグ] で、スキャンするインスタンスの [キー] と対応する [値] を入力します。各キーに最大 5 つの異なる値を指定し、合計 25 個のタグをスキャンに含めることができます。
7. [CIS Benchmark レベル] で、基本的なセキュリティ設定には [レベル 1]、高度なセキュリティ設定には [レベル 2] を選択できます。
8. [ターゲットアカウント] で、CIS スキャンに含めるアカウントを指定します。詳細については、「を使用した Amazon Inspector CIS スキャンの管理に関する考慮事項 AWS Organizations」を参照してください。

   アカウントが委任管理者アカウントである場合は、[すべてのアカウント] を選択するか、[アカウントを指定] を選択できます。[すべてのアカウント] オプションでは、組織内のすべてのアカウントが対象となります。[アカウントを指定] は、組織内の個別のアカウントのみが対象です。このオプションを選択した場合は、アカウント番号をカンマで区切って、複数のアカウントを指定できます。アカウント ID の代わりに SELF を入力して、アカウントのスキャン設定を作成することもできます。

   アカウントが組織内のスタンドアロンアカウントまたはメンバーアカウントである場合は、[セルフ] を選択してアカウントのスキャン設定を作成できます。
9. [スケジュール] で、スキャン設定の作成が完了するとすぐに実行される [1 回限りのスキャン]、または指定した時点で実行される [定期的なスキャン] を選択します。
10. 選択内容を確認し、[削除] を選択します。

CIS スキャン結果の表示

Amazon Inspector は、一意のスキャン ID を使用してスキャンの結果を実行および収集するスキャン設定ごとに、スキャンジョブを作成します。CIS スキャン結果は 90 日間利用できます。CIS スキャン結果は、チェックまたはスキャンされたリソース別に表示できます。

- スキャン結果をチェック別に集計 – スキャン中に実行された個別のチェックごとにスキャン結果をグループ化します。チェックごとに、不合格になった、スキップされた、または合格したリソースの数に関するレポートが表示されます。
- スキャンされたリソース別に集計されたスキャン結果 – スキャン中にスキャン対象とするスキャンされたリソースごとにスキャン結果をグループ化します。リソースごとに、リソースが不合格になった、スキップされた、または合格したチェックの数に関するレポートが表示されます。

このトピックでは、CIS スキャンの結果を表示する方法について説明します。

スキャン結果を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. AWS リージョン ドロップダウンを使用して、CIS スキャン設定を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [オンデマンドスキャン] を選択し、[CIS スキャン] を選択します。
4. [スキャン結果] タブを選択します。
5. [スケジュール作成者] 列で、表示するスキャンスケジュール ID を選択します。または、表示するスキャンスケジュール ID を持つ行を選択し、[詳細を表示] を選択します。
6. [チェック] を選択すると、実行された各チェックが表示されます。または、[スキャンされたリソース] を選択すると、スキャン中にターゲットになり、スキャンされた各リソースが表示されます。

スケジュールされた CIS スキャンの詳細を表示することもできます。

スケジュールされた CIS スキャンの詳細を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. AWS リージョン ドロップダウンを使用して、CIS スキャン設定を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [オンデマンドスキャン] を選択し、[CIS スキャン] を選択します。
4. [スケジュール] タブを選択します。
5. [スキャン設定名] 列で、表示するスキャン設定の名前を選択します。または、表示するスキャン設定で行を選択し、[詳細を表示] を選択します。

CIS スキャン設定の編集

このトピックでは、CIS スキャン設定を編集する方法について説明します。

CIS スキャン設定を編集するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. AWS リージョン ドロップダウンを使用して、CIS スキャン設定を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [オンデマンドスキャン] を選択し、[CIS スキャン] を選択します。
4. [スケジュール] タブを選択します。
5. 編集するスキャン設定ポリシーがある行を選択してから、[編集] を選択します。

CIS スキャン結果のダウンロード

Amazon Inspector コンソールまたは API を使用して、CIS スキャンの PDF または CSV をダウンロードできます。

> **Note**
> ダウンロードできるのは、2024 年 5 月 3 日以降に収集された CIS スキャンの CIS スキャン結果の CSV ファイルのみです。

このトピックでは、Amazon Inspector コンソールを使用して CIS スキャンをダウンロードする方法について説明します。

コンソールから CIS スキャン結果をダウンロードするには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. AWS リージョン ドロップダウンを使用して、CIS スキャン設定を作成した AWS リージョン を選択します。
3. ナビゲーションペインから [オンデマンドスキャン] を選択し、[CIS スキャン] を選択します。
4. [スキャン結果] タブを選択します。
5. [スケジュール作成者] 列で、表示するスキャンスケジュール ID を選択します。または、表示するスキャンスケジュール ID を持つ行を選択し、[詳細を表示] を選択します。

6. [ダウンロード] を選択してから、[PDF] または [CSV] を選択します。アカウントが委任管理者アカウントである場合は、[アカウントを選択] を選択して、特定のメンバーアカウントの結果をダウンロードできます。

Amazon Inspector コードセキュリティ

Amazon Inspector は、ソフトウェアの脆弱性や意図しないネットワークの露出についてワークロードを自動的に検出し、継続的にスキャンする脆弱性管理サービスです。Code Security を使用すると、Amazon Inspector はファーストパーティアプリケーションのソースコード、サードパーティーアプリケーションの依存関係、Infrastructure as Code の脆弱性をスキャンします。Amazon Inspector コンソールまたは Amazon Inspector API を使用してコードセキュリティをアクティブ化できます。Code Security をアクティブ化したら、スキャン設定を作成してコードリポジトリに適用し、スキャンの頻度とタイミングを決定できます。スキャン設定はいつでも表示、編集、削除できます。Code Security AWS リージョン が利用可能な の詳細については、「リージョンとエンドポイント」を参照してください。料金の詳細については、Amazon Inspector の料金」を参照してください。

> **Note**
>
> Amazon Inspector Code Security によって生成されたセキュリティ検出結果は、Amazon Inspector と Security Hub の統合では使用できません。ただし、これらの特定の検出結果には、Amazon Inspector コンソールおよび Amazon Inspector API からアクセスできます。

コードセキュリティの前提条件

コードセキュリティの使用を開始する前に、コードセキュリティをアクティブ化し、データの暗号化方法を決定する必要があります。これは、統合認証情報、コード、または統合、コードリポジトリ、プロジェクトに関連するその他の情報などです。デフォルトでは、データは AWS 所有キーで暗号化されます。つまり、キーは サービスによって作成、所有、管理されます。データの暗号化に使用されるキーを所有および管理する場合は、カスタマーマネージド KMS キーを作成できます。

コードセキュリティのアクティブ化

コードセキュリティは、すべての自動スキャンタイプをアクティブ化するのと同じ方法でアクティブ化します。詳細については、「スキャンタイプをアクティブ化する」を参照してください。

にアクセスするためのカスタマーマネージドキーの作成 AWS KMS

デフォルトでは、データは AWS 所有キーで暗号化されます。つまり、キーは サービスによって作成、所有、管理されます。データの暗号化に使用されるキーを所有および管理する場合は、カ

スタマーマネージド KMS キーを作成できます。Amazon Inspector はデータとやり取りしません。Amazon Inspector は、ソースコードプロバイダーのリポジトリからのみメタデータを取り込みます。カスタマーマネージド KMS キーの作成方法については、 AWS Key Management Service ユーザーガイドの「KMS キーの作成」を参照してください。

ポリシーの例

カスタマーマネージドキーを作成するときは、次のサンプルポリシーを使用します。

```
{
    "Version": "2012-10-17",
    "Id": "key-policy",
    "Statement": [
        {
            "Sid": "Allow Q to use Encrypt Decrypt GenerateDataKey and
 GenerateDataKeyWithoutPlaintext",
            "Effect": "Allow",
            "Principal": {
                "Service": "q.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:qdeveloper:codesecurity-scope":
 "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:inspector2:Region:111122223333:codesecurity-integration/*"
                }
            }
        },
        {
            "Sid": "Allow Q to use DescribeKey",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "q.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        },
        {
            "Sid": "Allow Inspector to use Encrypt Decrypt GenerateDataKey and
 GenerateDataKeyWithoutPlaintext using FAS",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{111122223333}:role/inspectorCodeSecurity"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "inspector2.Region.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:qdeveloper:codesecurity-scope":
 "111122223333"
                }
            }
        },
        {
            "Sid": "Allow Inspector to use DescribeKey using FAS",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{111122223333}:role/inspectorCodeSecurity"
            },
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "inspector2.Region.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

KMS キーを作成したら、次の Amazon Inspector APIsを使用できます。

- UpdateEncryptionKey – for `CODE_REPOSITORY``resourceType`と をスキャンタイプ`CODE`として使用して、カスタマーマネージド KMS キーの使用を設定します。
- GetEncryptionKey – for `CODE_REPOSITORY``resourceType`と をスキャンタイプ`CODE`として使用して、KMS キー設定の取得を設定します。
- ResetEncryptionKey – `CODE_REPOSITORY` for `resourceType`および で `CODE`を使用して KMS キー設定をリセットし、 AWS 所有の KMS キーを使用します。

Amazon Inspector とコードリポジトリ間の統合の作成

このセクションでは、Amazon Inspector とコードリポジトリの統合を作成する方法について説明します。統合を作成すると、コードセキュリティページの Amazon Inspector コンソールにすべてのコードリポジトリがプロジェクトとして一覧表示されます。このセクションの他のトピックでは、統合とプロジェクトにアクセスする方法について説明します。

Code Security は最大 100,000 個のプロジェクトのみをインポートし、各リポジトリのデフォルトブランチのみがモニタリングされます。プロジェクトは、最大 3 つのデフォルトのスキャン設定に関連付けることができます。

Code Security は、アカウントあたり最大 100 個の統合のみをサポートします。Code Security 統合には、委任管理者アカウント/メンバーアカウント関係の概念はありません。

制限が発生しないように、統合に同じホストを複数回使用しないことをお勧めします。

GitHub SaaS、、および との統合にはGitHub Enterprise Cloud、パブリックインターネットアクセスGitHub Enterprise Serverが必要です。

> **Important**
>
> サードパーティーの統合は、セキュリティ上の懸念に対処するなど、何らかの理由で事前の通知なしに一時的または永続的に無効になる場合があります。

Amazon Inspector と の統合の作成 GitHub

このトピックでは、Amazon Inspector と の統合を作成する方法について説明しますGitHub。

> **Note**
>
> 初めて統合を作成する場合は、ステップ 2 でデフォルトのスキャン設定を作成するように求められます。スキャン設定を作成するときは、スキャンするスキャン頻度、スキャン分析、リポジトリを選択します。デフォルトのスキャン設定の作成は、一般的なスキャン設定の作成と同じです。ただし、デフォルトのスキャン設定は、Amazon Inspector にインポートされた新規および既存のプロジェクトに自動的に関連付けられます。デフォルトのスキャン設定を作成する場合は、この設定を続行を選択します。デフォルトのスキャン設定は 1 回しか作成できません。デフォルトのスキャン設定を作成すると、デフォルトのスキャン設定を再度作成するように求められません。デフォルトのスキャン設定は、アカウントごとに 1 回、組織ごとに 1 回のみ作成できます。デフォルトのスキャン設定を設定しない場合は、設定をスキップを選択します。ただし、 は次回統合を作成するときにデフォルトのスキャン設定を作成するように求められます。デフォルトのスキャン設定を作成するか、デフォルトのスキャン設定の作成をスキップすると、統合ワークフローのステップ 3 に移動し、統合の詳細を入力します。

GitHub SaaS、、および との統合にはGitHub Enterprise Cloud、パブリックインターネットアクセスGitHub Enterprise Serverが必要です。

> **Note**
>
> Amazon Inspector は、デフォルトのブランチのみをスキャンおよびモニタリングします。新しいデフォルトブランチを作成すると、Amazon Inspector は新しいデフォルトブランチをスキャンして更新します。

> **Important**
>
> 統合の作成を完了する前に、Amazon Inspector と 間の接続を承認するように指示されますGitHub。この手順を完了するには、このステップを完了する必要があります。ポップアップを閉じると、続行できなくなります。

Amazon Inspector と の統合を作成するには GitHub

1. 自分の認証情報を使用してサインインします。https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。Connect to を選択し、 を選択しますGitHub。
3. 統合の詳細で、統合の名前を入力し、GitHub への接続を選択します。
4. ポップアップで Authorize を選択して、Amazon Inspector と 間の接続を作成しますGitHub。
5. 成功バナーで、GitHub 接続作成ページに移動を選択します。
6. GitHub アプリケーションのインストール ID を入力します。GitHub アプリケーションをインストールした場合は、GitHub Apps ページまたはGitHubアプリケーション URL の末尾GitHubにある でインストール ID を確認できます。GitHub アプリケーションをインストールしていない場合は、新しいアプリケーションのインストールを選択します。これにより、GitHub組織を選択し、リポジトリの範囲を指定するGitHub場所に移動します。
7. [Connect to GitHub] (GitHub に接続) を選択します。

統合を作成すると、Amazon Inspector がアクセストークンを更新できないシナリオが発生する可能性があります。これは、統合ホストが使用できない場合やAmazon Inspector で他の通信の問題が発生した場合に発生する可能性があります。問題を修正するには、コードセキュリティページの統合タブから接続を再認証します。Status 列の下には、統合が非アクティブと表示され、Amazon Inspector は再認証するオプションを提供します。再認証を選択します。接続設定を完了できる統合ワークフローにリダイレクトされます。

統合のシステム設定を削除すると、無期限に接続が失われる可能性があります。この場合、統合を削除し、新しい統合を作成する必要があります。統合を削除すると、統合に関連付けられたすべてのプロジェクトとスキャン設定が失われます。

Amazon Inspector と の統合の作成 GitLab Self Managed

このトピックでは、Amazon Inspector と のコードリポジトリの統合を作成する方法について説明しますGitLab Self Managed。

必要な情報

接続を作成するときは、以下が必要です。

- 統合名 – 統合の本文に追加された名前です。

- エンドポイント URL – GitLab Self Managedインスタンスへのアクセスに使用される URL です。
- 個人用アクセストークン – 個人用アクセストークンは管理者アカウントから に作成GitLab Self Managedされ、、`api`、`read_api``read_repository`、および のスコープを含める必要があります`write_repository`。

> **Note**
>
> Amazon Inspector は、デフォルトのブランチのみをスキャンおよびモニタリングします。新しいデフォルトブランチを作成すると、Amazon Inspector は新しいデフォルトブランチをスキャンして更新します。

Amazon Inspector と の統合の作成 GitLab Self Managed

次の手順では、Amazon Inspector と のコードリポジトリ間の接続を作成する方法について説明しますGitLab Self Managed。

> **Note**
>
> 初めて統合を作成する場合は、ステップ 2 でデフォルトのスキャン設定を作成するように求められます。スキャン設定を作成するときは、スキャンするスキャン頻度、スキャン分析、リポジトリを選択します。デフォルトのスキャン設定の作成は、一般的なスキャン設定の作成と同じです。ただし、デフォルトのスキャン設定は、Amazon Inspector にインポートされた新規および既存のプロジェクトに自動的に関連付けられます。デフォルトのスキャン設定を作成する場合は、この設定を続行を選択します。デフォルトのスキャン設定は 1 回しか作成できません。デフォルトのスキャン設定を作成すると、デフォルトのスキャン設定を再度作成するように求められません。デフォルトのスキャン設定は、アカウントごとに 1 回、組織ごとに 1 回のみ作成できます。デフォルトのスキャン設定を構成しない場合は、設定をスキップを選択します。ただし、次回統合を作成するときに、デフォルトのスキャン設定を作成するように求められます。デフォルトのスキャン設定を作成するか、デフォルトのスキャン設定の作成をスキップすると、統合ワークフローのステップ 3 に移動し、統合の詳細を入力します。

> **Important**
>
> 統合の作成を完了する前に、Amazon Inspector とGitLabセルフマネージド間の接続を承認するように求められます。この手順を完了するには、このステップを完了する必要があります。ポップアップを閉じると、続行できなくなります。

GitLab セルフマネージドとの接続を作成するには

1. 自分の認証情報を使用してサインインします。https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。Connect to を選択し、GitLab Self Managed を選択します。
3. 統合の詳細に、次のように入力します。
   a. 統合名には、統合の本文に追加された名前を入力します。
   b. エンドポイント URL には、GitLabセルフマネージドインスタンスへのアクセスに使用される URL を入力します。
   c. 個人用アクセストークンには、必要なスコープで個人用アクセストークンを入力します。
4. への接続を選択しますGitLab。
5. ポップアップウィンドウで Authorize を選択して、Amazon Inspector と 間の接続の作成を完了しますGitLab。

統合を作成すると、Amazon Inspector がアクセストークンを更新できないシナリオが発生する可能性があります。これは、統合ホストが使用できない場合やAmazon Inspector で他の通信の問題が発生した場合に発生する可能性があります。問題を修正するには、コードセキュリティページの統合タブから接続を再認証します。Status 列の下には、統合が非アクティブと表示され、Amazon Inspector は再認証するオプションを提供します。再認証を選択します。接続設定を完了できる統合ワークフローにリダイレクトされます。

統合のシステム設定を削除すると、無期限に接続が失われる可能性があります。この場合、統合を削除し、新しい統合を作成する必要があります。統合を削除すると、統合に関連付けられたすべてのプロジェクトとスキャン設定が失われます。

コードリポジトリとの統合の表示

このトピックでは、Amazon Inspector コンソールで統合を表示する方法について説明します。

Amazon Inspector コンソールで統合を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. [統合] を選択します。このタブから、設定されたすべての統合を確認し、すべての統合に関する基本情報を確認できます。この情報には、統合の名前、統合のステータス、ソースコードプロバイダー名が含まれます。

プロバイダーへの再認証

統合を作成すると、Amazon Inspector がアクセストークンを更新できないシナリオが発生する可能性があります。これは、統合ホストが使用できない場合やAmazon Inspector で他の通信の問題が発生した場合に発生する可能性があります。問題を修正するには、コードセキュリティページの統合タブから接続を再認証します。Status 列の下には、統合が非アクティブと表示され、Amazon Inspector は再認証するオプションを提供します。再認証を選択します。接続設定を完了できる統合ワークフローにリダイレクトされます。

統合のシステム設定を削除すると、無期限に接続が失われる可能性があります。この場合、統合を削除し、新しい統合を作成する必要があります。統合を削除すると、統合に関連付けられたすべてのプロジェクトとスキャン設定が失われます。

コードリポジトリの表示

このトピックでは、Amazon Inspector コンソールでコードリポジトリを表示する方法について説明します。

Amazon Inspector コンソールでコードリポジトリを表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. Code リポジトリを選択します。このタブから、プロジェクトとしてリストされているすべてのコードリポジトリを確認し、それらに関する基本情報を確認できます。この情報には、各プロジェクトの名前とスキャンステータスが含まれます。プロジェクトに関連付けられている設定と、プロジェクトが最後にスキャンされた日時を確認することもできます。検索バーでプロジェクトをフィルタリングすることもできます。

プロジェクトの詳細の表示

このトピックでは、Amazon Inspector コンソールでプロジェクトの詳細を表示する方法について説明します。アカウントが組織の委任管理者である場合は、メンバーアカウントに属するプロジェクトの詳細を表示できます。

Amazon Inspector コンソールでコードプロジェクトを表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. Code リポジトリを選択します。このタブから、プロジェクトとしてリストされているすべてのコードリポジトリを確認し、それらに関する基本情報を確認できます。この情報には、各プロジェクトの名前とスキャンステータスが含まれます。プロジェクトに関連付けられている設定と、プロジェクトが最後にスキャンされた日時を確認することもできます。検索バーでプロジェクトをフィルタリングすることもできます。
4. プロジェクトを選択します。または、プロジェクトを選択し、詳細の表示を選択します。プロジェクトの詳細画面から、プロジェクトに関する基本情報を確認できます。この情報には、プロジェクトの名前と ID、および統合 ARN が含まれます。これには、プロジェクトがスキャンされた日時と提供タイプに関する情報が含まれます。プロジェクトに関連付けられた検出結果を確認したり、検出結果をエクスポートしたり、検出結果の抑制ルールを作成することもできます。

統合の削除

次の手順では、Amazon Inspector コンソールで統合を削除する方法について説明します。統合を削除すると、統合に関連付けられたすべてのプロジェクトとスキャン設定が失われます。

Amazon Inspector コンソールで統合を削除するには。

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. [統合] を選択します。このタブから、設定されたすべての統合を確認し、すべての統合に関する基本情報を確認できます。この情報には、統合の名前、統合のステータス、統合プロバイダータイプが含まれます。
4. 統合を選択し、削除を選択します。

スキャン設定の作成

スキャン設定を作成する前に、Amazon Inspector との統合を作成する必要があります。初めて統合を作成するときは、デフォルトのスキャン設定を作成するように求められます。このトピックでは、一般的なスキャン設定を作成する方法について説明します。デフォルトのスキャン設定と一般的なスキャン設定の違いは、デフォルトのスキャン設定が新しいプロジェクトに自動的にアタッチされることです。デフォルトのスキャン設定の作成はスキップできます。

Code Security は、最大 500 個の一般的なスキャン設定のみをサポートします。コードセキュリティは、アカウントおよび組織ごとに 1 つのデフォルトのスキャン設定のみをサポートします。スキャン設定は、最大 100,000 個のプロジェクトにのみ関連付けることができます。

プロジェクトは、合計で最大 4 つのスキャン設定に関連付けることができます。これには、デフォルトのスキャン設定が作成された場合のデフォルトのスキャン設定が含まれます。組織のスキャン設定にタグを付けることはできません。

組織の委任管理者がスキャン設定を作成すると、スキャン設定は組織レベルで作成され、組織内のすべてのメンバーアカウントに適用されます。委任管理者がデフォルトのスキャン設定を作成する場合も同様です。

スキャン設定を作成するときは、スキャンするスキャン頻度、スキャン分析、リポジトリを選択します。スキャン頻度は、変更ベース、定期的、またはカスタマイズできます。変更ベースの定期的なスキャンでは、定期的なスキャンを有効にするオプションを利用できます。定期的なスキャンを有効にする場合は、スキャン頻度をスキャンが発生した曜日または月に設定します。カスタマイズされたスキャンでは、コードが変更されたときのスキャンと定期的なスキャンを有効にするオプションが提供されます。コードが変更されたときにスキャンを有効にする場合は、マージリクエストとプルリクエストに含めるスキャントリガーを指定します。

コミット ID が一定の時間内に変更されていない場合、スキャンはスキップできます。定期スキャンの場合、コミット ID が 1 週間スキャン間で変更されていない場合、スキャンはスキップされます。オンデマンドスキャンの場合、コミット ID が 24 時間以内にスキャン間で変更されていない場合、スキャンはスキップされます。

> **Note**
>
> スキャン設定にマージリクエストとプルリクエストのトリガーのみがある場合、上位 25 件の検出結果のみがソースコード管理プラットフォームでのみ表示されます。Amazon Inspector には何も表示されません。

一般的なスキャン設定を作成するには

1. 自分の認証情報を使用してサインインします。https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. 「設定」を選択し、「スキャン設定の作成」を選択します。
4. スキャンの詳細で、次の操作を行います。
   - 設定名に、スキャン設定の名前を入力します。
5. スキャン頻度で、変更ベースの定期的なスキャンまたはカスタマイズされたスキャンタイプとトリガーを選択して、コードがスキャンされる頻度を指定します。
   a. (オプション 1) 変更ベースの定期スキャンを選択した場合は、定期スキャンを有効にするまたは定期スキャンを無効にするを選択します。
      - 。 定期的なスキャンを有効にする を選択した場合は、コードをスキャンする曜日を選択してスキャン頻度を設定します。
   b. (オプション 2) カスタマイズされたスキャンを選択した場合は、コードが変更されたときにスキャンを有効にし、定期的なスキャンを有効にするかどうかを決定します。
      i. コードが変更された場合はスキャンを有効にする、コードが変更された場合はスキャンを無効にするを選択します。コードの変更時にスキャンを有効にする を選択した場合は、ドロップダウンからスキャンがトリガーされるタイミングを指定します。
      ii. 定期スキャンを有効にするまたは定期スキャンを無効にするを選択します。定期的なスキャンを有効にする を選択した場合は、コードをスキャンする曜日を選択してスキャン頻度を設定します。イベントベースのトリガーをスキャンすることもできます。これらのイベントには、デフォルトブランチに対してプルリクエストが開かれている場合と、コミットがデフォルトブランチにプッシュされる場合が含まれます。
6. スキャン分析で、完全なスキャン分析を設定するか、カスタマイズされたスキャン分析を設定するかを決定します。
   a. (オプション 1) スキャン分析を完了する を選択した場合は、次のスキャン分析をすべて適用します。
      - 静的アプリケーションセキュリティテスト – ソースコードの脆弱性を分析します。
      - IaC スキャン – インフラストラクチャを設定およびプロビジョニングするスクリプトとコードを分析します。

- 静的ソフトウェア構成分析 — アプリケーションのオープンソースパッケージを調べます。

b. (オプション 2) カスタマイズされたスキャン分析を選択した場合は、ドロップダウンメニューから前述のスキャン分析タイプを少なくとも 1 つ選択する必要があります。

7. (オプション) タグでは、プロジェクトに適用するキーと値のペアを作成します。最大 50 個のタグを作成できます。
8. [次へ] を選択します。
9. リポジトリの選択で、すべてのリポジトリまたは特定のリポジトリを選択します。

   a. (オプション 1) すべてのリポジトリを選択すると、既存のリポジトリのいずれかに対してスキャンが有効になります。

   b. (オプション 2) 特定のリポジトリを選択した場合、スキャンは指定したリポジトリに対してのみ有効になります。

10. [次へ] を選択します。
11. 選択内容を確認し、スキャン設定の作成を選択します。

> **Note**
>
> 一般的なスキャン設定は、既存のすべてのコードリポジトリにのみ適用されます。新しいコードリポジトリには適用されません。

スキャン設定の表示

次の手順では、Amazon Inspector コンソールでスキャン設定を表示する方法について説明します。

> **Note**
>
> 組織レベルでスキャン設定を表示すると、コードセキュリティ画面の詳細の一部は を反映するために異なります AWS アカウント。

スキャン設定の詳細を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。

2. ナビゲーションペインから、コードセキュリティを選択します。
3. 設定 を選択して、スキャン設定のリストを表示します。委任管理者の場合、リストには組織のスキャン設定が含まれます。各スキャン設定の名前と、各スキャン設定を作成したユーザー (AWS アカウント ID または組織 ID) を確認できます。設定に適用されているスキャンタイプとスキャン分析タイプを表示することもできます。検索バーのさまざまなフィールドでスキャン設定をフィルタリングすることもできます。

スキャン設定の詳細の表示

次の手順では、Amazon Inspector コンソールでスキャン設定の詳細を表示する方法について説明します。

スキャン設定の詳細を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. [設定] を選択します。
4. 詳細を表示する設定を選択します。スキャン設定の詳細画面には、スキャン設定の概要が表示されます。この画面から、スキャン設定 ARN、有効なスキャン頻度タイプ、有効なスキャン分析タイプを表示できます。この画面からスキャン設定を削除することもできます。組織に属するスキャン設定を表示している場合は、この画面から編集することもできます。

スキャン設定の編集

スキャン設定はいつでも編集できます。スキャン設定を編集するときは、スキャンするスキャン頻度、スキャン分析、タグ、リポジトリを変更できます。たとえば、スキャン設定を編集して、特定のリポジトリのスキャンを一時停止します。次の手順では、スキャン設定を編集する方法について説明します。

スキャン設定を編集するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. [設定] を選択します。

4. 編集する設定を選択し、編集を選択します。編集する設定を選択し、編集を選択することもできます。

スキャン設定の削除

スキャン設定はいつでも削除できます。このトピックでは、スキャン設定を削除する方法について説明します。

スキャン設定を削除するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. [設定] を選択します。
4. 削除する設定を選択し、削除を選択します。または、削除する設定を選択し、削除を選択します。

オンデマンドスキャンの実行

プロジェクトに対してオンデマンドで を実行できます。オンデマンドスキャンを実行すると、設定されたすべてのスキャン設定の和集合が、選択したプロジェクトに適用されます。アカウントが組織の委任管理者アカウントである場合は、メンバーアカウントに属するプロジェクトに対してオンデマンドスキャンを実行できます。次の手順では、Amazon Inspector コンソールでオンデマンドスキャンを実行する方法について説明します。

オンデマンドスキャンを実行するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、コードセキュリティを選択します。
3. Code リポジトリを選択します。
4. スキャンするプロジェクトを選択し、オンデマンドスキャンを選択します。

Amazon Inspector コードセキュリティでサポートされている言語

このトピックには、Amazon Inspector コードセキュリティでサポートされている言語が含まれています。

SAST でサポートされている言語

- C# (6.0 .Net 以降を除くすべてのバージョンが推奨されます)
- C (C11 以前)
- C++ (C++ 17 以前)
- Go (Go 1.18 のみ)
- Java (Java 17 以前)
- JavaScript (EMCMAScript 2021 以前)
- JSX (React 17 以前)
- Kotlin (Kotlin 2.0 以前)
- PHP (PHP 8.2 以前)
- Python (Python 3.11 Python 以前、3 シリーズ内)
- Ruby (Ruby 2.7 および 3.2 のみ)
- Rust
- Scala (Scala 3.2.2 以前)
- Shell
- TSX
- TypeScript (すべてのバージョン)

ソフトウェア構成分析でサポートされている言語

- Go (Go 1.18 のみ)
- Java (Java 17 以前)
- JavaScript (EMCMAScript 2021 以前)
- PHP (PHP 8.2 以前)
- Python (Python 3.11 Python 以前、3 シリーズ内)
- .Net
- Ruby (Ruby 2.7 および 3.2 のみ)

- Rust

Infrastructure as Code の言語

- AWS CDK（Python および TypeScript）
- AWS CloudFormation (2010 年 9 月 9 日）
- Terraform (1.6.2 以前）

コードセキュリティの無効化

コードセキュリティの無効化の詳細については、「スキャンタイプの無効化」を参照してください。

Amazon Inspector の検出結果について

Amazon Inspector は、Amazon EC2 インスタンス、Amazon ECR コンテナイメージ、Lambda 関数の脆弱性を検出すると、検出結果を生成します。また、ファーストパーティアプリケーションのソースコード、サードパーティーアプリケーションの依存関係、Infrastructure as Code で検出されたコードの脆弱性に関する検出結果も生成します。検出結果は、 AWS リソースの 1 つに影響を与える脆弱性に関する詳細なレポートです。

検出結果は脆弱性にちなんで命名され、重要度評価、影響を受ける AWS リソースと非 AWS リソースに関する情報、検出された脆弱性の修復方法の詳細を提供します。Amazon Inspector は、すべてのアクティブな検出結果を、ユーザーがそれらを修復するまで保存します。

リソースが削除、終了、またはスキャンの対象ではなくなると、Amazon Inspector はリソースに関連付けられた検出結果を自動的に閉じ、3 日後に検出結果を削除します。他の理由で検出結果が閉じられた場合は、30 日後に削除されます。

> **Note**
>
> Amazon Inspector は、脆弱性の原因となった問題が再発した場合、検出結果を閉じてから 7 日以内に修復された検出結果を再開します。

Amazon Inspector を無効にすると、検出結果は 24 時間後に削除されます。リソースが終了すると、リソースに関連する検出結果は 3 日後に削除されます。スキャンの対象ではなくなったリソースにアタッチされた検出結果でも同じことが起こります。がアカウント AWS を停止すると、検出結果は 90 日後に削除されます。停止したインスタンスの検出結果はアクティブのままです。

検出結果の状態

Amazon Inspector では、検出結果は以下の状態に分類されます。

アクティブ

Amazon Inspector は、修正されていない検出結果を [アクティブ] として分類します。

抑制

Amazon Inspector は、1 つ以上の抑制ルールの対象となる検出結果を [抑制] として分類します。

[Closed] (クローズ)

検出結果が修正されると、Amazon Inspector は検出結果を [終了] として分類します。

トピック

- Amazon Inspector の検出結果タイプ
- Amazon Inspector 検出結果の表示
- Amazon Inspector 検出結果の詳細の表示
- Amazon Inspector スコアの表示と脆弱性インテリジェンスの詳細の理解
- Amazon Inspector の検出結果の重要度レベルについて

Amazon Inspector の検出結果タイプ

このセクションでは、Amazon Inspector のさまざまな検出結果タイプについて説明します。

トピック

- パッケージ脆弱性
- コードの脆弱性
- ネットワーク到達可能性

パッケージ脆弱性

パッケージ脆弱性の検出結果は、共通脆弱性識別子 (CVEs) にさらされる AWS 環境内のソフトウェアパッケージを識別します。攻撃者は、こうしたパッチが適用されていない脆弱性を利用し、データの機密性、完全性、可用性を侵害したり、他のシステムにアクセスしたりする可能性があります。CVE システムは、セキュリティの脆弱性や露出についての既知の情報を参照する方法です。詳細については、https://www.cve.org/ を参照してください。

Amazon Inspector では、EC2 インスタンス、ECR コンテナイメージ、および Lambda 関数についてパッケージ脆弱性検出結果を生成できます。パッケージ脆弱性検出結果には、この検出結果タイプに固有の詳細が追加されており、インスペクタスコアと脆弱性インテリジェンスがこれに該当します。

コードの脆弱性

コード脆弱性の検出結果は、悪用される可能性のあるコード行を特定するのに役立ちます。コードの脆弱性には、暗号化の欠落、データリーク、インジェクションの欠陥、暗号化の弱さなどがあります。Amazon Inspector は、Lambda 関数のスキャンとそのコードセキュリティ機能を通じてコード脆弱性の検出結果を生成します。

Amazon Inspector は、自動推論と機械学習を使用して Lambda 関数アプリケーションコードを評価し、アプリケーションコードを分析して全体的なセキュリティコンプライアンスを確認します。Amazon CodeGuru と共同で開発された内部ディテクターに基づいて、ポリシー違反と脆弱性を識別します。可能な検出のリストについては、「CodeGuru Detector Library」を参照してください。

コードスキャンは、検出された脆弱性を強調するためにコードのスニペットをキャプチャします。たとえば、コードスニペットでは、ハードコードされた認証情報やその他の機密マテリアルがプレーンテキストで表示される場合があります。CodeGuru は、コードの脆弱性に関連するコードスニペットを保存します。デフォルトでは、コードは AWS 所有キーで暗号化されます。ただし、この情報をより詳細に制御したい場合は、カスタマーマネージドキーを作成してコードを暗号化できます。詳細については、「検出結果のコードの保管時の暗号化」を参照してください。

> **Note**
>
> 組織の委任管理者は、メンバーアカウントに属するコードスニペットを表示できません。

ネットワーク到達可能性

ネットワーク到達可能性の検出結果は、環境内の Amazon EC2 インスタンスへのネットワークパスが開いていることを示しています。インターネットゲートウェイ (Application Load Balancer または Classic Load Balancer の背後にあるインスタンスを含む)、VPC ピアリング接続、または仮想ゲートウェイを介した VPN など、VPC エッジから TCP および UDP ポートに到達可能な場合、これらの結果が表示されます。これらの結果では、セキュリティグループ、アクセス制御リスト、インターネットゲートウェイなどの管理が不適切であったり、潜在的に悪意のあるアクセスを許している可能性があるなど、過度に寛容なネットワーク設定をハイライトします。

Amazon Inspector は、Amazon EC2 インスタンスのネットワーク到達可能性の検出結果のみを生成します。Amazon Inspector は、Amazon Inspector を有効にすると、12 時間ごとにネットワーク到達可能性の検出結果のスキャンを実行します。

Amazon Inspector は、ネットワークパスをスキャンする際に以下の設定を評価します。

- Amazon EC2 インスタンス
- アプリケーション ロード バランサー
- Direct Connect
- 弾性ロードバランサ
- 弾性ネットワークインターフェース

- インターネットゲートウェイ
- ネットワークアクセスコントロールリスト
- ルートテーブル
- セキュリティグループ
- サブネット
- 仮想プライベートクラウド
- 仮想プライベートゲートウェイ
- VPC エンドポイント
- ゲートウェイ VPC エンドポイント
- VPC ピアリング接続
- VPN 接続

Amazon Inspector 検出結果の表示

Amazon Inspector の検出結果は、Amazon Inspector コンソールで表示できるほか、Amazon Inspector `ListFindings` API を使用して表示できます。Amazon Inspector コンソールでは、Amazon Inspector ダッシュボードと [検出結果] 画面で検出結果を表示できます。検出結果は、AWS Security Hub および Amazon Elastic Container Registry (Amazon ECR) で表示することもできます。デフォルトでは、Amazon Inspector ダッシュボードと [検出結果] 画面にアクティブな検出結果が表示されます。検出結果をカテゴリ別に表示することもできます。このセクションの手順では、Amazon Inspector コンソールおよび Amazon Inspector API で検出結果を表示する方法について説明します。

Console

Amazon Inspector の検出結果を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. (オプション) ナビゲーションペインから、[ダッシュボード] を選択します。ダッシュボードには、環境のカバレッジ概要と、緊急の検出結果のみが表示されます。
3. (オプション) ナビゲーションペインから、[検出結果] を選択します。[検出結果] 画面には、すべてのアクティブな検出結果がテーブルに表示され、ステータスとフィルター基準で検出結果をフィルタリングできます。また、抑制ルールを作成して、検出結果を表示から除外することもできます。検出結果の詳細を表示するには、検出結果の名前を選択します。

4. (オプション) ナビゲーションペインから、次のいずれかのオプションを選択して、カテゴリ別に検出結果を表示します。

   - 脆弱性別 – 最も緊急の脆弱性を示します。
   - アカウント別 – すべてのアカウントとスキャン範囲、および重要度評価が「緊急」と「高い」検出結果の合計数を表示します。

     > **Note**
     > このカテゴリは委任管理者のみが利用できます。

   - インスタンス別 – 最も脆弱な Amazon EC2 インスタンスを表示します。

     > **Note**
     > このカテゴリにグループ化された検出結果には、ネットワークの可用性に関する情報は含まれません。

   - コンテナイメージ別 – 最も脆弱な Amazon ECR コンテナイメージを表示します。
   - コンテナリポジトリ別 – 最も脆弱なリポジトリを表示します。
   - Lambda 関数別 – 最も脆弱な Lambda 関数を表示します。

API

Amazon Inspector の検出結果を表示するには

- ListFindings API オペレーションを実行します。リクエストでは、特定の検出結果を返すように filterCriteria を指定できます。

Amazon Inspector 検出結果の詳細の表示

このセクションの手順は、Amazon Inspector 検出結果の詳細を表示する方法を示しています。

検出結果の詳細を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. 検出結果を表示するリージョンを選択します。

3. ナビゲーションペインで、[検出結果] を選択して検出結果リストを表示する
4. (オプション) フィルターバーを使用して特定の検出結果を選択します。詳細については、「Amazon Inspector の検出結果のフィルタリング」を参照してください。
5. 検出結果を選択して、その詳細パネルを表示します。

[検出結果の詳細] パネルには、検索結果の基本的な識別機能が含まれています。これには、検出結果のタイトル、特定された脆弱性の基本的な説明、修復の提案、および重要度スコアが含まれます。スコアリングについては、「Amazon Inspector の検出結果の重要度レベルについて」を参照してください。

検出結果の詳細は、検出結果の種類と影響を受けるリソースによって異なります。

すべての検出結果には、検出結果が特定された AWS アカウント ID 番号、重要度、検出結果タイプ、検出結果が作成された日付、そのリソースに関する詳細を含むリソースの影響を受けるセクションが含まれます。

検出結果のタイプは、その検出結果に対して利用可能な修復と脆弱性インテリジェンス情報を決定します。検出結果タイプによって、利用可能な検出結果の詳細が異なります。

パッケージの脆弱性

EC2 インスタンス、ECR コンテナイメージ、Lambda 関数に関しては、パッケージの脆弱性の検出結果を利用できます。詳細については、「パッケージ脆弱性」を参照してください。

パッケージの脆弱性の検出結果には Amazon Inspector スコアの表示と脆弱性インテリジェンスの詳細の理解 も含まれます。

この検出結果タイプには以下の詳細があります。

- 修正可能 — 影響を受けるパッケージの新しいバージョンで脆弱性が修正されているかどうかを示します。次のいずれかの値があります。
  - `YES`、これは影響を受けるパッケージのすべてに修正されたバージョンがあることを意味します。
  - `NO`、これは影響を受けるパッケージに修正されたバージョンがないことを意味します。
  - `PARTIAL`、これは影響を受けるパッケージの 1 つ以上 (すべてではない) に修正されたバージョンがあることを意味します。
- 考えられる攻撃 — 脆弱性に既知の悪用があることを示します。
  - `YES`、これは環境内で発見された脆弱性には既知の悪用があるということです。Amazon Inspector では、環境における悪用の使用状況を可視化することはできません。

  - NO、これはこの脆弱性には既知の悪用がないことを意味します。
- 影響を受けるパッケージ — 検出結果で脆弱であると特定された各パッケージと、各パッケージの詳細を一覧表示します。
- ファイルパス — 検出結果に関連付けられた EBS ボリューム ID とパーティション番号です。このフィールドは、エージェントレススキャン を使用してスキャンされた EC2 インスタンスの検出結果に存在します。
- インストール済みバージョン/修正バージョン — 脆弱性が検出された、現在インストールされているパッケージのバージョン番号。インストールされているバージョン番号とスラッシュ (/) の後の値を比較します。2 番目の値は、検出された脆弱性を修正するパッケージのバージョン番号です。このバージョン番号は、共通脆弱性識別子 (CVE) または検出結果に関連するアドバイザリによって提供されています。脆弱性が複数のバージョンで修正されている場合、このフィールドには修正を含む最新バージョンが表示されます。修正がない場合、この値は `None available` です。

> **Note**
>
> Amazon Inspector がこのフィールドを検出結果に含み始める前に検出結果が検出された場合には、このフィールドの値は空となります。ただし、修正できる場合もあります。

- パッケージマネージャー — このパッケージの設定に使用されるパッケージマネージャー。
- 修復 — 更新されたパッケージまたはプログラミングライブラリから修正が入手できる場合、このセクションには更新を行うために実行できるコマンドが含まれています。提供されたコマンドをコピーして、ご使用の環境で実行できます。

> **Note**
>
> 修復コマンドはベンダーのデータフィードから提供され、システム設定によって異なる場合があります。より具体的なガイダンスについては、検出結果の参考資料またはオペレーティングシステムのマニュアルを参照してください。

- 脆弱性の詳細 — National Vulnerability Database (NVD)、REDHAT、または別の OS ベンダーなど、検出結果で特定された CVE の Amazon Inspector 優先ソースへのリンクを提供します。さらに、検出結果の重要度スコアも表示されます。重要度スコアリングの詳細については、「Amazon Inspector の検出結果の重要度レベルについて」を参照してください。それぞれのスコアリングベクトルを含め、以下のスコアが含まれています。

  - Exploit Prediction Scoring System (EPSS) スコア
  - Inspector スコア
  - Amazon CVE の CVSS 3.1
  - NVD の CVSS 3.1
  - NVD の CVSS 2.0 (該当する場合、過去の CVE について)
- 関連する脆弱性 — 検出結果に関連する他の脆弱性を指定します。通常、これらは同じパッケージバージョンに影響する他の CVE、または検出結果の CVE と同じグループ内のベンダーが判断した他の CVE です。

コードの脆弱性

コード脆弱性の検出結果は Lambda 関数でのみ確認できます。詳細については、「コードの脆弱性」を参照してください。この検出結果タイプには以下の詳細があります。

- 修正可能 — コード脆弱性の場合、この値は常に YES です。
- ディテクター名 — コードの脆弱性を検出するために使用される CodeGuru ディテクターの名前。可能な検出のリストについては、Q Detector Library を参照してください。
- ディテクタータグ — ディテクターに関連付けられた CodeGuru タグ。CodeGuru はタグを使用して検出を分類します。
- 関連する CWE — コードの脆弱性に関連する共通脆弱性タイプ (CWE) の ID。
- ファイルパス — コード脆弱性のファイルの場所。
- 脆弱性の場所 — Lambda コードスキャンコードの脆弱性の場合、このフィールドには Amazon Inspector が脆弱性を検出した正確なコード行が表示されます。
- 推奨される是正 — 検出結果を修正するためにコードを編集する方法を提示します。

ネットワーク到達可能性

ネットワーク到達可能性の検出結果は EC2 インスタンスでのみ確認できます。詳細については、「ネットワーク到達可能性」を参照してください。この検出結果タイプには以下の詳細があります。

- ポート範囲を開く — EC2 インスタンスにアクセスできるポート範囲。
- ネットワークパスを開く — EC2 インスタンスへのオープンアクセスパスを表示します。パス上の項目を選択すると、詳細が表示されます。
- 修復 — 開いているネットワークパスを閉鎖する方法を推奨します。

Amazon Inspector スコアの表示と脆弱性インテリジェンスの詳細の理解

Amazon Inspector は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの検出結果のスコアを作成します。Amazon Inspector コンソールで Amazon Inspector スコアと脆弱性インテリジェンスの詳細を表示できます。Amazon Inspector スコアには、共通脆弱性評価システムのメトリクスと比較できる詳細情報が表示されます。これらの詳細は、パッケージの脆弱性の検出結果にのみ表示されます。このセクションでは、Amazon Inspector スコアを解釈し、脆弱性インテリジェンスの詳細を理解する方法について説明します。

Amazon Inspector スコア

Amazon Inspector スコアは、Amazon Inspector が各 EC2 インスタンス検出結果ごとに作成される、状況に応じたスコアです。Amazon Inspector スコアは、CVSS v3.1 の基本スコア情報を、ネットワーク到達可能性の結果や悪用可能性データなど、スキャン中にコンピューティング環境から収集された情報と関連付けることによって決定されます。たとえば、脆弱性がネットワーク上で悪用可能であるのに、Amazon Inspector が脆弱なインスタンスへのオープンネットワークパスがインターネットから利用できないと判断した場合、検出結果の Amazon Inspector スコアは基本スコアよりも低くなる可能性があります。

検出結果の基本スコアは、ベンダーが提供する CVSS v3.1 の基本スコアです。RHEL、Debian、または Amazon ベンダーベースのスコアがサポートされています。その他のベンダー、またはベンダーがスコアを提供していない場合、Amazon Inspector は National Vulnerability Database (NVD) の基本スコアを使用します。Amazon Inspector は、共通脆弱性評価システムバージョン 3.1 の計算機を使用してスコアを計算します。個々の検出結果の基本スコアのソースは、[脆弱性の詳細] の下にある検出結果の詳細で、[脆弱性ソース] (または検出結果の JSON では packageVulnerabilityDetails.source) として確認できます。

> **Note**
>
> Amazon Inspector スコアは、Ubuntu を実行している Linux インスタンスでは使用できません。これは、Ubuntu が独自の脆弱性重要度を定義しており、関連する CVE の重要度とは異なる可能性があるためです。

Amazon Inspector スコアの詳細

検出結果の詳細ページを開くと、[Inspector スコアと脆弱性インテリジェンス] タブを選択できます。このパネルには、ベーススコアと [Inspector スコア] の差が表示されます。このセクションでは、Amazon Inspector がソフトウェアパッケージの Amazon Inspector スコアとベンダースコアの組み合わせに基づいて重要度評価を割り当てる方法について説明します。スコアが異なる場合は、このパネルに理由の説明が表示されます。

[CVSS スコアメトリクス] セクションには、CVSS ベーススコアメトリクスと [Inspector スコア] の比較表があります。比較されるメトリクスは、first.org が管理する CVSS 仕様書で定義されている基本メトリクスです。基本メトリクスの概要を以下に示します。

攻撃ベクトル

脆弱性が悪用される可能性がある状況。Amazon Inspector の検出結果では、[ネットワーク]、[隣接ネットワーク]、または [ローカル] を選択できます。

攻撃の複雑さ

これは、攻撃者が脆弱性を悪用したときに直面する難易度を示しています。スコアが低いということは、攻撃者が脆弱性を悪用するために必要な追加条件がほとんどないか全くないことを意味します。高スコアは、攻撃者がこの脆弱性を利用した攻撃を成功させるために多大な労力を費やす必要があることを意味します。

必要な権限

これは、攻撃者が脆弱性を悪用するのに必要な権限レベルを表します。

ユーザーインタラクション

このメトリクスは、この脆弱性を利用した攻撃を成功させるには、攻撃者以外の人間のユーザーが必要かどうかを示します。

スコープ

これは、ある脆弱なコンポーネントの脆弱性が、その脆弱なコンポーネントのセキュリティ範囲外のコンポーネントのリソースに影響を与えるかどうかを示します。この値が [未変更] の場合、影響を受けるリソースと影響を受けるリソースは同じです。この値が「変更済」の場合、脆弱なコンポーネントを悪用して、異なるセキュリティ機関が管理するリソースに影響を与える可能性があります。

機密性

脆弱性が悪用された場合に、リソース内のデータの機密性がどの程度影響を受けるかを測定します。その範囲は、機密性が失われない「なし」から、リソース内のすべての情報が漏洩したり、パスワードや暗号化キーなどの機密情報が漏洩したりする可能性のある「高」までさまざまです。

整合性

これは、脆弱性が悪用された場合に、影響を受けるリソース内のデータの完全性がどの程度影響を受けるかを測定します。攻撃者が影響を受けるリソース内のファイルを変更すると、完全性が危険にさらされます。スコアの範囲は、攻撃者がどの情報も変更することを許可しない「なし」から、脆弱性が悪用された場合、攻撃者がいずれかまたはすべてのファイルを変更することができる「高」まであります。

現在利用できるリージョン

これは、脆弱性が悪用された場合に、影響を受けるリソースの可用性に与える影響の度合いを測定します。スコアの範囲は、脆弱性が可用性にまったく影響を与えない場合の「なし」から、悪用された場合、攻撃者のリソースの利用を完全に拒否したり、サービスを利用できなくしたりすることができる「高」まであります。

脆弱性インテリジェンス

このセクションでは、Amazon が提供する CVE に関する情報のほか、Recorded Future や Cybersecurity and Infrastructure Security Agency (CISA) などの業界標準のセキュリティインテリジェンスソースについてもまとめています。

> **Note**
>
> CISA、Amazon、または Recorded Future のインテリジェンスは、すべての CVE で利用できるわけではありません。

脆弱性インテリジェンスの詳細は、コンソールまたは BatchGetFindingDetails API を使用して表示できます。以下の詳細が、コンソールで使用可能です。

ATT&CK

このセクションでは、CVE に関連する MITRE の戦術、技法、手順 (TTP) について説明します。関連する TTP が表示されます。該当する TTP が 3 つ以上ある場合は、リンクを選択すると詳細

なリストが表示されます。戦術や技法を選択すると、MITRE のウェブサイトにその情報が表示されます。

CISA

このセクションでは、脆弱性に関連する日付について説明します。Cybersecurity and Infrastructure Security Agency (CISA) が、活発な悪用の証拠に基づいてその脆弱性を「Known Exploited Vulnerabilities Catalog」(悪用された既知の脆弱性カタログ) に追加した日付と、CISA がシステムにパッチを適用する期限。この情報は CISA から提供されています。

既知のマルウェア

このセクションでは、この脆弱性を悪用する既知のエクスプロイトキットとツールを一覧表示します。

証拠

このセクションでは、この脆弱性に関係する最も緊急のセキュリティイベントを要約します。緊急度が同じイベントが 3 つ以上ある場合は、最新のイベントの上位 3 つが表示されます。

最終レポート時刻

このセクションには、この脆弱性が一般に悪用されたことが判明した最終日が表示されます。

Amazon Inspector の検出結果の重要度レベルについて

Amazon Inspector が検出結果を生成すると、その検出結果に重要度の評価が割り当てられます。重要度の評価は、検出結果の評価と優先順位付けに役立ちます。検出結果の重要度評価は、数値スコアと、[参考]、[低]、[中]、[高]、[緊急] というレベルに対応しています。Amazon Inspector は、検出結果タイプに基づいて検出結果の重要度の評価を決定します。このセクションでは、Amazon Inspector が各検出結果タイプの重要度の評価を決定する方法について説明します。

ソフトウェアパッケージの脆弱性の重要度

Amazon Inspector は、ソフトウェアパッケージの脆弱性の重要度スコアの基礎として NVD/CVSS スコアを使用します。NVD/CVSS スコアは、NVD によって公開され、CVSS によって定義される脆弱性重要度スコアです。NVD/CVSS スコアは、攻撃の複雑さ、悪用コードの成熟度、必要な権限などのセキュリティメトリクスで構成されています。Amazon Inspector は、脆弱性の重要度を反映した 1 から 10 までの数値スコアを生成します。Amazon Inspector では、これを基本スコアとして分類しています。これは、脆弱性の重要度がその固有の特性に従って反映され、時間が経過しても変化しな

いためです。このスコアは、デプロイされたさまざまな環境における最悪の場合の妥当な影響も想定しています。CVSS v3 標準では、CVSS スコアを以下の重要度評価にマッピングしています。

スコア	Rating
0	Informational
0.1–3.9	Low
4.0–6.9	Medium
7.0–8.9	High
9.0–10.0	Critical

パッケージ脆弱性の検出結果は、重要度が「未選別」である場合もあります。つまり、ベンダーは検出された脆弱性の脆弱性スコアをまだ設定していないということです。この場合、検出結果の参照 URL を使用して脆弱性を調査し、それに応じて対応することをおすすめします。

パッケージ脆弱性の検出結果には、検出結果の詳細の一部として、以下のスコアと関連するスコアリングベクトルが含まれます。

- EPSS スコア
- Inspector スコア
- Amazon CVE の CVSS 3.1
- NVD の CVSS 3.1
- NVD の CVSS 2.0 (該当する場合)

コード脆弱性の重要度

コードの脆弱性検出結果では、Amazon Inspector は検出結果を生成した Amazon CodeGuru ディテクターによって定義された重要度レベルを使用します。各ディテクターには CVSS v3 スコアリングシステムを使用して重要度が割り当てられます。CodeGuru が使用する重要度の説明については、「CodeGuru ガイド」の「重要度の定義」を参照してください。重要度別のディテクターのリストについては、以下のサポートされているプログラミング言語から選択してください。

- 重要度別 Python ディテクター

- 重要度別 Java ディテクター

ネットワーク到達可能性の重要度

Amazon Inspector は、公開されているサービス、ポート、プロトコル、およびオープンパスのタイプに基づいて、ネットワーク到達可能性の脆弱性の重要度を判断します。次の表では、これらの重要度評価を定義しています。Open path rating 列の値は、仮想ゲートウェイ、ピア接続された VPCs、AWS Direct Connect ネットワークからのオープンパスを表します。公開されている他のすべてのサービス、ポート、プロトコルには「情報重要度」という評価があります。

サービス	TCP ポート	UDP ポート	インターネットパス評価	オープンパス評価
DHCP	67, 68, 546, 547	67, 68, 546, 547	Medium	Informational
Elasticsearch	9300, 9200	NA	Medium	Informational
FTP	21	21	High	Medium
Global catalog LDAP	3268	NA	Medium	Informational
Global catalog LDAP over TLS	3269	NA	Medium	Informational
HTTP	80	80	Low	Informational
HTTPS	443	443	Low	Informational
Kerberos	88, 464, 543, 544, 749, 751	88, 464, 749, 750, 751, 752	Medium	Informational
LDAP	389	389	Medium	Informational
LDAP over TLS	636	NA	Medium	Informational
MongoDB	27017, 27018, 27019, 28017	NA	Medium	Informational
MySQL	3306	NA	Medium	Informational

NetBIOS	137, 139	137, 138	Medium	Informational
NFS	111, 2049, 4045, 1110	111, 2049, 4045, 1110	Medium	Informational
Oracle	1521, 1630	NA	Medium	Informational
PostgreSQL	5432	NA	Medium	Informational
Print services	515	NA	High	Medium
RDP	3389	3389	Medium	Low
RPC	111, 135, 530	111, 135, 530	Medium	Informational
SMB	445	445	Medium	Informational
SSH	22	22	Medium	Low
SQL Server	1433	1434	Medium	Informational
Syslog	601	514	Medium	Informational
Telnet	23	23	High	Medium
WINS	1512, 42	1512, 42	Medium	Informational

Amazon Inspector での検出結果の管理

Amazon Inspector を使用すると、検出結果をさまざまな方法で管理できます。検出結果は、そのステータスに基づいてフィルタリングできます。検出結果は、フィルター条件に基づいて検索できます。抑制ルールを作成して、検出結果のリストから検出結果を除外できます。検出結果を AWS Security Hub、Amazon EventBridge、Amazon Simple Storage Service (Amazon S3) にエクスポートすることもできます。

トピック

- Amazon Inspector の検出結果のフィルタリング
- Amazon Inspector 検出結果の抑制
- Amazon Inspector 検出結果レポートのエクスポート
- Amazon EventBridge を使用して Amazon Inspector の検出結果に対するカスタムレスポンスを作成する

Amazon Inspector の検出結果のフィルタリング

Amazon Inspector の検出結果は、フィルター条件を使用してフィルタリングできます。検出結果がフィルター条件と一致しない場合、Amazon Inspector は検出結果をビューから除外します。このセクションでは、フィルター条件を使用して Amazon Inspector 検出結果をフィルタリングする方法について説明します。

Amazon Inspector コンソールでフィルターを作成

各検出結果ビューでは、フィルター機能を使用して特定の特性を持つ検出結果を検索できます。別のタブ付きビューに移動すると、フィルターは削除されます。

フィルタは、フィルタ属性とフィルタ値からなるフィルタ基準で構成されます。フィルター条件に一致しない検出結果は検出結果リストから除外されます。例えば、管理者アカウントに関連付けられているすべての検出結果を表示するには、 AWS アカウント ID 属性を選択し、12 桁の AWS アカウント ID の値とペアリングします。

すべての検出結果に適用されるフィルター条件もあれば、特定のリソースタイプや検出結果タイプのみに適用されるフィルター条件もあります。

検出結果ビューにフィルターを適用するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインで 調査結果を選択します。デフォルトビューでは、「アクティブ」ステータスのすべての検出結果が表示されます。
3. 検索結果を条件でフィルタリングするには、[フィルタを追加] バーを選択すると、そのビューに適用可能なすべてのフィルタ条件のリストが表示されます。さまざまなビューで、さまざまなフィルター条件を使用できます。
4. フィルターに使う条件をリストから選択します。
5. 条件入力ペインから、その条件を定義するために必要なフィルター値を入力します。
6. [適用] を選択して、そのフィルター条件を現在の結果に適用します。フィルター入力バーをもう一度選択すると、他のフィルター条件を引き続き追加できます。
7. (オプション) 抑制された検出結果または終了した結果を表示するには、フィルターバーの [アクティブ] を選択し、次に [抑制] または [終了] を選択します。[すべて表示] を選択すると、アクティブな結果、抑制された結果、終了した結果が同じビューに表示されます。

Amazon Inspector 検出結果の抑制

抑制ルールを作成して、条件に一致する検出結果を非表示にできます。例えば、抑制ルールを作成し、重要度評価に基づいて検出結果を非表示にできます。Amazon Inspector が抑制ルールに一致する検出結果を生成する場合、Amazon Inspector は検出結果を抑制し、非表示にします。Amazon Inspector では、抑制された検出結果は修復されるまで保存されます。抑制された検出結果が修正されると、Amazon Inspector は検出結果を終了します。抑制された検出結果は、コンソールで表示できます。

最も重要な検出結果を優先するための抑制ルールを作成します。抑制ルールは検出結果を非表示にするだけであり、検出結果には影響しません。検出結果を終了または修正する抑制ルールを作成することはできません。Amazon EventBridge ルール AWS Security Hub を使用して、 で不要な検出結果を抑制することもできます。このセクションの手順は、抑制ルールを作成、表示、編集、削除する方法を示しています。

> **Note**
>
> 組織の委任管理者のみが抑制ルールを作成および管理できます。

抑制ルールを作成する

抑制ルールを作成して、デフォルトで表示される検出結果のリストを絞り込むことができます。CreateFilter API を使用し、`action` の値として `SUPRESS` を指定することで、抑制ルールをプログラムで作成できます。

> **Note**
>
> 抑制ルールを作成および管理できるのは、スタンドアロンアカウントと Amazon Inspector の委任管理者のみです。組織のメンバーには、ナビゲーションペインに抑制ルールのオプションが表示されません。

抑制ルールを作成するには (コンソール)

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインで、[抑制ルール] を選択します。次に、[Create rule (ルールを作成)] を選択します。
3. 各条件について、以下を実行します。
   - フィルターバーを選択すると、抑制ルールに追加できるフィルター条件のリストが表示されます。
   - 抑制ルールのフィルター条件を選択します。
4. 条件を追加し終えたら、ルールの名前と、必要に応じて説明を入力します。
5. [ルールを保存] を選択します。Amazon Inspector は、新しい抑制ルールを直ちに適用し、条件に一致する結果はすべて非表示にします。

抑制された検出結果を表示する

デフォルトでは、Amazon Inspector は抑制された検出結果を Amazon Inspector コンソールに表示しません。ただし、特定のルールによって抑制された結果は表示できます。

抑制された検出結果を表示するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。

2. ナビゲーションペインで、[抑制ルール] を選択します。
3. 抑制ルールリストで、ルールのタイトルを選択します。

抑制ルールの編集

抑制ルールはいつでも変更ができます。

抑制ルールを変更するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、抑制ルールを選択します。
3. 変更する抑制ルールの名前を選択し、編集を選択します。
4. 意図した変更を行い、保存を選択します。

抑制ルールを削除する

抑制ルールは削除できます。抑制ルールを削除すると、Amazon Inspector は、ルールの基準を満たし、他のルールによって抑制されていない、新規および既存の結果の抑制を停止します。

抑制ルールを削除したすると、ルールの基準を満たした検出結果の新規および現在の発生は、ステータスが [アクティブ] になります。これは、それらが Amazon Inspector コンソールにデフォルトで表示されることを意味します。さらに、Amazon Inspector はこれらの結果を AWS Security Hub と Amazon EventBridge にイベントとして発行します。

抑制ルールを削除するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインで、[抑制ルール] を選択します。
3. 削除する抑制ルールのタイトルの横にあるチェックボックスをオンにします。
4. [削除] を選択し、その選択を確定してルールを完全に削除します。

Amazon Inspector 検出結果レポートのエクスポート

検出結果レポートは、検出結果の詳細なスナップショットを提供する CSV または JSON ファイルです。検出結果レポートは AWS Security Hub、、Amazon EventBridge、および Amazon Simple Storage Service (Amazon S3) にエクスポートできます。検出結果レポートを設定するときは、どの検出結果をレポートに含めるかを指定します。デフォルトでは、検出結果レポートには、すべてのアクティブな検出結果のデータが含まれます。組織の委任管理者である場合、検出結果レポートには、組織内のすべてのメンバーアカウントの検出結果データが含まれます。検出結果レポートをカスタマイズするには、フィルターを作成して適用します。

検出結果レポートをエクスポートすると、Amazon Inspector は AWS KMS key 指定した を使用して検出結果データを暗号化します。Amazon Inspector が検出結果を暗号化すると、検出結果は指定された Amazon S3 バケットに保存されます。 AWS KMS キーは、Amazon S3 バケット AWS リージョン と同じ で使用する必要があります。 AWS KMS キーポリシーでは、Amazon Inspector による使用を許可し、Amazon S3 バケットポリシーではAmazon Inspector によるオブジェクトの追加を許可する必要があります。検出結果レポートをエクスポートしたら、Amazon S3 バケットからダウンロードするか、新しい場所に転送できます。Amazon S3 バケットを、エクスポートされた他の検出結果レポートのリポジトリとして使用することもできます。

このセクションでは、Amazon Inspector コンソールで検出結果レポートをエクスポートする方法について説明します。以下のタスクでは、アクセス許可の検証、Amazon S3 バケットの設定、 の設定 AWS KMS key、検出結果レポートの設定とエクスポートを行う必要があります。

> **Note**
>
> Amazon Inspector CreateFindingsReport API を使用して検出結果レポートをエクスポートする場合、アクティブな検出結果の表示のみができます。抑制された検出結果または終了した検出結果を表示するには、フィルター条件の一部として SUPPRESSED または CLOSED を指定する必要があります。

タスク

- ステップ 1: アクセス許可を確認する
- ステップ 2: S3 バケットを設定する
- ステップ 3: AWS KMS keyを設定する
- ステップ 4: 調査結果レポートを設定しエクスポートする

- エクスポートエラーのトラブルシューティング

ステップ 1: アクセス許可を確認する

> **Note**
>
> 検出結果レポートの初回エクスポート後、ステップ 1 ~ 3 はオプションになります。これらのステップは、同じ Amazon S3 バケットと、エクスポートされた AWS KMS key 他の検出結果レポートのどちらを使用するかに基づいています。ステップ 1 ~ 3 の完了後に検出結果レポートをプログラムでエクスポートする場合は、Amazon Inspector API の CreateFindingsReport オペレーションを使用してください。

Amazon Inspector から調査結果レポートをエクスポートする前に、調査結果レポートのエクスポートと、レポートの暗号化と保存のためのリソースの設定の両方に必要なアクセス許可があることを確認してください。アクセス許可を確認するには、 AWS Identity and Access Management (IAM) を使用して、IAM ID にアタッチされている IAM ポリシーを確認します。次にこれらのポリシー内の情報を、調査結果レポートをエクスポートするために実行を許可されなければならない以下のアクションのリストと比較します。

Amazon Inspector

Amazon Inspector の場合、次のアクションの実行が許可されていることを確認します。

- `inspector2:ListFindings`
- `inspector2:CreateFindingsReport`

これらのアクションにより、アカウントの検出結果データを取得し、そのデータを調査結果レポートにエクスポートできます。

サイズの大きいレポートをプログラムでエクスポートする予定の場合は、レポートのステータスを確認する、`inspector2:CancelFindingsReport`、進行中のエクスポートをキャンセルする操作が許可されていることも確認してください。`inspector2:GetFindingsReportStatus`

AWS KMS

では AWS KMS、次のアクションを実行できることを確認します。

- `kms:GetKeyPolicy`
- `kms:PutKeyPolicy`

これらのアクションにより、Amazon Inspector にレポートの暗号化に使用させたい AWS KMS key のキーポリシーを取得して更新できます。

Amazon Inspector コンソールを使用してレポートをエクスポートするには、次の AWS KMS アクションを実行できることも確認します。

- `kms:DescribeKey`
- `kms:ListAliases`

これらのアクションにより、アカウントの AWS KMS keys に関する情報を取得して表示することが許可されます。その後、これらのキーのいずれかを選択してレポートを暗号化できます。

レポートを暗号化するために新しい KMS キーを作成することを計画している場合、`kms:CreateKey` アクションの実行も許可される必要があります。

Amazon S3

Amazon S3 の場合、次のアクションの実行が許可されていることを確認します。

- `s3:CreateBucket`
- `s3:DeleteObject`
- `s3:PutBucketAcl`
- `s3:PutBucketPolicy`
- `s3:PutBucketPublicAccessBlock`
- `s3:PutObject`
- `s3:PutObjectAcl`

これらのアクションにより、Amazon Inspector でレポートを保存する S3 バケットを作成して設定できます。また、バケットにオブジェクトを追加したり、バケットからオブジェクトを削除したりすることもできます。

Amazon Inspector コンソールを使用してレポートをエクスポートする予定がある場合は、`s3:ListAllMyBuckets` および `s3:GetBucketLocation` アクションの実行が許可されていることも確認してください。これらのアクションにより、アカウントの S3 バケットに関する情報を取得して表示することができます。その後、レポートを保存するバケットを 1 つ選択できます。

必要なアクションの 1 つ以上を実行できない場合は、次のステップに進む前に、 AWS 管理者にサポートを依頼してください。

ステップ 2: S3 バケットを設定する

アクセス許可を確認したら、調査結果レポートを保存する S3 バケットを設定します。自分のアカウントの既存のバケットでも、別の が所有する既存のバケット AWS アカウント でもアクセスが許可されます。レポートを新しいバケットに保存する場合は、先に進む前にバケットを作成してください。

S3 バケットは、エクスポートする検出結果データ AWS リージョン と同じ にある必要があります。例えば、Amazon Inspector を米国東部 (バージニア北部) リージョンで使用していて、そのリージョンの検出結果データをエクスポートする場合、バケットも米国東部 (バージニア北部) リージョンにある必要があります。

さらに、バケットのポリシーでは、Amazon Inspector がバケットにオブジェクトを追加することを許可する必要があります。このトピックでは、バケットポリシーを更新する方法について説明し、ポリシーに追加するステートメントの例も示します。バケットポリシーの追加と更新の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットポリシーの使用」を参照してください。

別のアカウントが所有する S3 バケットにレポートを保存する場合は、バケットの所有者と連携してバケットのポリシーを更新します。また、バケットの URI も取得します。レポートをエクスポートするときに、この URI を入力する必要があります。

バケットポリシーを更新するには

1. 認証情報を使用してサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3) を開きます。
2. ナビゲーションペインで、バケットを選択します。
3. 調査結果レポートを保存する S3 バケットを選択します。
4. アクセス許可 タブを選択します。
5. バケットポリシー セクションで、編集 を選択します。
6. 次のステータス例をクリップボードにコピーします。

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

```
    "Principal": {
     "Service": "inspector2.amazonaws.com"
    },
    "Action": [
     "s3:PutObject",
     "s3:PutObjectAcl",
     "s3:AbortMultipartUpload"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "111122223333"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:inspector2:Region:111122223333:report/*"
     }
    }
   }
  ]
}
```

7. Amazon S3 コンソールのバケットポリシーエディタで、前のステートメントをポリシーに貼り付けてポリシーに追加します。

   ステートメントをポリシーに追加するときに、構文が有効であることを確認します。バケットポリシーは JSON 形式を使用します。これは、ステートメントをポリシーに追加する場所に応じて、ステートメントの前後にカンマを追加する必要があることを意味します。ステートメントを最後のステートメントとして追加する場合は、前のステートメントの右中括弧の後にカンマを追加します。最初のステートメントとして追加するか、既存の 2 つのステートメントの間に追加する場合は、右中括弧の後にカンマを追加します。

8. 環境に合った正しい値でステートメントを更新します。

   - *amzn-s3-demo-bucket* はバケットの名前です。
   - *111122223333* は、お客様の AWS アカウントのアカウント ID です。
   - *#####*は、Amazon Inspector を使用していて、Amazon Inspector にバケットへのレポートの追加を許可する AWS リージョン です。たとえば、米国東部 (バージニア北部) リージョンの場合は `us-east-1` です。

> **Note**
>
> 手動で有効にした で Amazon Inspector を使用している場合は AWS リージョン、`Service`フィールドの値に適切なリージョンコードも追加します。このフィールドは Amazon Inspector サービスプリンシパルを指定します。
> たとえば、リージョンコード `me-south-1` が設定されている中東 (バーレーン) リージョンで Amazon Inspector を使用している場合は、ステートメントで `inspector2.amazonaws.com` を `inspector2.me-south-1.amazonaws.com` に置き換えます。

これらのステートメント例は、2 つの IAM グローバル条件キーを使用する条件を定義していることにご注意してください。

- [aws:SourceAccount] — この条件により、Amazon Inspector はアカウントのレポートのみをバケットに追加することができます。これにより、Amazon Inspector が他のアカウントのバケットにレポートを追加できなくなります。具体的には、条件は、`aws:SourceArn` 条件で指定されたリソースおよびアクションに対して、バケットを使用できるアカウントを指定します。

  バケット内の追加アカウントのレポートを保存するには、追加のアカウントごとにアカウント ID をこの条件に追加します。以下に例を示します。

  ```
  "aws:SourceAccount": [111122223333,444455556666,123456789012]
  ```

- aws:SourceArn — この条件により、バケットに追加されているオブジェクトのソースに基づいて、バケットへのアクセスを制限します。これにより、他の AWS のサービス がバケットにオブジェクトを追加できなくなります。また、Amazon Inspector がお客様のアカウントで他のアクションを実行中にオブジェクトをバケットに追加するのを防ぐこともできます 具体的には、Amazon Inspectorは、オブジェクトが調査結果レポートであり、かつ、それらのレポートがアカウントによって作成され、かつ、条件で指定されたリージョンにある場合にのみ、オブジェクトをバケットに追加することができます。

  Amazon Inspector が追加のアカウントで指定したアクションを実行することを許可するには、追加のアカウントごとに Amazon リソースネーム (ARN) をこの条件に追加します。以下に例を示します。

```
"aws:SourceArn": [
    "arn:aws:inspector2:Region:111122223333:report/*",
    "arn:aws:inspector2:Region:444455556666:report/*",
    "arn:aws:inspector2:Region:123456789012:report/*"
]
```

`aws:SourceAccount` と `aws:SourceArn` の条件によって指定されたアカウントは、一致する必要があります。

どちらの条件も、Amazon Inspector が Amazon S3 とのトランザクション中に [混乱した代理] として使用されるのを防ぐのに役立ちます。お勧めしませんが、バケットポリシーからこれらの条件を削除できます。

9. バケットポリシーの更新が完了したら、変更を保存する を選択します。

ステップ 3: AWS KMS keyを設定する

アクセス許可を確認して S3 バケットを設定したら、Amazon Inspector で調査結果レポートを暗号化するためにどの AWS KMS key を使用するかを決定します。キーは、カスタマーマネージドキーで、対称暗号化 KMS キーである必要があります。さらに、キーは、レポートを保存するように設定した S3 バケット AWS リージョン と同じ にある必要があります。

キーは、ご自分のアカウントの既存の KMS キーでも、別のアカウントが所有する既存の KMS キーでもかまいません。新しい KMS キーを使用する場合は、先に進む前にキーを作成します。別のアカウントが所有する既存のキーを使用したい場合は、そのキーの Amazon リソースネーム (ARN) を取得します。Amazon Inspector からレポートをエクスポートするときに、この ARN を入力する必要があります。KMS キー設定の作成と確認については、「AWS Key Management Service デベロッパーガイド」の「キーの管理」を参照してください。

使用する KMS キーを決定した後、Amazon Inspector にそのキーを使用するアクセス許可を付与します。そうしないと、Amazon Inspector がレポートを暗号化しエクスポートすることができません。Amazon Inspector にキーを使用するアクセス許可を付与するには、キーのキーポリシーを更新します。キーポリシーと KMS キーへのアクセス管理の詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMSのキーポリシー」を参照してください。

> **Note**
>
> 以下の手順は、Amazon Inspector が既存のキーを使用できるように更新するためのものです。既存のキーがない場合は、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

キーポリシーを更新するには

1. 認証情報を使用してサインインし、 AWS KMS コンソールを https://console.aws.amazon.com/kms://www.com で開きます。
2. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
3. レポートの暗号化に使用する KMS キーを選択します。キーは対称暗号化 (SYMMETRIC_DEFAULT) キーである必要があります。
4. アクセスポリシー タブで 編集 を選択します。[編集] ボタンのあるキーポリシーが表示されない場合は、まず [ポリシービューへの切り替え] を選択する必要があります。
5. 次のステートメント例をクリップボードにコピーします。

```
{
    "Sid": "Allow Amazon Inspector to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": "inspector2.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:inspector2:Region:111122223333:report/*"
        }
    }
}
```

6. AWS KMS コンソールのキーポリシーエディタで、前述のステートメントをキーポリシーに貼り付けてポリシーに追加します。

   ステートメントをポリシーに追加するときに、構文が有効であることを確認します。キーポリシーは JSON 形式を使用します。これは、ステートメントをポリシーに追加する場所に応じて、ステートメントの前後にカンマを追加する必要があることを意味します。ステートメントを最後のステートメントとして追加する場合は、前のステートメントの右中括弧の後にカンマを追加します。最初のステートメントとして追加するか、既存の 2 つのステートメントの間に追加する場合は、右中括弧の後にカンマを追加します。

7. 環境に合った正しい値でステートメントを更新します。

   - *111122223333* は、お客様の AWS アカウントのアカウント ID です。
   - *#####*は、Amazon Inspector AWS リージョン が キーを使用してレポートを暗号化できるようにする です。たとえば、米国東部 (バージニア北部) リージョンの場合は `us-east-1` です。

   > **Note**
   >
   > 手動で有効にした で Amazon Inspector を使用している場合は AWS リージョン、`Service`フィールドの値に適切なリージョンコードも追加します。たとえば、リージョンコードが設定されている中東 (バーレーン) リージョンで Amazon Inspector を使用している場合は、`inspector2.amazonaws.com` を `inspector2.me-south-1.amazonaws.com` に置き換えます。

   前のステップのバケットポリシーのサンプルステートメントと同様に、この例の `Condition` フィールドでは 2 つの IAM グローバル条件キーを使用しています。

   - [aws:SourceAccount] — この条件により、Amazon Inspector がお客様のアカウントに対してのみ指定されたアクションを実行することを許可します。具体的には、`aws:SourceArn` 条件で指定されたリソースおよびアクションに対して、指定されたアクションを実行できるアカウントを決定します。

     Amazon Inspector が追加のアカウントに対して指定されたアクションを実行することを許可するには、追加の各アカウントのアカウント ID をこの条件に追加します。以下に例を示します。

```
"aws:SourceAccount": [111122223333,444455556666,123456789012]
```

- aws:SourceArn — この条件により、他の AWS のサービス が指定されたアクションを実行するのを防ぎます。また、Amazon Inspector がお客様のアカウントで他のアクションを実行中にキーを使用するのを防ぐこともできます。つまり、Amazon Inspector は、オブジェクトが調査結果レポートであり、それらのレポートがアカウントによって作成され、条件で指定されたリージョンにある場合にのみ、S3 オブジェクトをキーで暗号化することができます。

  Amazon Inspector が追加のアカウントで指定したアクションを実行することを許可するには、追加のアカウントごとに ARN をこの条件に追加します。以下に例を示します。

  ```
  "aws:SourceArn": [
      "arn:aws:inspector2:us-east-1:111122223333:report/*",
      "arn:aws:inspector2:us-east-1:444455556666:report/*",
      "arn:aws:inspector2:us-east-1:123456789012:report/*"
  ]
  ```

  `aws:SourceAccount` と `aws:SourceArn` の条件によって指定されたアカウントは、一致する必要があります。

これらの条件は、 トランザクション中に Amazon Inspector が混乱した代理として使用されるのを防ぐのに役立ちます AWS KMS。お勧めしませんが、ステートメントからこれらの条件を削除できます。

8. キーポリシーの更新が完了したら、[変更を保存する] を選択します。

ステップ 4: 調査結果レポートを設定しエクスポートする

> **Note**
>
> 検出結果レポートは一度に 1 つしかエクスポートできません。エクスポートが進行中の場合は、エクスポートが完了するまで待ってから別の検出結果レポートをエクスポートする必要があります。

アクセス許可を確認し、調査結果レポートを暗号化して保存するリソースを設定したら、レポートを設定してエクスポートします。

調査結果レポートの設定とエクスポートをするには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインで [検出結果] の [すべての検出結果] を選択します。
3. (オプション) 検出結果 テーブルの上にあるフィルターバーを使用して、レポートに含める検出結果を指定するフィルター条件を追加します。条件を追加すると、Amazon Inspector は条件に一致する検出結果のみを含むようにテーブルを更新します。このテーブルには、レポートに含まれるデータのプレビューが表示されます。

   > **Note**
   >
   > フィルター条件を追加することをお勧めします。そうしないと、レポートには、ステータスがアクティブ AWS リージョン である現在の のすべての検出結果のデータが含まれます。お客様が組織の Amazon Inspector 管理者である場合、これには、組織内のすべてのメンバーアカウントの検出結果データが含まれます。
   > レポートにすべてまたは多数の検出結果データが含まれている場合、レポートの生成とエクスポートには時間がかかり、エクスポートは一度に 1 つのレポートしか生成できません。

4. [検出結果をエクスポート] を選択します。
5. [エクスポート設定] セクションの [エクスポートファイルタイプ] で、レポートのファイル形式を指定します。

   - データを含む JavaScript オブジェクト表記法 (.json) ファイルを作成するには、[JSON] を選択します。

     [JSON] オプションを選択した場合、レポートには各検出結果のすべてのフィールドが含まれます。使用可能な JSON フィールドのリストについては、Amazon Inspector API リファレンスの「検出結果データタイプ」を参照してください。

   - データを含むカンマ区切り値 (CSV) ファイルを作成するには、[CSV] を選択します。

     CSV オプションを選択した場合、レポートには各検出結果のフィールドのサブセット、つまり検出結果の主要な属性を報告する約 45 のフィールドのみが含まれます。フィールドには、[検出結果のタイプ]、[タイトル]、[重要度]、[ステータス]、[説明]、[初回確認日]、[最終確認日]、[使用可能な修正]、[AWS アカウント ID]、[リソース ID]、[リソースタグ]、および [対策] が含まれます。これらのフィールドに加え、各検出結果のスコアリングの詳細と参照 URL

をキャプチャするフィールドもあります。以下は、検出結果レポートの CSV ヘッダーのサンプルです。

AWS Account Id | Severity | Description | Title | Remediation | Resource Type | Resource Id | Resource Partition | Resource Region | Resource Tags | Resource Details | Vulnerability Id | Vulnerability Vendor | Vulnerability Reference Urls | Vulnerability Vendor Severity | Vulnerability Vendor Severity Source | Vulnerability Cvss Vector | Vulnerability Cvss Base Score | Vulnerability Cvss Version | Vulnerability Cvss Source | Vulnerability Exploit Available | Vulnerability Last Exploited At | Vulnerability Fixed In Version | Vulnerability Fix Available | Vulnerability Related Vulnerabilities | Vulnerability Source URL | Vulnerability Source Update At | Vulnerability Detection Platforms | Vulnerability Detection Path | Vulnerability Vendor Severity Scores | Vulnerability Vendor Severity Score Version | Vulnerability Epss Score | Vulnerability Epss Percentile | Vulnerability Epss Score Date | Vulnerability Epss Model Version | Vulnerability Epss Model Date | Vulnerability Epss Data Source | Vulnerability Epss Data Hash | Vulnerability Epss Data Timestamp | Vulnerability Epss Data Version | Vulnerability Epss Data Source URL | Vulnerability Epss Data Signature | Vulnerability Epss Data Public Key Id | Vulnerability Epss Data Signature Algorithm | Vulnerability Epss Data Hash Algorithm | Vulnerability Epss Data Content Type | Vulnerability Epss Data Expiration | Vulnerability Epss Data Retrieved At | Vulnerability Epss Data Last Modified At | Vulnerability Epss Data Cache Control | Vulnerability Epss Data Etag | Vulnerability Epss Data Error Message | Vulnerability Epss Data Error Code | Vulnerability Epss Data Error Type | Vulnerability Epss Data Error Retrieved At | Vulnerability Epss Data Error Last Updated At

6. [エクスポートの場所] の [S3 URI] で、レポートを保存する S3 バケットを指定します。

   - アカウントが所有するバケットにレポートを保存するには、[S3 の参照] を選択します。Amazon Inspector は、アカウントの S3 バケットのテーブルを表示します。使用したいバケットを選択し、[選択] を選択します。

     > **Tip**
     >
     > レポートの Amazon S3 パスプレフィックスも指定するには、[S3 URI] ボックスの値にスラッシュ (/) とプレフィックスを追加します。その後、Amazon Inspector はレポートをバケットに追加するときにプレフィックスを含め、Amazon S3 はプレフィックスで指定されたパスを生成します。
     > たとえば、 AWS アカウント ID をプレフィックスとして使用し、アカウント ID が 111122223333「」の場合、S3 URI ボックスの値**/111122223333**に を追加します。[プレフィックス] は、バケット内のディレクトリパスと類似しています。これにより、類似ファイルをファイルシステム上のフォルダにまとめて保存する場合と同様に、バケット内の類似オブジェクトをまとめてグループ化できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「フォルダを使用して Amazon S3 コンソールのオブジェクトを整理する」を参照してください。

   - 別のアカウントが所有するバケットにレポートを保存するには、そのバケットの URI を入力します。たとえば **s3://DOC-EXAMPLE_BUCKET**、DOC-EXAMPLE_BUCKET はバケットの名前です。バケット所有者は、この情報をバケットのプロパティで確認できます。

7. KMS キー AWS KMS key で、レポートの暗号化に使用する を指定します。

   - ご自分のアカウントのキーを使用するには、リストからキーを選択します。リストには、アカウントのカスタマーマネージド、対称暗号化 KMS キーが表示されます。
   - 別のアカウントによって所有されているキーを使用するには、キーの Amazon リソースネーム (ARN) を入力します。キーの所有者は、キーのプロパティでこの情報をユーザーに代わっ

て確認できます。詳細については、「AWS Key Management Service デベロッパーガイド」の「キー ID と ARN の検索」を参照してください。

8. [エクスポート] をクリックします。

Amazon Inspector は調査結果レポートを生成し、指定した KMS キーで暗号化して、指定した S3 バケットに追加します。レポートに含めるように選択した検出結果の数によっては、このプロセスに数分または数時間かかる場合があります。エクスポートが完了すると、Amazon Inspector は調査結果レポートが正常にエクスポートされたことを示すメッセージを表示します。オプションで、メッセージ内の [レポートを表示] を選択し、Amazon S3 のレポートに移動します。

一度に 1 つのレポートしかエクスポートできないことに注意してください。エクスポートが進行中の場合は、エクスポートが完了するまで待ってから別のレポートをエクスポートしてください。

エクスポートエラーのトラブルシューティング

調査結果レポートをエクスポートしようとしたときにエラーが発生した場合、Amazon Inspector では、エラーを説明するメッセージが表示されます。このトピックに記載されている情報を参考にして、考えられるエラーの原因と解決策を特定してください。

たとえば、S3 バケットが現在の にあり AWS リージョン 、バケットのポリシーで Amazon Inspector がバケットにオブジェクトを追加できることを確認します。また、 AWS KMS key が現在のリージョンで有効になっていることを確認し、キーポリシーで Amazon Inspector がキーを使用することが許可されていることを確認します。

エラーに対処したら、もう一度レポートのエクスポートを試します。

「Cannot have multiple reports」エラー

レポートを作成しようとしても、Amazon Inspector が既にレポートを生成している場合、「Reason: Cannot have multiple reports in-progress」というエラーが表示されます。このエラーは、Amazon Inspector がアカウントに対して一度に 1 つのレポートしか生成できないために発生します。

エラーを解決するには、他のレポートが終了するのを待つか、キャンセルしてから新しいレポートをリクエストしてください。

GetFindingsReportStatus オペレーションを使用してレポートのステータスを確認できます。この操作は、現在生成中のすべてのレポートのレポート ID を返します。

必要な場合は、CancelFindingsReport オペレーションを使用し、`GetFindingsReportStatus` オペレーションで指定されたレポート ID を使用して、進行中のエクスポートをキャンセルできます。

Amazon EventBridge を使用して Amazon Inspector の検出結果に対するカスタムレスポンスを作成する

Amazon Inspector は、新たに生成された検出結果や集計された検出結果に対して、Amazon EventBridge のイベントを作成します。Amazon Inspector は、検出結果の状態に対する変更のイベントも作成します。つまり、リソースの再起動やリソースに関連付けられているタグの変更などのアクションを実行すると、検出結果に関する新しいイベントが生成されます。Amazon Inspector が更新された検出結果の新しいイベントを作成すると、検出結果 `id` は同じままになります。

> **Note**
>
> アカウントが Amazon Inspector の委任管理者アカウントである場合、EventBridge はお客様のアカウントと、イベントの発行元であるメンバーアカウントにイベントを発行します。

Amazon Inspector で EventBridge イベントを使用すると、タスクを自動化し、検出結果によって明らかになったセキュリティ上の問題に対応できるようになります。EventBridge イベントに基づいて Amazon Inspector の検出結果に関する通知を受け取るには、EventBridge ルールを作成し、Amazon Inspector のターゲットを指定する必要があります。EventBridge ルールにより、EventBridge は Amazon Inspector の検出結果通知を送信でき、ターゲットは通知の送信先を指定します。

Amazon Inspector は、現在 Amazon Inspector を使用している のデフォルトのイベントバスにイベントを出力 AWS リージョン します。つまり、Amazon Inspector を AWS リージョン アクティブ化し、EventBridge イベントを受信するように Amazon Inspector を設定した各 のイベントルールを設定する必要があります。Amazon Inspector は、ベストエフォートベースでイベントを発行します。

このセクションでは、イベントスキーマの例と、EventBridge ルールの作成方法について説明します。

イベントスキーマ

以下は、EC2 の検出結果イベントの Amazon Inspector イベントフォーマットの例です。他の検出結果タイプやイベントタイプのスキーマの例については、「EventBridge スキーマ」を参照してください。

```
{
    "version": "0",
    "id": "66a7a279-5f92-971c-6d3e-c92da0950992",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "111122223333",
    "time": "2023-01-19T22:46:15Z",
    "region": "us-east-1",
    "resources": ["i-0c2a343f1948d5205"],
    "detail": {
        "awsAccountId": "111122223333",
        "description": "\n It was discovered that the sound subsystem in the Linux
 kernel contained a\n race condition in some situations. A local attacker could use
 this to cause\n a denial of service (system crash).",
        "exploitAvailable": "YES",
        "exploitabilityDetails": {
            "lastKnownExploitAt": "Oct 24, 2022, 11:08:59 PM"
        },
        "findingArn": "arn:aws:inspector2:us-east-1:111122223333:finding/FINDING_ID",
        "firstObservedAt": "Jan 19, 2023, 10:46:15 PM",
        "fixAvailable": "YES",
        "lastObservedAt": "Jan 19, 2023, 10:46:15 PM",
        "packageVulnerabilityDetails": {
            "cvss": [{
                "baseScore": 4.7,
                "scoringVector": "CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H",
                "source": "NVD",
                "version": "3.1"
            }],
            "referenceUrls": ["https://lore.kernel.org/all/
CAFcO6XN7JDM4xSXGhtusQfS2mSBcx50VJKwQpCq=WeLt57aaZA@mail.gmail.com/", "https://
ubuntu.com/security/notices/USN-5792-1", "https://ubuntu.com/security/notices/
USN-5791-2", "https://ubuntu.com/security/notices/USN-5791-1", "https://ubuntu.com/
security/notices/USN-5793-2", "https://git.kernel.org/pub/scm/linux/kernel/git/
torvalds/linux.git/commit/?id=8423f0b6d513b259fdab9c9bf4aaa6188d054c2d", "https://
ubuntu.com/security/notices/USN-5793-1", "https://ubuntu.com/security/notices/
USN-5792-2", "https://ubuntu.com/security/notices/USN-5791-3", "https://ubuntu.com/
security/notices/USN-5793-4", "https://ubuntu.com/security/notices/USN-5793-3",
 "https://git.kernel.org/linus/8423f0b6d513b259fdab9c9bf4aaa6188d054c2d(6.0-rc5)",
 "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-3303"],
            "relatedVulnerabilities": [],
            "source": "UBUNTU_CVE",
            "sourceUrl": "https://people.canonical.com/~ubuntu-security/cve/2022/
CVE-2022-3303.html",
```

```
            "vendorCreatedAt": "Sep 27, 2022, 11:15:00 PM",
            "vendorSeverity": "medium",
            "vulnerabilityId": "CVE-2022-3303",
            "vulnerablePackages": [{
                "arch": "X86_64",
                "epoch": 0,
                "fixedInVersion": "0:5.15.0.1027.31~20.04.16",
                "name": "linux-image-aws",
                "packageManager": "OS",
                "remediation": "apt update && apt install --only-upgrade linux-image-
aws",
                "version": "5.15.0.1026.30~20.04.16"
            }]
        },
        "remediation": {
            "recommendation": {
                "text": "None Provided"
            }
        },
        "resources": [{
            "details": {
                "awsEc2Instance": {
                    "iamInstanceProfileArn": "arn:aws:iam::111122223333:instance-
profile/AmazonSSMRoleForInstancesQuickSetup",
                    "imageId": "ami-0b7ff1a8d69f1bb35",
                    "ipV4Addresses": ["172.31.85.212", "44.203.45.27"],
                    "ipV6Addresses": [],
                    "launchedAt": "Jan 19, 2023, 7:53:14 PM",
                    "platform": "UBUNTU_20_04",
                    "subnetId": "subnet-8213f2a3",
                    "type": "t2.micro",
                    "vpcId": "vpc-ab6650d1"
                }
            },
            "id": "i-0c2a343f1948d5205",
            "partition": "aws",
            "region": "us-east-1",
            "type": "AWS_EC2_INSTANCE"
        }],
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "CVE-2022-3303 - linux-image-aws",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Jan 19, 2023, 10:46:15 PM"
```

```
    }
}
```

Amazon Inspector の検出結果を通知する EventBridge ルールの作成

Amazon Inspector の検出結果の可視性を高めるために、EventBridge を使用して自動検出結果アラートを設定し、メッセージングハブに送信することができます。このトピックでは、E メール、Slack、または Amazon Chime に、`CRITICAL` および `HIGH` の重要度の検出結果に対するアラートを送信する方法を説明します。Amazon Simple Notification Service のトピックを設定し、EventBridge イベントルールに関連付ける方法を説明します。

ステップ 1. Amazon SNS トピックおよびエンドポイントの設定

自動アラートを設定するには、まず Amazon Simple Notification Service でトピックを設定し、エンドポイントを追加する必要があります。詳細については、「SNS ガイド」を参照してください。

この手順では、Amazon Inspector の検出結果データを送信したい場所を設定します。SNS トピックは、イベントルールの作成中または作成後に EventBridge イベントルールに追加できます。

Email setup

SNS トピックの作成

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソール にサインインします。
2. ナビゲーションペインから [トピック] 、[トピックの作成] を選択します。
3. [トピックの作成] セクションで [標準] を選択します。次に、**`Inspector_to_Email`** のようなトピック名を入力します。その他の詳細はオプションです。
4. [Create Topic] (トピックの作成) を選択します。新しいトピックの詳細が表示された新しいパネルが開きます。
5. [サブスクリプション] セクションで、[サブスクリプションの作成] を選択します。
6. a. [Protocol] (プロトコル) メニューから [Email] (E メール) を選択します。

   b. [エンドポイント] フィールドに、通知を受信する E メールアドレスを入力します。

> **Note**
> サブスクリプションを作成後、E メールクライアントを通じてサブスクリプションを確認する必要があります。

c. [Create subscription] を選択してください。

7. 受信トレイでサブスクリプションのメッセージを検索し、[サブスクリプションを確認] を選択します。

Slack setup

SNS トピックの作成

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソール にサインインします。
2. ナビゲーションペインから [トピック] 、[トピックの作成] を選択します。
3. [トピックの作成] セクションで [標準] を選択します。次に、**`Inspector_to_Slack`** のようなトピック名を入力します。その他の詳細はオプションです。[トピックを作成] を選択してエンドポイントの作成を完了します。

チャットアプリケーションクライアントでの Amazon Q Developer の設定

1. のチャットアプリケーションコンソールで Amazon Q Developer に移動しますhttps://console.aws.amazon.com/chatbot/。
2. [設定されたクライアント] ペインから [新しいクライアントを設定] を選択します。
3. [Slack] を選択して確認し、[設定] を選択します。

> **Note**
> Slack を選択するときは、許可を選択して、チャットアプリケーションで Amazon Q Developer がチャネルにアクセスするためのアクセス許可を確認する必要があります。

4. [新しいチャネルを設定] を選択し、設定の詳細ペインを開きます。

   a. チャンネルの名前を入力します。

b. [Slack チャネル] で、使用したいチャネルを選択します。

c. Slack で、チャネル名を右クリックして [リンクのコピー] を選択して、コピーのリンクを選択することでプライベートチャンネルのチャネル ID をコピーします。

d. で AWS Management Console、チャットアプリケーションの Amazon Q Developer ウィンドウで、Slack からコピーしたチャネル ID をプライベートチャネル ID フィールドに貼り付けます。

e. [アクセス許可] で、まだロールを持っていない場合は、テンプレートを使用して IAM ロールを作成することを選択します。

f. [ポリシー] テンプレートで、[通知の許可] を選択します。これは、チャットアプリケーションにおける Amazon Q Developer の IAM ポリシーテンプレートです。このポリシーは、CloudWatch アラーム、イベント、ログ、および Amazon SNS トピックに必要な読み取りおよびリスト許可を提供します。

g. チャネルガードレールポリシーには、AmazonInspector2ReadOnlyAccess を選択します。

h. 以前に SNS トピックを作成したリージョンを選択し、Slack チャネルに通知を送信するために作成した Amazon SNS トピックを選択します。

5. [Configure] (設定) を選択します。

Amazon Chime setup

SNS トピックの作成

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソール にサインインします。
2. ナビゲーションペインから [トピック]、[トピックの作成] を選択します。
3. [トピックの作成] セクションで [標準] を選択します。次に、**`Inspector_to_Chime`** のようなトピック名を入力します。その他の詳細はオプションです。[トピックを作成] を選択して完了します。

チャットアプリケーションクライアントでの Amazon Q Developer の設定

1. のチャットアプリケーションコンソールで Amazon Q Developer に移動しますhttps://console.aws.amazon.com/chatbot/。
2. [Configured clients] (設定されたクライアント) パネルから [Configure new client] (新しいクライアントを設定) を選択します。

3. [Chime]、[設定] を選択して確認します。
4. [Configuration details] (設定の詳細) ペインから、チャンネルの名前を入力します。
5. Amazon Chime で目的のチャットルームを開きます。
   a. 右上の歯車アイコンを選択してから、[Manage webhooks and bots] (ウェブフックとボットの管理) を選択します。
   b. [Copy URL] (URL をコピー)を選択し、Webhook URL をクリップボードにコピーします。
6. で AWS Management Console、チャットアプリケーションの Amazon Q Developer ウィンドウで、コピーした URL を Webhook URL フィールドに貼り付けます。
7. [アクセス許可] で、まだロールを持っていない場合は、テンプレートを使用して IAM ロールを作成することを選択します。
8. [ポリシー] テンプレートで、[通知の許可] を選択します。これは、チャットアプリケーションにおける Amazon Q Developer の IAM ポリシーテンプレートです。CloudWatch アラーム、イベント、ログ、および Amazon SNS トピックに必要な読み取りおよびリスト許可を提供します。
9. 以前に SNS トピックを作成したリージョンを選択し、Amazon Chime ルームに通知を送信するために作成した Amazon SNS トピックを選択します。
10. [設定] を選択します。

ステップ 2. Amazon Inspector の検出結果の EventBridge ルールを作成する

1. 自分の認証情報を使用してサインインします。
2. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
3. ナビゲーションペインから [ルール] を選択し、[ルールの作成] を選択します。
4. ルールの名前と必要に応じて説明を入力します。
5. [イベントパターンを持つルール] を選択してから、[次へ] を選択します。
6. [イベントパターン] ペインで [カスタムパターン (JSON エディタ)] を選択します。
7. 以下の JSON をエディタに貼り付けます。

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"],
```

```
  "detail": {
    "severity": ["HIGH", "CRITICAL"],
    "status": ["ACTIVE"]
  }
}
```

> **Note**
>
> このパターンは、Amazon Inspector によって検出されたアクティブな `CRITICAL` または `HIGH` 重要度の検出結果に関する通知を送信します。

イベントパターンの入力が終了したら、[次へ] を選択します。

8. [ターゲットを選択] ページで、[AWS のサービス] を選択します。次に、[ターゲットタイプの選択] で [SNS トピック] を選択します。
9. [トピック] で、ステップ 1 で作成した SNS トピックの名前を選択します。次いで、[次へ] を選択します。
10. 必要に応じてオプションのタグを追加し、[次へ] を選択します。
11. ルールを確認し、[ルールの作成] を選択します。

Amazon Inspector マルチアカウント環境の EventBridge

Amazon Inspector の委任された管理者である場合、EventBridge ルールはメンバーアカウントの該当する検出結果に基づいてアカウントに表示されます。前のセクションで説明したように、管理者アカウントの EventBridge を通じて検出結果通知を設定すると、複数のアカウントに関する通知が届きます。つまり、ご自身のアカウントで生成された結果とイベントに加え、メンバーアカウントによって生成された結果とイベントが通知されます。

結果の JSON 詳細から `accountId` を使用して、Amazon Inspector の検出結果の元となったメンバーアカウントを特定することができます。

Amazon Inspector でのダッシュボードの操作

このダッシュボードには、Amazon Inspector がスキャンするリソースの集約された統計のスナップショットが表示されます。ダッシュボードを使用して、環境のカバレッジと緊急の検出結果について説明します。

> **Note**
>
> アカウントが組織の委任管理者アカウントである場合、ダッシュボードには、アカウントと組織内の他のすべてのアカウントに関する情報が表示されます。

このトピックでは、ダッシュボードを表示し、ダッシュボードを構成するコンポーネントを理解する方法について説明します。

トピック

- ダッシュボードの表示
- ダッシュボードコンポーネントを理解しデータを解釈する

ダッシュボードの表示

ダッシュボードには、環境の概要と緊急の検出結果が表示されます。ダッシュボードは 5 分ごとにデータを自動的に更新します。画面の右上隅にある更新アイコンを選択すると、データを手動で更新できます。項目を選択すると、項目のサポートデータを表示できます。

> **Note**
>
> アカウントが組織の委任管理者アカウントである場合は、[アカウント] フィールドにメンバーアカウント ID を入力することで、メンバーアカウントの集計された統計を表示できます。

ダッシュボードを表示するには:

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。

2. ナビゲーションペインから、[ダッシュボード] を選択します。

ダッシュボードコンポーネントを理解しデータを解釈する

ダッシュボードの各セクションでは、主要なメトリクスと検出結果データに関するインサイトが提供されるため、現在の AWS リソースの脆弱性体制を理解できます AWS リージョン。

環境カバレッジ

環境カバレッジセクションには、Amazon Inspector によってスキャンされたリソースに関する統計が表示されます。このセクションでは、Amazon Inspector によってスキャンされた Amazon EC2 インスタンス、Amazon ECR イメージ、 AWS Lambda 関数の数と割合を確認できます。Amazon Inspector Amazon Inspector の委任管理者 AWS Organizations として で複数のアカウントを管理する場合、組織アカウントの合計数、Amazon Inspector がアクティブ化された数、およびその結果の組織カバレッジの割合も表示されます。このセクションを使用して、Amazon Inspector の対象とならないリソースを特定することもできます。これらのリソースには、悪用されて組織を危険にさらす可能性のある脆弱性が含まれている可能性があります。詳細については、「AWS 環境の Amazon Inspector カバレッジの評価」を参照してください。

カバレッジグループを選択すると、選択したグループのアカウント管理ページに移動します。アカウント管理ページには、Amazon Inspector の対象となっているアカウント、Amazon EC2 インスタンス、Amazon ECR リポジトリに関する詳細が表示されます。

次のカバレッジグループが使用できます。

- アカウント
- インスタンス
- コンテナリポジトリ
- コンテナイメージ
- Lambda

緊急の検出結果

「緊急の検出結果」セクションには、環境内の重大な脆弱性の数と、環境内のすべての検出結果の合計数が表示されます。このセクションには、リソースと評価タイプごとの数が表示されます。緊急の検出結果と Amazon Inspector が緊急度を判断する方法の詳細については、「Amazon Inspector の検出結果について」を参照してください。

[緊急の検出結果グループ] を選択すると、[すべての検出結果] ページに移動し、選択したグループに一致するすべての緊急の検出結果を表示するフィルタが自動的に適用されます。

以下の緊急の検出結果グループを使用できます。

- Amazon Inspector コードスキャンの検出結果
- Amazon EC2 インスタンスの検出結果
- Amazon ECR コンテナイメージの検出結果
- Lambda 関数の検出結果

リスクベースの修正

リスクベースの修復セクションには、環境内のほとんどのリソースに影響を与える緊急の脆弱性がある上位 5 つのソフトウェアパッケージが表示されます。これらのパッケージを修正することで、環境に重大なリスクの数を大幅に減らすことができます。ソフトウェアパッケージ名を選択すると、関連する脆弱性の詳細と影響を受けるリソースが表示されます。

最も緊急の検出結果があるアカウント

最も重要な検出結果を持つアカウントセクションには、最も重要な検出結果を持つ環境内の上位 5 つの AWS アカウントと、そのアカウントの検出結果の合計数が表示されます。このセクションは、Amazon Inspector がマルチアカウントスキャン用に設定されている場合にのみ、委任管理者アカウントから表示できます AWS Organizations。このビューは、委任された管理者が組織内でどのアカウントが最も危険にさらされているかを把握するのに役立ちます。

[アカウント ID] を選択すると、影響を受けるメンバーアカウントに関する詳細情報が表示されます。

最も緊急の検出結果がある Amazon ECR リポジトリ

[最も緊急の検出結果がある Elastic Container Registry (ECR) リポジトリ] セクションには、お客様の環境内で最も緊急のコンテナイメージの検出結果がある上位 5 つの Amazon ECR リポジトリが表示されます。ビューには、リポジトリ名、 AWS アカウント識別子、リポジトリ作成日、重大な脆弱性の数、脆弱性の合計数が表示されます。このビューは、どのリポジトリが最も危険にさらされているかを特定するのに役立ちます。

リポジトリ名を選択すると、影響を受けるリポジトリに関する詳細が表示されます。

最も緊急の検出結果があるコンテナイメージ

[最も緊急の検出結果があるコンテナイメージ] セクションには、環境内で最も緊急の検出結果をがある上位 5 つのコンテナイメージが表示されます。ビューには、イメージタグデータ、リポジトリ名、イメージダイジェスト、 AWS アカウント識別子、重大な脆弱性の数、脆弱性の合計数が表示されます。このビューは、どのコンテナイメージを再構築して再起動する必要があるかをアプリケーション所有者が特定するのに役立ちます。

[コンテナイメージ] を選択すると、影響を受けるコンテナイメージに関する詳細が表示されます。

最も緊急の検出結果があるインスタンス

[最も緊急の検出結果があるインスタンス] セクションには、最も緊急の検出結果がある上位 5 つの Amazon EC2 インスタンスが表示されます。ビューには、インスタンス識別子、 AWS アカウント識別子、Amazon マシンイメージ (AMI) 識別子、緊急な脆弱性の数、および脆弱性の総数が表示されます。このビューは、インフラストラクチャの所有者がどのインスタンスにパッチを適用する必要があるかを特定するのに役立ちます。

[インスタンス ID] を選択すると、影響を受ける Amazon EC2 インスタンスの詳細が表示されます。

最も緊急の検出結果がある Amazon マシンイメージ (AMI)

[最も緊急の検出結果がある Amazon マシンイメージ (AMI)] セクションには、環境内で最も緊急の検出結果を含む上位 5 つの AMI が表示されます。ビューには、AMI 識別子、 AWS アカウント識別子、環境内で実行されている影響を受ける EC2 インスタンスの数、AMI の作成日、AMI のオペレーティングシステムプラットフォーム、緊急な脆弱性の数、および脆弱性の総数が表示されます。このビューは、インフラストラクチャーの所有者が再構築が必要な AMI を特定するのに役立ちます。

[影響を受けるインスタンス] を選択すると、影響を受ける AMI から起動したインスタンスの詳細情報が表示されます。

AWS Lambda 最も重要な検出結果を持つ 関数

最も緊急の検出結果が出たAWS Lambda 関数のセクションには、環境内の最も緊急な検出結果を含む上位 5 つの Lambda 関数が表示されます。ビューには、Lambda 関数名、 AWS アカウント識別子、ランタイム環境、重大な脆弱性の数、高い脆弱性の数、脆弱性の合計数が表示されます。このビューは、インフラストラクチャの所有者が修正が必要な Lambda 機能を特定するのに役立ちます。

関数名を選択すると、影響を受ける AWS Lambda 関数に関する詳細情報が表示されます。

最も重要な検出結果を含む Amazon Inspector コードスキャン

最も重要なコードの脆弱性を持つプロジェクトセクションには、重要な検出結果を持つ上位 5 つのプロジェクトが表示されます。プロジェクトを選択して、検出結果の詳細を表示できます。プロジェクトを選択すると、検出結果があるリポジトリに移動します。検出結果タブには、検出結果の名前とその重要度評価が表示されます。検出結果の生成に使用された分析のタイプを示します。また、検出結果の経過時間とそのステータスも表示されます。

Amazon Inspector 脆弱性データベースの検索

Amazon Inspector の脆弱性データベースで、共通脆弱性識別子 (CVE) を検索できます。Amazon Inspector は、脆弱性データベースからの情報を使用して、CVE ID に関連する詳細を生成します。これらの詳細は、CVE の詳細画面で確認できます。Amazon Inspector は、脆弱性データベース内のソフトウェア脆弱性を追跡して検出結果を生成します。Amazon Inspector は、CVE の詳細画面の [検出プラットフォーム] セクションに一覧表示されているプラットフォームを持つ CVE のみをサポートします。このセクションでは、CVE ID を使用して Amazon Inspector の脆弱性データベースを検索する方法について説明します。

> **Note**
>
> 現在のところ、CVE 検索では Microsoft Windows がサポートされていません。

脆弱性データベースの検索

このセクションでは、コンソールおよび Amazon Inspector API で脆弱性データベースを検索する方法について説明します。

> **Note**
>
> 脆弱性データベースを検索する AWS リージョン 前に、現在の で Amazon Inspector をアクティブ化する必要があります。

Console

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、[脆弱性データベース検索] を選択します。
3. 検索バーに CVE ID を入力し、[検索] を選択します。

API

Amazon Inspector の SearchVulnerabilities API を実行し、`filterCriteria` として 1 つの CVE ID を次の形式で指定します: `CVE-<year>-<ID>`。

CVE の詳細について

このセクションでは、CVE の詳細ページを解釈する方法について説明します。

CVE の詳細

CVE の詳細セクションには、以下の情報が含まれます。

- CVE の説明と ID
- CVE の重要度
- 共通脆弱性評価システム (CVSS) および Exploit Prediction Scoring System (EPSS) スコア
- 検出プラットフォーム

> **Note**
>
> このフィールドが空の場合、Amazon Inspector は CVE ID の検出をサポートしていません。

- 共通脆弱性タイプ一覧 (CWE)
- ベンダーの作成日および更新日

脆弱性インテリジェンス

脆弱性インテリジェンスセクションでは、悪用ターゲットや、悪用が明らかになった既知の最終日などの脅威インテリジェンスデータを提供します。

また、Cybersecurity and Infrastructure Security Agency (CISA) からのデータも提供します。これには、修復アクション、CVE が Known Exploited Vulnerability カタログに追加された日付、および CISA が連邦政府機関による CVE の修復を期待する日時が含まれます。

リファレンス

リファレンスセクションには、CVE の詳細に関するリソースへのリンクが用意されています。

Amazon Inspector による SBOM のエクスポート

ソフトウェア部品表 (SBOM) は、コードベースに含まれるすべてのオープンソースソフトウェアコンポーネントとサードパーティーソフトウェアコンポーネントのネストされたインベントリです。Amazon Inspector は、環境内の個々のリソースに SBOM を提供します。Amazon Inspector コンソールまたは Amazon Inspector API を使用して、リソースの SBOM を生成できます。Amazon Inspector がサポートおよびモニタリングしているすべてのリソースの SBOM をエクスポートできます。エクスポートされた SBOM は、ソフトウェアサプライに関する情報を提供します。AWS 環境のカバレッジを評価することで、リソースのステータスを確認できます。このセクションでは、SBOM を設定し、エクスポートする方法について説明します。

> **Note**
>
> 現在のところ、Amazon Inspector では Windows Amazon EC2 インスタンス用の SBOM のエクスポートはサポートしていません。

Amazon Inspector 形式

Amazon Inspector では、CycloneDX 1.4 および SPDX 2.3 互換フォーマットでの SBOM のエクスポートをサポートしています。Amazon Inspector では、選択した Amazon S3 バケットに SBOM を `JSON` ファイルとしてエクスポートします。

> **Note**
>
> Amazon Inspector からの SPDX 形式のエクスポートは SPDX 2.3 を使用するシステムと互換性がありますが、クリエイティブコモンズゼロ (CC0) フィールドは含まれていません。これは、このフィールドを含めると、ユーザーがマテリアルを再配布したり編集したりできるようになるためです。

Amazon Inspector の CycloneDX 1.4 SBOM 形式の例

```
                    {
"bomFormat": "CycloneDX",
"specVersion": "1.4",
"version": 1,
```

```
  "metadata": {
    "timestamp": "2023-06-02T01:17:46Z",
    "component": null,
    "properties": [
      {
        "name": "imageId",
        "value":
 "sha256:c8ee97f7052776ef223080741f61fcdf6a3a9107810ea9649f904aa4269fdac6"
      },
      {
        "name": "architecture",
        "value": "arm64"
      },
      {
        "name": "accountId",
        "value": "111122223333"
      },
      {
        "name": "resourceType",
        "value": "AWS_ECR_CONTAINER_IMAGE"
      }
    ]
  },
  "components": [
    {
      "type": "library",
      "name": "pip",
      "purl": "pkg:pypi/pip@22.0.4?path=usr/local/lib/python3.8/site-packages/
pip-22.0.4.dist-info/METADATA",
      "bom-ref": "98dc550d1e9a0b24161daaa0d535c699"
    },
    {
      "type": "application",
      "name": "libss2",
      "purl": "pkg:dpkg/libss2@1.44.5-1+deb10u3?
arch=ARM64&epoch=0&upstream=libss2-1.44.5-1+deb10u3.src.dpkg",
      "bom-ref": "2f4d199d4ef9e2ae639b4f8d04a813a2"
    },
    {
      "type": "application",
      "name": "liblz4-1",
      "purl": "pkg:dpkg/liblz4-1@1.8.3-1+deb10u1?
arch=ARM64&epoch=0&upstream=liblz4-1-1.8.3-1+deb10u1.src.dpkg",
      "bom-ref": "9a6be8907ead891b070e60f5a7b7aa9a"
```

```
    },
    {
      "type": "application",
      "name": "mawk",
      "purl": "pkg:dpkg/mawk@1.3.3-17+b3?
arch=ARM64&epoch=0&upstream=mawk-1.3.3-17+b3.src.dpkg",
      "bom-ref": "c2015852a729f97fde924e62a16f78a5"
    },
    {
      "type": "application",
      "name": "libgmp10",
      "purl": "pkg:dpkg/libgmp10@6.1.2+dfsg-4+deb10u1?
arch=ARM64&epoch=2&upstream=libgmp10-6.1.2+dfsg-4+deb10u1.src.dpkg",
      "bom-ref": "52907290f5beef00dff8da77901b1085"
    },
    {
      "type": "application",
      "name": "ncurses-bin",
      "purl": "pkg:dpkg/ncurses-bin@6.1+20181013-2+deb10u3?
arch=ARM64&epoch=0&upstream=ncurses-bin-6.1+20181013-2+deb10u3.src.dpkg",
      "bom-ref": "cd20cfb9ebeeadba3809764376f43bce"
    }
  ],
  "vulnerabilities": [
    {
      "id": "CVE-2022-40897",
      "affects": [
        {
          "ref": "a74a4862cc654a2520ec56da0c81cdb3"
        },
        {
          "ref": "0119eb286405d780dc437e7dbf2f9d9d"
        }
      ]
    }
  ]
}
```

Amazon Inspector の SPDX 2.3 SBOM 形式の例

```
{
```

```
 "name": "409870544328/EC2/i-022fba820db137c64/ami-074ea14c08effb2d8",
 "spdxVersion": "SPDX-2.3",
 "creationInfo": {
  "created": "2023-06-02T21:19:22Z",
  "creators": [
   "Organization: 409870544328",
   "Tool: Amazon Inspector SBOM Generator"
  ]
 },
 "documentNamespace": "EC2://i-022fba820db137c64/AMAZON_LINUX_2/null/x86_64",
 "comment": "",
 "packages": [{
   "name": "elfutils-libelf",
   "versionInfo": "0.176-2.amzn2",
   "downloadLocation": "NOASSERTION",
   "sourceInfo": "/var/lib/rpm/Packages",
   "filesAnalyzed": false,
   "externalRefs": [{
    "referenceCategory": "PACKAGE-MANAGER",
    "referenceType": "purl",
    "referenceLocator": "pkg:rpm/elfutils-libelf@0.176-2.amzn2?
arch=X86_64&epoch=0&upstream=elfutils-libelf-0.176-2.amzn2.src.rpm"
   }],
   "SPDXID": "SPDXRef-Package-rpm-elfutils-libelf-ddf56a513c0e76ab2ae3246d9a91c463"
  },
  {
   "name": "libcurl",
   "versionInfo": "7.79.1-1.amzn2.0.1",
   "downloadLocation": "NOASSERTION",
   "sourceInfo": "/var/lib/rpm/Packages",
   "filesAnalyzed": false,
   "externalRefs": [{
     "referenceCategory": "PACKAGE-MANAGER",
     "referenceType": "purl",
     "referenceLocator": "pkg:rpm/libcurl@7.79.1-1.amzn2.0.1?
arch=X86_64&epoch=0&upstream=libcurl-7.79.1-1.amzn2.0.1.src.rpm"
    },
    {
     "referenceCategory": "SECURITY",
     "referenceType": "vulnerability",
     "referenceLocator": "CVE-2022-32205"
    }
   ],
   "SPDXID": "SPDXRef-Package-rpm-libcurl-710fb33829bc5106559bcd380cddb7d5"
```

```
  },
  {
   "name": "hunspell-en-US",
   "versionInfo": "0.20121024-6.amzn2.0.1",
   "downloadLocation": "NOASSERTION",
   "sourceInfo": "/var/lib/rpm/Packages",
   "filesAnalyzed": false,
   "externalRefs": [{
    "referenceCategory": "PACKAGE-MANAGER",
    "referenceType": "purl",
    "referenceLocator": "pkg:rpm/hunspell-en-US@0.20121024-6.amzn2.0.1?
arch=NOARCH&epoch=0&upstream=hunspell-en-US-0.20121024-6.amzn2.0.1.src.rpm"
   }],
   "SPDXID": "SPDXRef-Package-rpm-hunspell-en-US-de19ae0883973d6cea5e7e079d544fe5"
  },
  {
   "name": "grub2-tools-minimal",
   "versionInfo": "2.06-2.amzn2.0.6",
   "downloadLocation": "NOASSERTION",
   "sourceInfo": "/var/lib/rpm/Packages",
   "filesAnalyzed": false,
   "externalRefs": [{
     "referenceCategory": "PACKAGE-MANAGER",
     "referenceType": "purl",
     "referenceLocator": "pkg:rpm/grub2-tools-minimal@2.06-2.amzn2.0.6?
arch=X86_64&epoch=1&upstream=grub2-tools-minimal-2.06-2.amzn2.0.6.src.rpm"
    },
    {
     "referenceCategory": "SECURITY",
     "referenceType": "vulnerability",
     "referenceLocator": "CVE-2021-3981"
    }
   ],
   "SPDXID": "SPDXRef-Package-rpm-grub2-tools-minimal-c56b7ea76e5a28ab8f232ef6d7564636"
  },
  {
   "name": "unixODBC-devel",
   "versionInfo": "2.3.1-14.amzn2",
   "downloadLocation": "NOASSERTION",
   "sourceInfo": "/var/lib/rpm/Packages",
   "filesAnalyzed": false,
   "externalRefs": [{
    "referenceCategory": "PACKAGE-MANAGER",
    "referenceType": "purl",
```

```
    "referenceLocator": "pkg:rpm/unixODBC-devel@2.3.1-14.amzn2?
arch=X86_64&epoch=0&upstream=unixODBC-devel-2.3.1-14.amzn2.src.rpm"
   }],
   "SPDXID": "SPDXRef-Package-rpm-unixODBC-devel-1bb35add92978df021a13fc9f81237d2"
  }
 ],
 "relationships": [{
   "spdxElementId": "SPDXRef-DOCUMENT",
   "relatedSpdxElement": "SPDXRef-Package-rpm-elfutils-libelf-
ddf56a513c0e76ab2ae3246d9a91c463",
   "relationshipType": "DESCRIBES"
  },
  {
   "spdxElementId": "SPDXRef-DOCUMENT",
   "relatedSpdxElement": "SPDXRef-Package-rpm-yajl-8476ce2db98b28cfab2b4484f84f1903",
   "relationshipType": "DESCRIBES"
  },
  {
   "spdxElementId": "SPDXRef-DOCUMENT",
   "relatedSpdxElement": "SPDXRef-Package-rpm-unixODBC-
devel-1bb35add92978df021a13fc9f81237d2",
   "relationshipType": "DESCRIBES"
  }
 ],
 "SPDXID": "SPDXRef-DOCUMENT"
}
```


SBOM 用フィルター

SBOM をエクスポートする際、フィルターを追加してリソースの特定のサブセットに関するレポートを作成できます。フィルターを指定しない場合、サポート対象のすべてのアクティブリソースの SBOM がエクスポートされます。また、委任された管理者の場合は、メンバー全員のリソースも含まれます。以下のフィルタが利用可能です。

- AccountID — このフィルターは、特定の AccountID に関連付けられている任意のリソースの SBOM をエクスポートするために使用できます。
- EC2 インスタンスタグ — このフィルターは、特定のタグを持つ EC2 インスタンスの SBOM をエクスポートするために使用できます。
- 関数名 — このフィルターは、特定の Lambda 関数の SBOM をエクスポートするために使用できます。

- イメージタグ — このフィルターは、特定のタグが付いたコンテナイメージの SBOM をエクスポートするために使用できます。
- Lambda 関数タグ — このフィルターを使用して、特定のタグを持つ Lambda 関数の SBOM をエクスポートできます。
- リソースタイプ — このフィルターは、リソースタイプ EC2/ECR/Lambda をフィルタリングするために使用できます。
- リソース ID — このフィルターは、特定のリソースの SBOM をエクスポートするために使用できます。
- リポジトリ名 — このフィルターを使用して、特定のリポジトリ内のコンテナイメージの SBOM を生成できます。

SBOM の設定とエクスポート

SBOMsエクスポートするには、まず Amazon S3 バケットと Amazon Inspector が使用できる AWS KMS キーを設定する必要があります。フィルタを使用して、リソースの特定のサブセットの SBOM をエクスポートできます。 AWS 組織内の複数のアカウントの SBOMs をエクスポートするには、Amazon Inspector の委任管理者としてサインインしている間に、以下の手順に従います。

前提条件

- Amazon Inspector によってアクティブにモニタリングされているサポート対象リソース。
- Amazon Inspector にオブジェクトの追加を許可するポリシーが設定された Amazon S3 バケット。ポリシーの設定については、「Configure export permissions」を参照してください。
- Amazon Inspector が を使用してレポートを暗号化できるようにするポリシーで設定された AWS KMS キー。ポリシーの設定については、「エクスポート用の AWS KMS キーを設定する」を参照してください。

> **Note**
>
> 以前に Amazon S3 バケットと検出結果のエクスポート用の AWS KMS キーを設定している場合は、SBOM エクスポートに同じバケットと キーを使用できます。

任意のアクセス方法を選択して、SBOM をエクスポートします。

Console

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ページの右上隅にある AWS リージョン セレクターを使用して、SBOM をエクスポートするリソースがあるリージョンを選択します。
3. ナビゲーションペインの [SBOM をエクスポート] を選択します。
4. (オプション) [SBOM のエクスポート] ページで、[フィルターを追加] メニューを使用して、レポートを作成するリソースのサブセットを選択します。フィルターが指定されていない場合、Amazon Inspector はすべてのアクティブなリソースのレポートをエクスポートします。委任された管理者の場合は、これには組織内のすべてのアクティブなリソースが含まれます。
5. [エクスポート設定] で、SBOM に必要な形式を選択します。
6. Amazon S3 URI を入力するか、[Amazon S3 を参照] を選択して SBOM を保存する Amazon S3 の場所を選択します。
7. Amazon Inspector がレポートの暗号化に使用するように設定したAWS KMS キーを入力します。

API

- リソースの SBOM をプログラムでエクスポートするには、Amazon Inspector API の CreatesBOMExport オペレーションを使用します。

  リクエストでは、`reportFormat` パラメータを使用して SBOM 出力形式を指定し、`CYCLONEDX_1_4` または `SPDX_2_3` を選択します。`s3Destination` パラメータは必須で、Amazon Inspector による書き込みを許可するポリシーで設定された S3 バケットを指定する必要があります。オプションで、`resourceFilterCriteria` パラメーターを使用して、レポートのスコープを特定のリソースに制限します。

AWS CLI

- を使用してリソースSBOMs をエクスポートするには、次のコマンド AWS Command Line Interface を実行します。

```
aws inspector2 create-sbom-export --report-format
FORMAT --s3-destination bucketName=amzn-s3-demo-
bucket1,keyPrefix=PREFIX,kmsKeyArn=arn:aws:kms:Region:111122223333:key/123
```

リクエスト内の *FORMAT* を任意の形式 (CYCLONEDX_1_4 または SPDX_2_3) に置き換えます。次に、S3 送信先の *user input placeholders* を、エクスポート先の S3 バケットの名前、S3 での出力に使用するプレフィックス、およびレポートの暗号化に使用している KMS キーの ARN に置き換えます。

Amazon Inspector イベントのAmazon EventBridge イベントスキーマ

Amazon EventBridge は、アプリケーションやその他の から、 AWS Lambda 関数、Amazon Simple Notification Service トピック、Amazon Kinesis Data Streams のデータストリームなどのターゲット AWS のサービス にリアルタイムデータのストリームを配信します。他のアプリケーション、サービス、システムとの統合をサポートするために、Amazon Inspector は検出結果をイベントとして自動的に EventBridge に公開します。Amazon Inspector を使用して、検出結果、カバレッジ、スキャンのイベントを発行できます。このセクションでは、EventBridge イベントのスキーマの例を示します。

トピック

- Amazon Inspector 用の Amazon EventBridge ベーススキーマ
- Amazon Inspector 検出結果イベントスキーマの例
- Amazon Inspector の初回スキャン完了イベントスキーマの例
- Amazon Inspector カバレッジイベントスキーマの例
- Amazon Inspector 自動有効化スキーマの例

Amazon Inspector 用の Amazon EventBridge ベーススキーマ

以下は、Amazon Inspector の EventBridge イベントのベーススキーマの例です。イベントの詳細は、イベントのタイプによって異なります。

```
{
    "version": "0",
    "id": "Event ID",
    "detail-type": "Inspector2 *event type*",
    "source": "aws.inspector2",
    "account": "AWS ##### ID (string)",
    "time": "event timestamp (string)",
    "region": "AWS ##### (string)",
    "resources": [
        *IDs or ARNs of the resources involved in the event*
    ],
    "detail": {
        *Details of an Amazon Inspector event type*
    }
```

```
}
```

Amazon Inspector 検出結果イベントスキーマの例

以下は、Amazon Inspector 検出結果の EventBridge イベントのスキーマの例です。検出結果イベントは、Amazon Inspector がリソースの 1 つでソフトウェアの脆弱性またはネットワークの問題を特定したときに作成されます。このタイプのイベントに対して通知を作成するガイドについては、「Amazon EventBridge を使用して Amazon Inspector の検出結果に対するカスタムレスポンスを作成する」を参照してください。

以下のフィールドは検出結果イベントを識別します。

- `detail-type` は `Inspector2 Finding` に設定されます。
- `detail` は検出結果について説明します。
- `detail.resources.tags` は、キーと値のデータが保存される場所です。

タブをフィルタリングして、さまざまなリソースのイベントスキーマの検出結果と検出結果タイプを表示できます。

Amazon EC2 package vulnerability finding

```
{
    "version": "0",
    "id": "4d621919-f1f4-4201-a0e2-37e4e330ff51",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-09-04T17:00:36Z",
    "region": "eu-central-1",
    "resources": [
        "i-12345678901234567"
    ],
    "detail": {
        "awsAccountId": "123456789012",
        "description": "In snapd versions prior to 2.62, snapd failed to properly
 check the destination of symbolic links when extracting a snap. The snap format
 is a squashfs file-system image and so can contain symbolic links and other file
 types. Various file entries within the snap squashfs image (such as icons and
 desktop files etc) are directly read by snapd when it is extracted. An attacker who
```

```
 could convince a user to install a malicious snap which contained symbolic links
 at these paths could then cause snapd to write out the contents of the symbolic
 link destination into a world-readable directory. This in-turn could allow an
 unprivileged user to gain access to privileged information.",
        "epss": {
            "score": 0.00043
        },
        "exploitAvailable": "NO",
        "findingArn": "arn:aws:inspector2:eu-
central-1:123456789012:finding/FINDING_ID",
        "firstObservedAt": "Wed Sep 04 16:59:44.356 UTC 2024",
        "fixAvailable": "YES",
        "inspectorScore": 4.8,
        "inspectorScoreDetails": {
            "adjustedCvss": {
                "adjustments": [],
                "cvssSource": "UBUNTU_CVE",
                "score": 4.8,
                "scoreSource": "UBUNTU_CVE",
                "scoringVector": "CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:L",
                "version": "3.1"
            }
        },
        "lastObservedAt": "Wed Sep 04 16:59:44.476 UTC 2024",
        "packageVulnerabilityDetails": {
            "cvss": [
                {
                    "baseScore": 4.8,
                    "scoringVector": "CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:L",
                    "source": "UBUNTU_CVE",
                    "version": "3.1"
                },
                {
                    "baseScore": 7.3,
                    "scoringVector": "CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H",
                    "source": "NVD",
                    "version": "3.1"
                }
            ],
            "referenceUrls": [
                "https://www.cve.org/CVERecord?id=CVE-2024-29069",
                "https://ubuntu.com/security/notices/USN-6940-1"
            ],
            "relatedVulnerabilities": [
```

```
                "USN-6940-1"
            ],
            "source": "UBUNTU_CVE",
            "sourceUrl": "https://people.canonical.com/~ubuntu-security/cve/2024/
CVE-2024-29069.html",
            "vendorCreatedAt": "Thu Jul 25 20:15:00.000 UTC 2024",
            "vendorSeverity": "medium",
            "vulnerabilityId": "CVE-2024-29069",
            "vulnerablePackages": [
                {
                    "arch": "ALL",
                    "epoch": 0,
                    "fixedInVersion": "0:2.63+22.04ubuntu0.1",
                    "name": "snapd",
                    "packageManager": "OS",
                    "remediation": "apt-get update && apt-get upgrade",
                    "version": "2.63"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "None Provided"
            }
        },
        "resources": [
            {
                "details": {
                    "awsEc2Instance": {
                        "iamInstanceProfileArn":
 "arn:aws:iam::123456789012:instance-profile/AmazonSSMRoleForInstancesQuickSetup",
                        "imageId": "ami-02ff980600c693b38",
                        "ipV4Addresses": [
                            "1.23.456.789",
                            "123.45.67.890"
                        ],
                        "ipV6Addresses": [],
                        "launchedAt": "Wed Sep 04 16:57:40.000 UTC 2024",
                        "platform": "UBUNTU_22_04",
                        "subnetId": "subnet-12345678",
                        "type": "t2.small",
                        "vpcId": "vpc-12345678"
                    }
                },
```

```
                "id": "i-12345678901234567",
                "partition": "aws",
                "region": "eu-central-1",
                "type": "AWS_EC2_INSTANCE"
            }
        ],
        "severity": "MEDIUM",
        "status": "CLOSED",
        "title": "CVE-2024-29069 - snapd",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Wed Sep 04 17:00:36.951 UTC 2024"
    }
}
```

Amazon EC2 network reachability finding

```
{
    "version": "0",
    "id": "9eb1603b-4263-19ec-8be2-33184694cb92",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-09-05T13:06:56Z",
    "region": "eu-central-1",
    "resources": ["i-12345678901234567"],
    "detail": {
        "awsAccountId": "123456789012",
        "description": "On the instance i-12345678901234567, the port range
 22-22 is reachable from the InternetGateway igw-261bab4d from an attached ENI
 eni-094ad651219472857.",
        "findingArn": "arn:aws:inspector2:eu-
central-1:123456789012:finding/FINDING_ID",
        "firstObservedAt": "Thu Sep 05 13:06:56.334 UTC 2024",
        "lastObservedAt": "Thu Sep 05 13:06:56.334 UTC 2024",
        "networkReachabilityDetails": {
            "networkPath": {
                "steps": [{
                    "componentId": "igw-261bab4d",
                    "componentType": "AWS::EC2::InternetGateway"
                }, {
                    "componentId": "acl-171b527d",
```

```
                    "componentType": "AWS::EC2::NetworkAcl"
                }, {
                    "componentId": "sg-0d34debf87410f2d9",
                    "componentType": "AWS::EC2::SecurityGroup"
                }, {
                    "componentId": "eni-094ad651219472857",
                    "componentType": "AWS::EC2::NetworkInterface"
                }, {
                    "componentId": "i-12345678901234567",
                    "componentType": "AWS::EC2::Instance"
                }]
            },
            "openPortRange": {
                "begin": 22,
                "end": 22
            },
            "protocol": "TCP"
        },
        "remediation": {
            "recommendation": {
                "text": "You can restrict access to your instance by modifying the
 Security Groups or ACLs in the network path."
            }
        },
        "resources": [{
            "details": {
                "awsEc2Instance": {
                    "iamInstanceProfileArn": "arn:aws:iam::123456789012:instance-
profile/AmazonSSMRoleForInstancesQuickSetup",
                    "imageId": "ami-02ff980600c693b38",
                    "ipV4Addresses": ["1.23.456.789", "123.45.67.890"],
                    "ipV6Addresses": [],
                    "launchedAt": "Wed Sep 04 17:41:24.000 UTC 2024",
                    "platform": "UBUNTU_22_04",
                    "subnetId": "subnet-12345678",
                    "type": "t2.small",
                    "vpcId": "vpc-12345678"
                }
            },
            "id": "i-12345678901234567",
            "partition": "aws",
            "region": "eu-central-1",
            "type": "AWS_EC2_INSTANCE"
        }],
```

```
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "Port 22 is reachable from an Internet Gateway - TCP",
        "type": "NETWORK_REACHABILITY",
        "updatedAt": "Thu Sep 05 13:06:56.334 UTC 2024"
    }
}
```

Amazon ECR package vulnerability finding

```
{
    "version": "0",
    "id": "5325facf-a1aa-7d97-6bce-25fde6f6d2fc",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-09-04T16:55:38Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:ecr:eu-central-1:123456789012:repository/inspector2/
sha256:84f507df33c6864d49c296fb734192696e4cb6f78166ac51ac8b9b118181085d"
    ],
    "detail.resources.tags.testkey": "allow",
    "detail": {
        "awsAccountId": "123456789012",
        "description": "Possible denial of service in X.509 name checks",
        "epss": {
            "score": 0.00045
        },
        "exploitAvailable": "NO",
        "findingArn": "arn:aws:inspector2:eu-
central-1:123456789012:finding/FINDING_ID",
        "firstObservedAt": "Wed Sep 04 16:55:38.411 UTC 2024",
        "fixAvailable": "YES",
        "lastObservedAt": "Wed Sep 04 16:55:38.411 UTC 2024",
        "packageVulnerabilityDetails": {
            "cvss": [],
            "referenceUrls": [
                "https://www.cve.org/CVERecord?id=CVE-2024-6119",
                "https://ubuntu.com/security/notices/USN-6986-1"
            ],
```

```
            "relatedVulnerabilities": [
                "USN-6986-1"
            ],
            "source": "UBUNTU_CVE",
            "sourceUrl": "https://people.canonical.com/~ubuntu-security/cve/2024/
CVE-2024-6119.html",
            "vendorCreatedAt": "Tue Sep 03 00:00:00.000 UTC 2024",
            "vendorSeverity": "medium",
            "vulnerabilityId": "CVE-2024-6119",
            "vulnerablePackages": [
                {
                    "arch": "ARM64",
                    "epoch": 0,
                    "fixedInVersion": "0:3.0.13-0ubuntu3.4",
                    "name": "libssl3t64",
                    "packageManager": "OS",
                    "release": "0ubuntu3.2",
                    "remediation": "apt-get update && apt-get upgrade",
                    "sourceLayerHash":
 "sha256:1567e7ea90b67fc95ccdeeec39bdc3045098dee7e0c604975b957a9f8c0e9616",
                    "version": "3.0.13"
                },
                {
                    "arch": "ARM64",
                    "epoch": 0,
                    "fixedInVersion": "0:3.0.13-0ubuntu3.4",
                    "name": "openssl",
                    "packageManager": "OS",
                    "release": "0ubuntu3.2",
                    "remediation": "apt-get update && apt-get upgrade",
                    "sourceLayerHash":
 "sha256:1567e7ea90b67fc95ccdeeec39bdc3045098dee7e0c604975b957a9f8c0e9616",
                    "version": "3.0.13"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "None Provided"
            }
        },
        "resources": [
            {
                "details": {
```

```
                    "awsEcrContainerImage": {
                        "architecture": "arm64",
                        "imageHash":
 "sha256:84f507df33c6864d49c296fb734192696e4cb6f78166ac51ac8b9b118181085d",
                        "imageTags": [
                            "ubuntu_latest"
                        ],
                        "platform": "UBUNTU_24_04",
                        "pushedAt": "Wed Sep 04 16:55:28.000 UTC 2024",
                        "registry": "123456789012",
                        "repositoryName": "inspector2"
                    }
                },
                "id": "arn:aws:ecr:eu-central-1:123456789012:repository/inspector2/
sha256:84f507df33c6864d49c296fb734192696e4cb6f78166ac51ac8b9b118181085d",
                "partition": "aws",
                "region": "eu-central-1",
                "type": "AWS_ECR_CONTAINER_IMAGE"
            }
        ],
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "CVE-2024-6119 - libssl3t64, openssl",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Wed Sep 04 16:55:38.411 UTC 2024"
    }
}
```

Lambda package vulnerability finding

```
{
    "version": "0",
    "id": "9eadd71a-e49c-9864-6ba9-2a5d3f83c88f",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-09-04T16:50:37Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:lambda:eu-central-1:123456789012:function:VulnerableFunction:
$LATEST"
```

```
    ],
    "detail": {
        "awsAccountId": "123456789012",
        "description": "Flask is a lightweight WSGI web application framework. When
 all of the following conditions are met, a response containing data intended for
 one client may be cached and subsequently sent by the proxy to other clients. If
 the proxy also caches `Set-Cookie` headers, it may send one client's `session`
 cookie to other clients. The severity depends on the application's use of the
 session and the proxy's behavior regarding cookies. The risk depends on all these
 conditions being met.\n\n1. The application must be hosted behind a caching proxy
 that does not strip cookies or ignore responses with cookies. 2. The application
 sets `session.permanent = True` 3. The application does not access or modify the
 session at any point during a request. 4. `SESSION_REFRESH_EACH_REQUEST` enabled
 (the default). 5. The application does not set a `Cache-Control` header to indicate
 that a page is private or should not be cached.\n\nThis happens because vulnerable
 versions of Flask only set the `Vary: Cookie` header when the session is ac",
        "epss": {
            "score": 0.00208
        },
        "exploitAvailable": "YES",
        "exploitabilityDetails": {
            "lastKnownExploitAt": "Sat Aug 31 00:04:50.000 UTC 2024"
        },
        "findingArn": "arn:aws:inspector2:eu-
central-1:123456789012:finding/FINDING_ID",
        "firstObservedAt": "Wed Sep 04 16:50:37.627 UTC 2024",
        "fixAvailable": "YES",
        "inspectorScore": 7.5,
        "inspectorScoreDetails": {
            "adjustedCvss": {
                "cvssSource": "NVD",
                "score": 7.5,
                "scoreSource": "NVD",
                "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N",
                "version": "3.1"
            }
        },
        "lastObservedAt": "Wed Sep 04 16:50:37.627 UTC 2024",
        "packageVulnerabilityDetails": {
            "cvss": [
                {
                    "baseScore": 7.5,
                    "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N",
                    "source": "NVD",
```

```
                    "version": "3.1"
                }
            ],
            "referenceUrls": [
                "https://www.debian.org/security/2023/dsa-5442",
                "https://lists.debian.org/debian-lts-announce/2023/08/msg00024.html"
            ],
            "relatedVulnerabilities": [],
            "source": "NVD",
            "sourceUrl": "https://nvd.nist.gov/vuln/detail/CVE-2023-30861",
            "vendorCreatedAt": "Tue May 02 18:15:52.000 UTC 2023",
            "vendorSeverity": "HIGH",
            "vendorUpdatedAt": "Sun Aug 20 21:15:09.000 UTC 2023",
            "vulnerabilityId": "CVE-2023-30861",
            "vulnerablePackages": [
                {
                    "epoch": 0,
                    "filePath": "requirements.txt",
                    "fixedInVersion": "2.3.2",
                    "name": "flask",
                    "packageManager": "PIP",
                    "version": "2.0.0"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "None Provided"
            }
        },
        "resources": [
            {
                "details": {
                    "awsLambdaFunction": {
                        "architectures": [
                            "X86_64"
                        ],
                        "codeSha256": "O7jkFEmfPB+CK3Y6Pby5zW9gjG
+zusAaqRRMGS8B27c=",
                        "executionRoleArn": "arn:aws:iam::123456789012:role/service-
role/VulnerableFunction-role-f9vs5mq8",
                        "functionName": "VulnerableFunction",
                        "lastModifiedAt": "Wed Sep 04 16:50:20.000 UTC 2024",
                        "packageType": "ZIP",
```

```
                        "runtime": "PYTHON_3_11",
                        "version": "$LATEST"
                    }
                },
                "id": "arn:aws:lambda:eu-
central-1:123456789012:function:VulnerableFunction:$LATEST",
                "partition": "aws",
                "region": "eu-central-1",
                "type": "AWS_LAMBDA_FUNCTION"
            }
        ],
        "severity": "HIGH",
        "status": "ACTIVE",
        "title": "CVE-2023-30861 - flask",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Wed Sep 04 16:50:37.627 UTC 2024"
    }
}
```

Lambda code vulnerability finding

```
{
    "version": "0",
    "id": "e764f7be-f931-ff1b-204b-8cab2d91724b",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-09-04T16:51:01Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:lambda:eu-central-1:123456789012:function:VulnerableFunction:
$LATEST"
    ],
    "detail": {
        "awsAccountId": "123456789012",
        "codeVulnerabilityDetails": {
            "cwes": [
                "CWE-798"
            ],
            "detectorId": "python/hardcoded-credentials@v1.0",
            "detectorName": "Hardcoded credentials",
```

```
            "detectorTags": [
                "secrets",
                "security",
                "owasp-top10",
                "top25-cwes",
                "cwe-798",
                "Python"
            ],
            "filePath": {
                "endLine": 6,
                "fileName": "lambda_function.py",
                "filePath": "lambda_function.py",
                "startLine": 6
            },
            "ruleId": "python-detect-hardcoded-aws-credentials"
        },
        "description": "Access credentials, such as passwords and access keys,
 should not be hardcoded in source code. Hardcoding credentials may cause leaks even
 after removing them. This is because version control systems might retain older
 versions of the code. Credentials should be stored securely and obtained from the
 runtime environment.",
        "findingArn": "arn:aws:inspector2:eu-
central-1:123456789012:finding/FINDING_ID",
        "firstObservedAt": "Wed Sep 04 16:51:01.869 UTC 2024",
        "lastObservedAt": "Wed Sep 04 16:51:01.869 UTC 2024",
        "remediation": {
            "recommendation": {
                "text": "Your code uses hardcoded AWS credentials which might
 allow unauthorized users access to your AWS account. These attacks can occur
 a long time after the credentials are removed from the code. We recommend that
 you set AWS credentials with environment variables or an AWS profile instead.
 You should consider deleting the affected account or rotating the secret key
 and then monitoring Amazon CloudWatch for unexpected activity.\n[https://
boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html](https://
boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html)"
            }
        },
        "resources": [
            {
                "details": {
                    "awsLambdaFunction": {
                        "architectures": [
                            "X86_64"
                        ],
```

```
                        "codeSha256": "O7jkFEmfPB+CK3Y6Pby5zW9gjG
+zusAaqRRMGS8B27c=",
                        "executionRoleArn": "arn:aws:iam::123456789012:role/service-
role/VulnerableFunction-role-f9vs5mq8",
                        "functionName": "VulnerableFunction",
                        "lastModifiedAt": "Wed Sep 04 16:50:20.000 UTC 2024",
                        "packageType": "ZIP",
                        "runtime": "PYTHON_3_11",
                        "version": "$LATEST"
                    }
                },
                "id": "arn:aws:lambda:eu-
central-1:123456789012:function:VulnerableFunction:$LATEST",
                "partition": "aws",
                "region": "eu-central-1",
                "type": "AWS_LAMBDA_FUNCTION"
            }
        ],
        "severity": "CRITICAL",
        "status": "ACTIVE",
        "title": "CWE-798 - Hardcoded credentials",
        "type": "CODE_VULNERABILITY",
        "updatedAt": "Wed Sep 04 16:51:01.869 UTC 2024"
    }
}
```

> **Note**
>
> 詳細値は、単一の検出結果の JSON の詳細をオブジェクトとして返します。配列内の複数の検出結果をサポートする検出結果レスポンスの構文全体は返されません。

Amazon Inspector の初回スキャン完了イベントスキーマの例

以下は、初回スキャンを完了するための Amazon Inspector イベントの EventBridge イベントスキーマの例です。このイベントは、Amazon Inspector がリソースの 1 つの初回スキャンを完了したときに作成されます。

以下のフィールドは初回スキャン完了イベントを識別します。

- `detail-type` フィールドは `Inspector2 Scan` に設定されます。
- この `detail` オブジェクトには、`CRITICAL`、`HIGH`、および `MEDIUM` など、該当する重要度カテゴリの検出結果の数を詳細に示す `finding-severity-counts` オブジェクトが含まれています。

オプションから選択すると、リソースタイプごとに異なる初回スキャンイベントスキーマが表示されます。

Amazon EC2 instance initial scan

```
{
    "version": "0",
    "id": "28a46762-6ac8-6cc4-4f55-bc9ab99af928",
    "detail-type": "Inspector2 Scan",
    "source": "aws.inspector2",
    "account": "111122223333",
    "time": "2023-01-20T22:52:35Z",
    "region": "us-east-1",
    "resources": [
        "i-087d63509b8c97098"
    ],
    "detail": {
        "scan-status": "INITIAL_SCAN_COMPLETE",
        "finding-severity-counts": {
            "CRITICAL": 0,
            "HIGH": 0,
            "MEDIUM": 0,
            "TOTAL": 0
        },
        "instance-id": "i-087d63509b8c97098",
        "version": "1.0"
    }
}
```

Amazon ECR image initial scan

```
{
    "version": "0",
```

```
    "id": "fdaa751a-984c-a709-44f9-9a9da9cd3606",
    "detail-type": "Inspector2 Scan",
    "source": "aws.inspector2",
    "account": "111122223333",
    "time": "2023-01-20T23:15:18Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecr:us-east-1:111122223333:repository/inspector2"
    ],
    "detail": {
        "scan-status": "INITIAL_SCAN_COMPLETE",
        "repository-name": "arn:aws:ecr:us-east-1:111122223333:repository/
inspector2",
        "finding-severity-counts": {
            "CRITICAL": 0,
            "HIGH": 0,
            "MEDIUM": 0,
            "TOTAL": 0
        },
        "image-digest":
 "sha256:965fbcae990b0467ed5657caceaec165018ef44a4d2d46c7cdea80a9dff0d1ea",
        "image-tags": [
            "ubuntu22"
        ],
        "version": "1.0"
    }
}
```

Lambda function initial scan

```
{
  "version": "0",
  "id": "4f290a7c-361b-c442-03c8-a629f6f20d6c",
  "detail-type": "Inspector2 Scan",
  "source": "aws.inspector2",
  "account": "111122223333",
  "time": "2023-02-23T18:06:03Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:lambda:us-west-2:111122223333:function:lambda-example:$LATEST"
```

```
  ],
  "detail": {
    "scan-status": "INITIAL_SCAN_COMPLETE",
    "finding-severity-counts": {
      "CRITICAL": 0,
      "HIGH": 0,
      "MEDIUM": 0,
      "TOTAL": 0
    },
    "version": "1.0"
  }
}
```

Amazon Inspector カバレッジイベントスキーマの例

以下は、カバレッジの Amazon Inspector イベントの EventBridge イベントスキーマの例です。このイベントは、リソースの Amazon Inspector スキャンカバレッジが変更されたときに作成されます。以下のフィールドはカバレッジイベントを識別します。

- `detail-type` フィールドは `Inspector2 Coverage` に設定されます。
- この `detail` オブジェクトには、リソースの新しいスキャンステータスを示す `scanStatus` オブジェクトが含まれています。

```
{
    "version": "0",
    "id": "000adda5-0fbf-913e-bc0e-10f0376412aa",
    "detail-type": "Inspector2 Coverage",
    "source": "aws.inspector2",
    "account": "111122223333",
    "time": "2023-01-20T22:51:39Z",
    "region": "us-east-1",
    "resources": [
        "i-087d63509b8c97098"
    ],
    "detail": {
        "scanStatus": {
            "reason": "UNMANAGED_EC2_INSTANCE",
```

```
            "statusCodeValue": "INACTIVE"
        },
        "scanType": "PACKAGE",
        "eventTimestamp": "2023-01-20T22:51:35.665501Z",
        "version": "1.0"
    }
}
```

Amazon Inspector 自動有効化スキーマの例

Amazon Inspector が組織内のメンバー数をサポートできない場合、自動有効化イベントは委任管理者に送信されます。次のフィールドは、自動有効化イベントを識別します。

- `detail-type` フィールドは `Inspector2 AutoEnable` に設定されます。
- `detail` オブジェクトは、自動有効化イベントが失敗した理由を説明します。

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "Inspector2 AutoEnable",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2024-08-21T02:36:48Z",
    "region": "us-east-1",
    "detail": {
        “version”: “1.0.0”,
        “AutoEnableStatus”: “Failed”,
        “Reason”: "The number of member accounts enabled with AWS Inspector has reached
 the maximum limit of 10,000"
        }
}
```

Linux および 用の Amazon Inspector SSM プラグイン Windows

このトピックでは、 Linuxおよび Windowsインスタンスの Amazon Inspector SSM プラグインについて説明します。

の Amazon Inspector SSM プラグイン Linux

Amazon Inspector はAmazon Inspector SSM プラグインを使用して Linux インスタンスでディープインスペクションスキャンを実行します。Amazon Inspector SSM プラグインは、 `/opt/aws/inspector/bin` ディレクトリの Linux インスタンスに自動的にインストールされます。実行可能ファイルの名前は `inspectorssmplugin` です。

Amazon Inspector は、Systems Manager Distributor を使用してインスタンスにプラグインをデプロイします。詳細検査スキャンを実行するには、Systems Manager Distributor と Amazon Inspectorが、Amazon EC2 インスタンスオペレーティングシステムをサポートしている必要があります。Systems Manager Distributor がサポートするオペレーティングシステムの詳細については、「AWS Systems Manager ユーザーガイド」の「サポートされているパッケージのプラットフォームとアーキテクチャ」を参照してください。

Amazon Inspector は、Amazon Inspector SSM プラグインによって詳細検査のために収集されたデータを管理するファイルディレクトリを作成します。これらのファイルディレクトリには、 `/opt/aws/inspector/var/input`と が含まれます`/opt/aws/inspector/var/output`。

の `packages.txt`ファイルは、ディープインスペクションが検出したパッケージへのフルパス`/opt/aws/inspector/var/output`を保存します。Amazon Inspector がインスタンスで同じパッケージを複数回検出した場合、`packages.txt` ファイルにはそのパッケージが見つかった各場所が一覧表示されます。

Amazon Inspector は、プラグインのログを `/var/log/amazon/inspector` ディレクトリに保存します。

Amazon Inspector SSM プラグインのアンインストール

`inspectorssmplugin` ファイルが誤って削除された場合、SSM 関連付け `InspectorLinuxDistributor-do-not-delete` は次のスキャン間隔で `inspectorssmplugin` ファイルの再インストールを試みます。

Amazon EC2 スキャンを非アクティブ化すると、プラグインはすべての Linux ホストから自動的にアンインストールされます。

の Amazon Inspector SSM プラグイン Windows

Amazon Inspector SSM プラグインは、Amazon Inspector が Windows インスタンスをスキャンするために必要です。Amazon Inspector SSM プラグインは `C:\Program Files\Amazon\Inspector` の Windows インスタンスに自動的にインストールされ、実行可能バイナリファイルは `InspectorSsmPlugin.exe` という名前になります。

Amazon Inspector SSM プラグインが収集するデータを保存するために、次のファイルの場所が作成されます。

- `C:\ProgramData\Amazon\Inspector\Input`
- `C:\ProgramData\Amazon\Inspector\Output`
- `C:\ProgramData\Amazon\Inspector\Logs`

> **Note**
>
> デフォルトでは、Amazon Inspector SSM プラグインは通常の優先度よりも低い優先度で実行されます。

> **Note**
>
> Windows インスタンスは、デフォルトのホスト管理設定で使用できます。ただし、`ssm:PutInventory` および `ssm:GetParameter` アクセス許可で設定されたロールを作成または使用する必要があります。

Amazon Inspector SSM プラグインのアンインストール

`InspectorSsmPlugin.exe` ファイルが誤って削除された場合、`InspectorDistributor-do-not-delete`関連付けは次のWindowsスキャン間隔で`InspectorSsmPlugin.exe`ファイルを再インストールします。Amazon Inspector SSM プラグインをアンインストールする場合は、`AmazonInspector2-ConfigureInspectorSsmPlugin`ドキュメントで Uninstall アクション

を使用できます。ただし、Amazon Inspector SSM プラグインはすべてのWindowsホストから自動的にアンインストールされます。 Amazon EC2

> **Note**
>
> Amazon Inspector を非アクティブ化する前に SSM エージェントをアンインストールした場合、Amazon Inspector SSM プラグインはWindowsホストに残りますが、Amazon Inspector SSM プラグインにデータを送信しません。詳細については、「Amazon Inspector の非アクティブ化」を参照してください。

Amazon Inspector SBOM Generator

ソフトウェア部品表 (SBOM) は、ソフトウェアの構築に必要なコンポーネント、ライブラリ、モジュールの正式に構造化されたリストです。Amazon Inspector SBOM Generator (Sbomgen) は、アーカイブ、コンテナイメージ、ディレクトリ、ローカルシステム、コンパイル済み Go および Rust バイナリ用の SBOM を生成するツールです。Sbomgen は、インストールされたパッケージに関する情報を含むファイルをスキャンします。Sbomgen は、関連ファイルを見つけると、パッケージ名、バージョン、およびその他のメタデータを抽出します。その後、Sbomgen はパッケージメタデータを CycloneDX SBOM に変換します。Sbomgen を使用して CycloneDX SBOM をファイルとして生成または STDOUT で生成したり、脆弱性検出のために SBOM を Amazon Inspector に送信したりできます。また、デプロイパイプラインの一部としてコンテナイメージを自動的にスキャンする CI/CD 統合の一部として、Sbomgen を使用できます。

サポートされているパッケージタイプ

Sbomgen は、次のパッケージタイプのインベントリを収集します。

- Alpine APK
- Debian/Ubuntu DPKG
- Red Hat RPM
- C#
- Go
- Java
- Node.js
- PHP
- Python
- Ruby
- Rust

サポートされているコンテナイメージ設定チェック

Sbomgen は、スタンドアロンの Dockerfile をスキャンし、既存のイメージからセキュリティ上の問題がないか履歴を構築できます。詳細については、「Amazon Inspector Dockerfile チェック」を参照してください。

Sbomgen のインストール

Sbomgen は、Linux オペレーティングシステムでのみ利用できます。

Sbomgen でローカルにキャッシュされたイメージを分析する場合は、Docker がインストールされている必要があります。リモートコンテナレジストリでホストされる `.tar` ファイルまたはイメージとしてエクスポートされたイメージを分析するために、Docker は必要はありません。

Amazon Inspector では、少なくとも次のハードウェア仕様のシステムから Sbomgen を実行することをお勧めします。

- 4x Core CPU
- 8 GB RAM

Sbomgen をインストールするには

1. 使用しているアーキテクチャの正しい URL から最新の Sbomgen zip ファイルをダウンロードします。

   Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/latest/linux/amd64/inspector-sbomgen.zip

   Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/latest/linux/arm64/inspector-sbomgen.zip

   または、以前のバージョンの Amazon Inspector SBOM Generator の zip ファイルをダウンロードすることもできます。
2. 次のコマンドを使用して、ダウンロードした zip ファイルを解凍します。

   ```
   unzip inspector-sbomgen.zip
   ```
3. 抽出されたディレクトリで次のファイルを確認します。

   - `inspector-sbomgen` – SBOM を生成するために実行するツールです。
   - `README.txt` — Sbomgen の使用に関するドキュメントです。
   - `LICENSE.txt` — このファイルには、Sbomgen のソフトウェアライセンスが含まれています。
   - `licenses` — このフォルダには、Sbomgen が使用するサードパーティパッケージのライセンス情報が含まれています。

- `checksums.txt` – このファイルは Sbomgen ツールのハッシュを提供します。
- `sbom.json` – Sbomgen ツールの CycloneDX SBOM です。
- `WhatsNew.txt` – このファイルには、要約された変更ログが含まれているため、Sbomgen バージョン間の大きな変更や改善点をすばやく確認できます。

4. (オプション) 以下のコマンドを使用して、ツールの信頼性と整合性を検証します。

```
sha256sum < inspector-sbomgen
```

- 結果を `checksums.txt` ファイルの内容と比較します。

5. 以下のコマンドを使用して、ツールに実行権限を付与します。

```
chmod +x inspector-sbomgen
```

6. 以下のコマンドを実行して、Sbomgen が正常にインストールされたことを確認します。

```
./inspector-sbomgen --version
```

次のような出力が表示されます。

```
Version: 1.X.X
```

Sbomgen を使用する

このセクションでは、Sbomgen を使用するさまざまな方法を示します。Sbomgen の使用方法の詳細については、組み込みの例を参照してください。これらの例を表示するには、`list-examples` コマンドを実行します。

```
./inspector-sbomgen list-examples
```

コンテナイメージの SBOM の生成と結果の出力

コンテナイメージの SBOM を生成し、結果をファイルに出力するには、Sbomgen を使用できます。この機能は、`container` サブコマンドを使用して有効にできます。

コマンドの例

次のスニペットでは、*`image:tag`* をイメージの ID に、*`output_path.json`* を保存する出力へのパスに置き換えることができます。

```
# generate SBOM for container image
./inspector-sbomgen container --image image:tag -o output_path.json
```

> **Note**
>
> スキャン時間とパフォーマンスは、イメージサイズと、レイヤーの数の少なさによって異なります。イメージを小さくすると、Sbomgen のパフォーマンスが向上するだけでなく、潜在的なアタックサーフェスも減少します。イメージを小さくすると、イメージの構築、ダウンロード、アップロードの時間も短縮されます。

Sbomgen で を使用する場合ScanSbom、Amazon Inspector スキャン API は 5,000 個を超えるパッケージを含む SBOMs を処理しません。このシナリオでは、Amazon Inspector スキャン API は HTTP 400 レスポンスを返します。

イメージにバルクメディアファイルまたはディレクトリが含まれている場合は、`--skip-files` 引数を使用して Sbomgen からそれらを除外することを検討してください。

例: 一般的なエラーケース

コンテナイメージのスキャンは、次のエラーが原因で失敗することがあります。

- `InvalidImageFormat` – 不正な TAR ヘッダー、マニフェストファイル、または設定ファイルを使用して不正な形式のコンテナイメージをスキャンするときに発生します。
- `ImageValidationFailure` – Content-Length ヘッダーの不一致、マニフェストダイジェストの誤り、SHA256 チェックサムの検証の失敗など、コンテナイメージコンポーネントのチェックサムまたはコンテンツの長さの検証が失敗した場合に発生します。
- `ErrUnsupportedMediaType` – イメージコンポーネントにサポートされていないメディアタイプが含まれている場合に発生します。サポートされているメディアタイプの詳細については、「サポートされているオペレーティングシステムとメディアタイプ」を参照してください。

Amazon Inspector は`application/vnd.docker.distribution.manifest.list.v2+json`メディアタイプをサポートしていません。ただし、Amazon Inspector はマニフェストリストをサポートしています。マニフェストリストを使用するイメージをスキャンする場合、 `--platform`引数で使用するプラットフォームを明示的に指定できます。`--platform` 引数を指定しない場合、Amazon Inspector SBOM Generator は実行中のプラットフォームに基づいてマニフェストを自動的に選択します。

ディレクトリとアーカイブからの SBOM の生成

Sbomgen を使用して、ディレクトリとアーカイブから SBOM を生成できます。この機能は、`directory` または `archive` サブコマンドを使用して有効にできます。Amazon Inspector では、ダウンロードした git リポジトリなどのプロジェクトフォルダから SBOM を生成するときに、この機能を使用することをお勧めしています。

コマンドの例 1

次のスニペットは、ディレクトリファイルから SBOM を生成するサブコマンドを示しています。

```
# generate SBOM from directory
./inspector-sbomgen directory --path /path/to/dir -o /tmp/sbom.json
```

コマンドの例 2

次のスニペットは、アーカイブファイルから SBOM を生成するサブコマンドを示しています。現在サポートされているアーカイブ形式は `.zip`、`.tar`、および `.tar.gz` です。

```
# generate SBOM from archive file (tar, tar.gz, and zip formats only)
./inspector-sbomgen archive --path testData.zip -o /tmp/sbom.json
```

Go または Rust のコンパイル済みバイナリからの SBOM の生成

Sbomgen を使用して、コンパイル済み Go および Rust バイナリから SBOM を生成できます。この機能は、`binary` サブコマンドを使用して有効にできます。

```
./inspector-sbomgen binary --path /path/to/your/binary
```

マウントされたボリュームから SBOM を生成する

Amazon Inspector SBOM Generator を使用して、マウントされたボリュームから SBOMs を生成できます。この機能は、`volume` サブコマンドを使用して有効にできます。この機能は、システムにマウントされた Amazon EBS ボリュームなどのストレージボリュームを分析するときに使用することをお勧めします。ディレクトリサブコマンドとは異なり、マウントされたボリュームスキャンは OS パッケージと OS 情報を検出します。

Amazon EBS ボリュームをスキャンするには、Amazon Inspector SBOM Generator がインストールされている Amazon EC2 インスタンスにアタッチし、そのインスタンスにマウントします。

Amazon Inspector 他の Amazon EC2 インスタンスで現在使用されている Amazon EBS ボリュームの場合、ボリュームの Amazon EBS スナップショットを作成し、スキャン目的でそのスナップショットから新しい Amazon EBS ボリュームを作成できます。Amazon EBS の詳細については、「Amazon Elastic Block Store ユーザーガイド」の「Amazon EBS とは」を参照してください。

コマンドの例

次のスニペットは、マウントされたボリュームから SBOM を生成するサブコマンドを示しています。`--path` 引数は、ボリュームがマウントされるルートディレクトリを指定する必要があります。

```
# generate SBOM from mounted volume
./inspector-sbomgen volume --path /mount/point/of/volume/root
```

コマンドの例

次のスニペットは、 `--exclude-suffix`引数を持つ特定のファイルパスを除外しながら、マウントされたボリュームから SBOM を生成するサブコマンドを示しています。`--exclude-suffix` 引数は、ボリュームにバルクファイル (ログファイルやメディアファイルなど) が含まれている場合に特に便利です。パスが指定されたサフィックスで終わるファイルとディレクトリはスキャンから除外されるため、スキャン時間とメモリ使用量を減らすことができます。

```
# generate SBOM from mounted volume with exclusions
./inspector-sbomgen volume --path /mount/point/of/volume/root \
--exclude-suffix .log \
--exclude-suffix cache
```

ターゲットボリューム内のすべてのファイルパスは、元のパスに正規化されます。たとえば、 のファイルを含む にマウント`/mnt/volume`されたボリュームをスキャンすると`/mnt/volume/var/lib/rpm/rpmdb.sqlite`、パスは生成された SBOM `/var/lib/rpm/rpmdb.sqlite`で に正規化されます。

脆弱性特定のための SBOM の Amazon Inspector への送信

SBOM の生成に加えて、Amazon Inspector スキャン API から 1 つのコマンドを使用してスキャンのために SBOM を送信できます。Amazon Inspector は、検出結果を Sbomgen に返す前に、SBOM の内容を評価して脆弱性がないかを確認します。入力に応じて、検出結果を表示またはファイルに書き込むことができます。

> **Note**
>
> この機能`InspectorScan-ScanSbom`を使用するには、 への読み取りアクセス許可 AWS アカウント を持つアクティブな が必要です。

この機能を有効にするには、`--scan-sbom` 引数を Sbomgen CLI に渡します。`--scan-sbom` 引数は、`archive`、`binary`、`container`、`directory`、`localhost` のいずれかの Sbomgen サブコマンドに渡すこともできます。

> **Note**
>
> Amazon Inspector スキャン API は、2,000 を超えるパッケージを含む SBOM を処理しません。このシナリオでは、Amazon Inspector スキャン API は HTTP 400 レスポンスを返します。

AWS プロファイルまたは IAM ロールを使用して、次の AWS CLI 引数を使用して Amazon Inspector を認証できます。

```
--aws-profile profile
--aws-region region
--aws-iam-role-arn role_arn
```

Sbomgen に次の環境変数を指定することで、Amazon Inspector に対して認証を行うこともできます。

```
AWS_ACCESS_KEY_ID=$access_key \
AWS_SECRET_ACCESS_KEY=$secret_key \
AWS_DEFAULT_REGION=$region \
./inspector-sbomgen arguments
```

レスポンス形式を指定するには、`--scan-sbom-output-format cyclonedx` 引数または `--scan-sbom-output-format inspector` 引数を使用します。

コマンドの例 1

このコマンドは、最新の Alpine Linux リリースの SBOM を作成し、その SBOM をスキャンして、脆弱性の結果を JSON ファイルに書き込みます。

```
./inspector-sbomgen container --image alpine:latest \
                          --scan-sbom \
                          --aws-profile your_profile \
                          --aws-region your_region \
                          --scan-sbom-output-format cyclonedx \
                          --outfile /tmp/inspector_scan.json
```

コマンドの例 2

このコマンドは、環境変数として AWS 認証情報を使用して Amazon Inspector を認証します。

```
AWS_ACCESS_KEY_ID=$your_access_key \
AWS_SECRET_ACCESS_KEY=$your_secret_key \
AWS_DEFAULT_REGION=$your_region \
./inspector-sbomgen container --image alpine:latest \
                          -o /tmp/sbom.json \
                          --scan-sbom \
                          --scan-sbom-output-format inspector
```

コマンドの例 3

このコマンドは、IAM ロールの ARN を使用して Amazon Inspector に対して認証を行います。

```
./inspector-sbomgen container --image alpine:latest \
                          --scan-sbom \
                          --aws-profile your_profile \
                          --aws-region your_region \
                          --outfile /tmp/inspector_scan.json
                          --aws-iam-role-arn arn:aws:iam::123456789012:role/your_role
```

追加のスキャナーを使用して検出機能を強化する

Amazon Inspector SBOM Generator は、使用されているコマンドに基づいて事前定義されたスキャナーを適用します。

デフォルトのスキャナーグループ

各 Amazon Inspector SBOM Generator サブコマンドは、次のデフォルトのスキャナーグループを自動的に適用します。

- `directory` サブコマンドの場合: binary、programming-language-packages、dockerfile スキャナーグループ
- `localhost` サブコマンドの場合: os、programming-language-packages、extra-ecosystems スキャナーグループ
- `container` サブコマンドの場合: os、programming-language-packages、extra-ecosystems、dockerfile、バイナリスキャナーグループ

特殊スキャナー

デフォルトのスキャナーグループ以外のスキャナーを含めるには、 `--additional-scanners`オプションに続いて追加するスキャナーの名前を使用します。これを行う方法を示すコマンドの例を次に示します。

```
# Add WordPress installation scanner to directory scan
./inspector-sbomgen directory --path /path/to/directory/ --additional-scanners
 wordpress-installation -o output.json
```

以下は、カンマ区切りのリストで複数のスキャナーを追加する方法を示すコマンドの例です。

```
./inspector-sbomgen container --image image:tag --additional-scanners scanner1,scanner2
 -o output.json
```

スキャンする最大ファイルサイズを調整してコンテナスキャンを最適化する

コンテナイメージを分析して処理すると、 はデフォルトで 200 MB 以下のファイルをSbomgen スキャンします。200 MB を超えるファイルには、パッケージメタデータはほとんど含まれません。200MB を超える Goまたは Rustバイナリをインベントリすると、ミスが発生する可能性があります。サイズ制限を調整するには、 `--max-file-size`引数を使用します。これにより、大きなファイルを含めるように制限を増やし、大きなファイルを除外してリソース使用量を減らすことができます。

例

次の例は、 `--max-file-size`引数を使用してファイルサイズを増やす方法を示しています。

```
# Increase the file size limit to scan files up to 300 MB
./inspector-sbomgen container --image alpine:latest \
--outfile /tmp/sbom.json \
--max-file-size 300000000
```

この設定を調整すると、ディスク使用量、メモリ消費量、全体的なスキャン期間を制御できます。

進行状況インジケータの無効化

Sbomgen は、回転する進行状況インジケータを表示しますが、それにより CI/CD 環境で過剰なスラッシュ文字が発生する可能性があります。

```
INFO[2024-02-01 14:58:46]coreV1.go:53: analyzing artifact
|
\
/
|
\
/
INFO[2024-02-01 14:58:46]coreV1.go:62: executing post-processors
```

`--disable-progress-bar` 引数を使用すると、この進行状況インジケータを無効にすることができます。

```
./inspector-sbomgen container --image alpine:latest \
--outfile /tmp/sbom.json \
--disable-progress-bar
```

Sbomgen を使用したプライベートレジストリへの認証

プライベートレジストリの認証情報を指定することで、プライベートレジストリでホストされているコンテナから SBOM を生成できます。これらの認証情報は、次のメソッドで提供できます。

キャッシュされた認証情報を使用した認証 (推奨)

この方法では、コンテナレジストリに対して認証を行います。例えば、Docker を使用している場合は、以下の Docker ログ記録コマンドを使用してコンテナレジストリに対して認証を行うことができます: `docker login`。

1. コンテナレジストリに対して認証を行います。例えば、Docker を使用している場合は、以下の Docker `login` コマンドを使用してレジストリに対して認証を行うことができます。
2. コンテナレジストリに対して認証を行ったら、レジストリにあるコンテナイメージで Sbomgen を使用します。以下の例を使用するには、*`image:tag`* を、スキャンするイメージの名前に置き換えてください。

```
./inspector-sbomgen container --image image:tag
```

インタラクティブメソッドを使用した認証

この方法では、ユーザー名をパラメータとして指定すると、Sbomgen から必要に応じて安全なパスワードの入力を求められます。

以下の例を使用するには、*`image:tag`* をスキャンするイメージの名前に置き換え、*`your_username`* をその画像にアクセスできるユーザー名に置き換えます。

```
./inspector-sbomgen container --image image:tag --username your_username
```

非インタラクティブなメソッドを使用した認証

このメソッドでは、パスワードまたはレジストリトークンを `.txt` ファイルに保存します。

> **Note**
>
> 現在のユーザーは、このファイルの読み取りのみが可能です。また、ファイルには、パスワードまたはトークンが 1 行にまとめられます。

以下の例を使用するには、*`your_username`* をユーザー名に置き換え、*`password.txt`* をパスワードまたはトークンを 1 行に含む `.txt` ファイルに置き換え、*`image:tag`* をスキャンするイメージの名前に置き換えます。

```
INSPECTOR_SBOMGEN_USERNAME=your_username \
INSPECTOR_SBOMGEN_PASSWORD=`cat password.txt` \
./inspector-sbomgen container --image image:tag
```

Sbomgen からの出力例

Sbomgen を使用してインベントリが作成されたコンテナイメージの SBOM の例を以下に示します。

コンテナイメージの SBOM


```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.5",
  "serialNumber": "urn:uuid:828875ef-8c32-4777-b688-0af96f3cf619",
  "version": 1,
  "metadata": {
    "timestamp": "2023-11-17T21:36:38Z",
    "tools": [
      {
        "vendor": "Amazon Web Services, Inc. (AWS)",
        "name": "Amazon Inspector SBOM Generator",
        "version": "1.0.0",
        "hashes": [
          {
            "alg": "SHA-256",
            "content":
 "10ab669cfc99774786301a745165b5957c92ed9562d19972fbf344d4393b5eb1"
          }
        ]
      }
    ],
    "component": {
      "bom-ref": "comp-1",
      "type": "container",
      "name": "fedora:latest",
      "properties": [
        {
          "name": "amazon:inspector:sbom_generator:image_id",
          "value":
 "sha256:c81c8ae4dda7dedc0711daefe4076d33a88a69a28c398688090c1141eff17e50"
        },
        {
          "name": "amazon:inspector:sbom_generator:layer_diff_id",
          "value":
 "sha256:eddd0d48c295dc168d0710f70364581bd84b1dda6bb386c4a4de0b61de2f2119"
        }
```

```
      ]
    }
  },
  "components": [
    {
      "bom-ref": "comp-2",
      "type": "library",
      "name": "dnf",
      "version": "4.18.0",
      "purl": "pkg:pypi/dnf@4.18.0",
      "properties": [
        {
          "name": "amazon:inspector:sbom_generator:source_file_scanner",
          "value": "python-pkg"
        },
        {
          "name": "amazon:inspector:sbom_generator:source_package_collector",
          "value": "python-pkg"
        },
        {
          "name": "amazon:inspector:sbom_generator:source_path",
          "value": "/usr/lib/python3.12/site-packages/dnf-4.18.0.dist-info/METADATA"
        },
        {
          "name": "amazon:inspector:sbom_generator:is_duplicate_package",
          "value": "true"
        },
        {
          "name": "amazon:inspector:sbom_generator:duplicate_purl",
          "value": "pkg:rpm/fedora/python3-dnf@4.18.0-2.fc39?
arch=noarch&distro=39&epoch=0"
        }
      ]
    },
    {
      "bom-ref": "comp-3",
      "type": "library",
      "name": "libcomps",
      "version": "0.1.20",
      "purl": "pkg:pypi/libcomps@0.1.20",
      "properties": [
        {
          "name": "amazon:inspector:sbom_generator:source_file_scanner",
          "value": "python-pkg"
```

```
        },
        {
          "name": "amazon:inspector:sbom_generator:source_package_collector",
          "value": "python-pkg"
        },
        {
          "name": "amazon:inspector:sbom_generator:source_path",
          "value": "/usr/lib64/python3.12/site-packages/libcomps-0.1.20-py3.12.egg-
info/PKG-INFO"
        },
        {
          "name": "amazon:inspector:sbom_generator:is_duplicate_package",
          "value": "true"
        },
        {
          "name": "amazon:inspector:sbom_generator:duplicate_purl",
          "value": "pkg:rpm/fedora/python3-libcomps@0.1.20-1.fc39?
arch=x86_64&distro=39&epoch=0"
        }
      ]
    }
  ]
}
```

Amazon Inspector SBOM Generator の以前のバージョン

このトピックでは、Amazon Inspector SBOM Generator の最新バージョンと以前のバージョンへのリンクを提供します。Sbomgen のインストールについては、「Sbomgen のインストール」を参照してください。

最新バージョン

- https://amazon-inspector-sbomgen.s3.amazonaws.com/latest/linux/amd64/inspector-sbomgen.zip
- https://amazon-inspector-sbomgen.s3.amazonaws.com/latest/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.8.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.8.0/linux/amd64/inspector-sbomgen.zip

- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.8.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.7.3

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.3/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.3/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.7.2

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.2/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.2/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.7.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.7.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.7.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.6.3

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.3/linux/amd64/inspector-sbomgen.zip

- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.3/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.6.2

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.2/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.2/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.6.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.6.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.6.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.5

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.5/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.5/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.4

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.4/linux/amd64/inspector-sbomgen.zip

- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.4/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.3

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.3/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.3/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.2

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.2/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.2/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.5.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.5.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.4.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.4.0/linux/amd64/inspector-sbomgen.zip

- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.4.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.3.2

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.2/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.2/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.3.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.3.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.3.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.2.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.2.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.2.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.2.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.2.0/linux/amd64/inspector-sbomgen.zip

- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.2.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.1.1

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.1.1/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.1.1/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.1.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.1.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.1.0/linux/arm64/inspector-sbomgen.zip

Sbomgen 1.0.0

- Linux AMD64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.0.0/linux/amd64/inspector-sbomgen.zip
- Linux ARM64: https://amazon-inspector-sbomgen.s3.amazonaws.com/1.0.0/linux/arm64/inspector-sbomgen.zip

Amazon Inspector SBOM Generator の包括的なオペレーティングシステムコレクション

Amazon Inspector SBOM Generator は、さまざまなオペレーティングシステムをスキャンして、システムコンポーネントの堅牢で詳細な分析を保証します。SBOM を生成すると、オペレーティングシステムの構成を把握できるため、システムマネージドパッケージの脆弱性を特定できます。このトピックでは、Amazon Inspector SBOM Generator がサポートするさまざまなオペレーティングシステムパッケージコレクションの主な機能について説明します。Amazon Inspector がサポートするオペレーティングシステムの詳細については、Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。

サポートされているオペレーティングシステムアーティファクト

Amazon Inspector SBOM Generator は、次のオペレーティングシステムアーティファクトをサポートしています。

プラットフォーム	バイナリ	ソース	ストリーム
Alma Linux	該当なし	はい	あり
Alpine Linux	あり	はい	該当なし
Amazon Linux	該当なし	はい	該当なし
CentOS	該当なし	はい	該当なし
Chainguard	はい	はい	該当なし
Debian	はい	はい	該当なし
Distroless	はい	はい	該当なし
Fedora	該当なし	はい	該当なし
MinimOS	はい	はい	該当なし
OpenSUSE	該当なし	はい	該当なし
Oracle Linux	該当なし	はい	該当なし
Photon OS	該当なし	はい	該当なし
RHEL	該当なし	はい	はい
Rocky Linux	該当なし	はい	はい
SLES	該当なし	はい	該当なし
Ubuntu	はい	はい	該当なし

APK ベースの OS パッケージコレクション

このセクションでは、 APKベースの OS パッケージコレクションでサポートされているプラットフォームと主要な機能について説明します。詳細については、 Alpine Linuxウェブサイトの「Alpine Package Keeper」を参照してください。

サポートされているプラットフォーム

サポートされているプラットフォームは次のとおりです。

- Alpine Linux

> **Note**
>
> APKベースのシステムでは、Amazon Inspector SBOM Generator は `/lib/apk/db/` ファイルからパッケージメタデータを収集します。

主な特徴

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされた各パッケージのバージョンを抽出します。
- ソースパッケージの識別 – インストールされた各パッケージのソースパッケージを識別します

例

次のスニペットは、 APK データベースファイルの例です。

```
C:Q1JlboSJkrN4qkDcokr4zenpcWEXQ=
P:zlib
V:1.2.13-r1
A:x86_64
S:54253
I:110592
T:A compression/decompression Library
U:https://zlib.net/
L:Zlib
```

```
o:zlib
```

DPKG ベースの OS パッケージコレクション

このセクションでは、 DPKGベースの OS パッケージコレクションでサポートされているプラットフォームと主要な機能について説明します。詳細については、 Debianウェブサイトの「Debian パッケージ」を参照してください。

サポートされているプラットフォーム

以下のプラットフォームがサポートされています。

- Debian
- Ubuntu

> **Note**
>
> DPKGベースのシステムでは、Amazon Inspector SBOM Generator は `/var/lib/dpkg/status` ファイルからパッケージメタデータを収集します。

主な特徴

DPKGベースの OS パッケージの主な機能を次に示します。

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされている各パッケージのバージョンを抽出します。
- ソースパッケージの識別 – インストールされた各パッケージのソースパッケージを識別します

例

次のスニペットは、 `/var/lib/dpkg/` ファイルの例です。

```
Package: zlib1g
Status: install ok installed
```

```
Priority: optional
Section: libs
Installed-Size: 168
Maintainer: Mark Brown <broonie@debian.org>
Architecture: amd64
Multi-Arch: same
Source: zlib
Version: 1:1.2.13.dfsg-1
Provides: libz1
Depends: libc6 (>= 2.14)
Breaks: libxml2 (<< 2.7.6.dfsg-2), texlive-binaries (<< 2009-12)
Conflicts: zlib1 (<= 1:1.0.4-7)
Description: compression library - runtime
 zlib is a library implementing the deflate compression method found
 in gzip and PKZIP.  This package includes the shared library.
Homepage: http://zlib.net/
```

RPM ベースの OS パッケージコレクション

このセクションでは、 RPMベースの OS パッケージコレクションでサポートされているプラットフォームと主要な機能について説明します。詳細については、 RPMウェブサイトの「RPM Package Manager」を参照してください。

サポートされているプラットフォーム

以下のプラットフォームがサポートされています。

- Alma Linux
- Amazon Linux
- CentOS
- Fedora
- OpenSUSE
- Oracle Linux
- PhotonOS
- RedHat Enterprise Linux
- Rocky Linux
- SUSE Linux Enterprise Server

> **Note**
>
> RPMベースのシステムでは、Amazon Inspector SBOM Generator は `/var/lib/rpm` ファイルからパッケージメタデータを収集します。

主な特徴

RPMベースの OS パッケージコレクションの主な機能を次に示します。

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされた各パッケージのバージョンを抽出します。
- ソースパッケージの識別 – インストールされた各パッケージのソースパッケージを識別します
- ストリームサポート – インストールされた各パッケージのストリームメタデータを抽出します

例

データベースRPMファイルスニペットの例を次に示します。

```
/usr/lib/sysimage/rpm/rpmdb.sqlite
/usr/lib/sysimage/rpm/Packages
/usr/lib/sysimage/rpm/Packages.db
/var/lib/rpm/rpmdb.sqlite
/var/lib/rpm/Packages
/var/lib/rpm/Packages.db
```

Chainguard イメージパッケージコレクション

このセクションでは、Chainguardイメージパッケージコレクションでサポートされているプラットフォームと主要な機能について説明します。詳細については、 Chainguardウェブサイトの「イメージ」を参照してください。

サポートされているプラットフォーム

次のプラットフォームがサポートされています。

- Wolfi Linux

> **Note**
>
> Chainguard イメージの場合、Amazon Inspector SBOM Generator は`/lib/apk/db/installed`ファイルからパッケージメタデータを収集します。

主な特徴

以下は主要な機能です。

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされた各パッケージのバージョンを抽出します
- ソースパッケージの識別 – インストールされた各パッケージのソースパッケージを識別します

例

次のスニペットは、Chainguardイメージファイルの例です。

```
P:wolfi-keys
V:1-r8
A:x86_64
L:MIT
T:Wolfi signing keyring
o:wolfi-keys
```

Distroless イメージパッケージコレクション

Distroless コンテナは、ディスLinuxトリビューション内のパッケージマネージャー、シェル、その他のユーティリティを除外するコンテナイメージです。 Distrolessコンテナには、アプリケーションを実行し、パフォーマンスとセキュリティを向上させるために必要な必須の依存関係のみが含まれます。

> **Note**
>
> Distroless イメージの場合、Amazon Inspector SBOM Generator は`/var/lib/dpkg/status.d`ファイルからパッケージメタデータを収集します。Debian および Ubuntuベース

のディストリビューションのみがサポートされています。これらは、Debian「」または「」を示す`/etc/os-release`ファイルシステムの NAMEフィールドで識別できますUbuntu。

主な特徴

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされている各パッケージのバージョンを抽出します。

例

Distroless イメージファイルの例を次に示します。

```
Package: tzdata
Version: 2021a-1+deb11u10
Architecture: all
Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>
Installed-Size: 3413
Depends: debconf (>= 0.5) | debconf-2.0
Provides: tzdata-bullseye
Section: localization
Priority: required
Multi-Arch: foreign
Homepage: https://www.iana.org/time-zones
Description: time zone and daylight-saving time data
 This package contains data required for the implementation of
 standard local time for many representative locations around the
 globe. It is updated periodically to reflect changes made by
 political bodies to time zone boundaries, UTC offsets, and
 daylight-saving rules.
```

MinimOS パッケージコレクション

このセクションでは、Minimusイメージパッケージコレクションでサポートされているプラットフォームと主要な機能について説明します。詳細については、ミニムスのウェブサイトを参照してください。

サポートされているプラットフォーム

以下のプラットフォームがサポートされています。

- MinimOS

> **Note**
>
> Minimus イメージの場合、Amazon Inspector SBOM Generator は `/lib/apk/db/installed` ファイルからパッケージメタデータを収集します。

主な特徴

以下は主要な機能です。

- パッケージ名のコレクション – インストールされた各パッケージの名前を抽出します。
- バージョンコレクション – インストールされている各パッケージの名前を抽出します。
- ソースパッケージの識別 – インストールされた各パッケージのソースパッケージを識別します

以下は、Minimusイメージファイルのスニペットです。

```
P:ca-certificates-bundle
V:20241121-r1
A:aarch64
L:MPL-2.0 AND MIT
T:
o:ca-certificates
```

プログラミング言語の依存関係の収集

Amazon Inspector SBOM Generator は、さまざまなプログラミング言語とフレームワークをサポートし、堅牢で詳細な依存関係のコレクションを構成します。SBOM を生成すると、ソフトウェアの構成を把握できるため、脆弱性を特定し、セキュリティ標準への準拠を維持できます。Amazon Inspector SBOM Generator は、次のプログラミング言語とファイル形式をサポートしています。

Go 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Go	Go	go.mod go.sum Go Binaries GOMODCACHE	該当なし 該当なし はい 該当なし	該当なし 該当なし 該当なし 該当なし	該当なし 該当なし 該当なし 該当なし	該当なし 該当なし 該当なし 該当なし	はい あり はい いいえ

go.mod/go.sum

go.mod および go.sum ファイルを使用して、Goプロジェクトの依存関係を定義およびロックします。Amazon Inspector SBOM Generator は、Goツールチェーンのバージョンに基づいてこれらのファイルを異なる方法で管理します。

主な特徴

- から依存関係を収集します go.mod (Goツールチェーンバージョンが 1.17 以降の場合)
- から依存関係を収集します go.sum (Goツールチェーンバージョンが 1.17 以前の場合)
- 宣言されたすべての依存関係と依存関係バージョンを識別go.modするための解析

go.mod ファイルの例

go.mod ファイルの例を次に示します。

```
module example.com/project

go 1.17

require (
github.com/gin-gonic/gin v1.7.2
golang.org/x/crypto v0.0.0-20210616213533-5cf6c0f8e123
)
```

go.sum ファイルの例

go.sum ファイルの例を次に示します。

```
github.com/gin-gonic/gin v1.7.2 h1:VZ7DdRl0sghbA6lVGSkX+UXO2+J0aH7RbsNugG+FA8Q=
github.com/gin-gonic/gin v1.7.2/go.mod h1:ILZ1Ngh2f1pL1ASUj7gGk8lGFeNC8cRTaN2ZhsBNbXU=
golang.org/x/crypto v0.0.0-20210616213533-5cf6c0f8e123 h1:b6rCu+qHze
+BUsmC3CZzH8aNu8LzPZTVsNTo64OypSc=
golang.org/x/crypto v0.0.0-20210616213533-5cf6c0f8e123/go.mod h1:K5Dkpb0Q4ewZW/
EzWlQphgJcUMBCzoWrLfDOVzpTGVQ=
```

> **Note**
>
> これらのファイルはそれぞれ、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Go バイナリ

Amazon Inspector SBOM Generator は、コンパイルされたGoバイナリから依存関係を抽出して、使用中のコードについて保証します。

> **Note**
>
> Amazon Inspector SBOM Generator は、公式Goコンパイラを使用して構築されたGoバイナリからのツールチェーンバージョンのキャプチャと評価をサポートしています。詳細につい

ては、 Goウェブサイトの「ダウンロードしてインストールする」を参照してください。などの別のベンダーのGoツールチェーンを使用している場合Red Hat、ディストリビューションとメタデータの可用性に潜在的な違いがあるため、評価が正確ではない可能性があります。

主な特徴

- バイナリから直接依存関係情報を抽出Goします
- バイナリ内に埋め込まれた依存関係を収集します
- バイナリのコンパイルに使用されるGoツールチェーンバージョンを検出して抽出します。

GOMODCACHE

Amazon Inspector SBOM Generator はGoモジュールキャッシュをスキャンして、インストールされた依存関係に関する情報を収集します。このキャッシュは、ダウンロードしたモジュールを保存して、異なるビルド間で同じバージョンが使用されていることを確認します。

主な特徴

- `GOMODCACHE` ディレクトリをスキャンしてキャッシュされたモジュールを識別します
- モジュール名、バージョン、ソース URLs

構造の例

以下は、`GOMODCACHE` 構造の例です。

```
~/go/pkg/mod/
### github.com/gin-gonic/gin@v1.7.2
### golang.org/x/crypto@v0.0.0-20210616213533-5cf6c0f8e123
```

> **Note**
>
> この構造は、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含め

ることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Java 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Java	Maven	コンパイルされたJavaアプリケーション (.jar/.war/.ear) pom.xml	該当なし 該当なし	該当なし 該当なし	はい はい	該当なし 該当なし	はい はい

Amazon Inspector SBOM Generator は、コンパイルされたJavaアプリケーションとpom.xmlファイルを分析してJava依存関係スキャンを実行します。コンパイルされたアプリケーションをスキャンすると、スキャナーは整合性検証用の SHA-1 ハッシュを生成し、埋め込みpom.propertiesファイルを抽出して、ネストされたpom.xmlファイルを解析します。

SHA-1 ハッシュコレクション (コンパイルされた .jar、.war、.ear ファイル用)

Amazon Inspector SBOM Generator は、コンパイルされたJavaアーティファクトの整合性とトレーサビリティを保証するために、プロジェクト内のすべての .ear、.jar、および.warファイルの SHA-1 ハッシュを収集しようとします。

主な特徴

- すべてのコンパイル済みJavaアーティファクトの SHA-1 ハッシュを生成します

アーティファクトの例

SHA-1 アーティファクトの例を次に示します。

```
{
  "bom-ref": "comp-52",
  "type": "library",
  "name": "jul-to-slf4j",
  "version": "2.0.6",
  "hashes": [
    {
      "alg": "SHA-1",
      "content": ""
    }
  ],
  "purl": "pkg:maven/jul-to-slf4j@2.0.6",
  "properties": [
    {
      "name": "amazon:inspector:sbom_generator:source_path",
      "value": "test-0.0.1-SNAPSHOT.jar/BOOT-INF/lib/jul-to-slf4j-2.0.6.jar"
    }
  ]
}
```

> **Note**
>
> このアーティファクトは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

pom.properties

`pom.properties` ファイルは、パッケージ名やパッケージバージョンなどのプロジェクトメタデータを保存Mavenするためにプロジェクトで使用されます。Amazon Inspector SBOM Generator はこのファイルを解析してプロジェクト情報を収集します。

主な特徴

- パッケージアーティファクト、パッケージグループ、パッケージバージョンを解析して抽出します

pom.properties ファイルの例

次は、pom.properties ファイルの例です。

```
#Generated by Maven
#Tue Mar 16 15:44:02 UTC 2021

version=1.6.0
groupId=net.datafaker
artifactId=datafaker
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

ネストされた**pom.xml**解析の除外

コンパイルされたJavaアプリケーションのスキャン時にpom.xml解析を除外する場合は、 --skip-nested-pomxml引数を使用します。

pom.xml

pom.xml ファイルは、Mavenプロジェクトのコア設定ファイルです。これには、プロジェクトとプロジェクトの依存関係に関する情報が含まれています。Amazon Inspector SBOM Generator は pom.xmlファイルを解析して依存関係を収集し、リポジトリ内のスタンドアロンファイルとコンパイル済みファイル内の.jarファイルをスキャンします。

主な特徴

- pom.xml ファイルからパッケージアーティファクト、パッケージグループ、パッケージバージョンを解析して抽出します。

サポートされているMavenスコープとタグ

依存関係は、次のMavenスコープで収集されます。

- compile
- provided
- ランタイム
- test
- システム
- インポート

依存関係は、 Maven タグ で収集されます<optional>true</optional>。

スコープを持つ**pom.xml**サンプルファイル

以下は、スコープを持つpom.xmlファイルの例です。

```
<dependency>
<groupId>jakarta.servlet</groupId>
<artifactId>jakarta.servlet-api</artifactId>
</version>6.0.0</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.28</version>
<scope>runtime</scope>
</dependency>
```

スコープのない**pom.xml**ファイルの例

以下は、スコープのないpom.xmlファイルの例です。

```
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.17.1</version>
</dependency>

<dependency>
<groupId>org.jenkins-ci.plugins</groupId>
<artifactId>plain-credentials</artifactId>
<version>183.va_de8f1dd5a_2b_</version>
</dependency>

<dependency>
<groupId>org.jenkins-ci.plugins</groupId>
<artifactId>jackson2-api</artifactId>
<version>2.15.2-350.v0c2f3f8fc595</version>
</dependency>
```

> **Note**
>
> これらのファイルはそれぞれ、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

JavaScript 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Javascript	Node Modules	node_modules/	該当なし	該当なし	はい	はい	はい

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
	NPM	`*/package.json`	該当なし	はい	該当なし	該当なし	いいえ
	PNPM		該当なし	あり	該当なし	該当なし	いいえ

	YARN	`package-lock.json` (v1, v2, and v3) / `npm-shrinkwrap.json`
		`pnpm-lock.yaml`
		`yarn.lock`
			該当なし	あり	該当なし	該当なし	いいえ

package.json

`package.json` ファイルはプロジェクトのコアコンポーネントですNode.js。インストールされたパッケージに関するメタデータが含まれています。Amazon Inspector SBOM Generator はこのファイルをスキャンして、パッケージ名とパッケージバージョンを識別します。

主な特徴

- JSON ファイル構造を解析してパッケージ名とバージョンを抽出します
- プライベート値を持つプライベートパッケージを識別します

package.json ファイルの例

次は、package.json ファイルの例です。

```
{
"name": "arrify",
"private": true,
"version": "2.0.1",
"description": "Convert a value to an array",
"license": "MIT",
"repository": "sindresorhus/arrify"
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

package-lock.json

package-lock.json ファイルは npm によって自動的に生成され、プロジェクトにインストールされた依存関係の正確なバージョンをロックします。すべての依存関係とそのサブ依存関係の正確なバージョンを保存することで、環境の一貫性を確保します。このファイルは、通常の依存関係と開発依存関係を区別できます。

主な特徴

- JSON ファイル構造を解析してパッケージ名とパッケージバージョンを抽出します
- 開発依存関係の検出をサポート

package-lock.json ファイルの例

次は、package-lock.json ファイルの例です。

```
"verror": {
"version": "1.10.0",
"resolved": "https://registry.npmjs.org/verror/-/verror-1.10.0.tgz",
"integrity": "sha1-OhBcoXBTr1XW4nDB+CiGguGNpAA=",
"requires": {
    "assert-plus": "^1.0.0",
    "core-util-is": "1.0.2",
    "extsprintf": "^1.2.0"
}
},
"wrappy": {
"version": "1.0.2",
"resolved": "https://registry.npmjs.org/wrappy/-/wrappy-1.0.2.tgz",
"integrity": "sha1-tSQ9jz7BqjXxNkYFvA0QNuMKtp8=",
"dev": true
},
"yallist": {
"version": "3.0.2",
"resolved": "https://registry.npmjs.org/yallist/-/yallist-3.0.2.tgz",
"integrity": "sha1-hFK0u36Dx8GI2AQcGoN8dz1ti7k="
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

npm-shrinkwrap.json

npm は、 および `npm-shrinkwrap.json` ファイルを自動的に生成`package-lock.json`して、プロジェクトにインストールされた依存関係の正確なバージョンをロックします。これにより、すべての依存関係とサブ依存関係の正確なバージョンを保存することで、環境の一貫性が保証されます。ファイルは、通常の依存関係と開発依存関係を区別します。

主な特徴

- JSON ファイル構造のpackage-lockバージョン 1、2、および 3 を解析してパッケージ名とバージョンを抽出する
- 開発者の依存関係の検出がサポートされています (package-lock.json は本番環境と開発の依存関係をキャプチャし、ツールが開発環境で使用されているパッケージを特定できるようにします)
- npm-shrinkwrap.json ファイルは package-lock.json ファイルよりも優先されます

例

次は、package-lock.json ファイルの例です。

```
"verror": {
            "version": "1.10.0",
            "resolved": "https://registry.npmjs.org/verror/-/verror-1.10.0.tgz",
            "integrity": "sha1-OhBcoXBTr1XW4nDB+CiGguGNpAA=",
            "requires": {
                "assert-plus": "^1.0.0",
                "core-util-is": "1.0.2",
                "extsprintf": "^1.2.0"
            }
        },
        "wrappy": {
            "version": "1.0.2",
            "resolved": "https://registry.npmjs.org/wrappy/-/wrappy-1.0.2.tgz",
            "integrity": "sha1-tSQ9jz7BqjXxNkYFvA0QNuMKtp8=",
            "dev": true
        },
        "yallist": {
            "version": "3.0.2",
            "resolved": "https://registry.npmjs.org/yallist/-/yallist-3.0.2.tgz",
            "integrity": "sha1-hFK0u36Dx8GI2AQcGoN8dz1ti7k="
}
```

pnpm-yaml.lock

pnpm-lock.yaml ファイルは、インストールされた依存関係バージョンのレコードを維持するために pnpm によって生成されます。また、開発の依存関係を個別に追跡します。

主な特徴

- YAML ファイル構造を解析してパッケージ名とバージョンを抽出します
- 開発依存関係の検出をサポート

例

次は、`pnpm-lock.yaml` ファイルの例です。

```
lockfileVersion: 5.3
importers:
my-project:
dependencies:
  lodash: 4.17.21
devDependencies:
  jest: 26.6.3
specifiers:
  lodash: ^4.17.21
  jest: ^26.6.3
packages:
/lodash/4.17.21:
resolution:
  integrity: sha512-xyz
engines:
  node: '>=6'
dev: false
/jest/26.6.3:
resolution:
  integrity: sha512-xyz
dev: true
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

yarn.lock

Amazon Inspector SBOM Generator は、コンパイルされたJavaアーティファクトの整合性とトレーサビリティを保証するために、プロジェクト内の `.ear`、`.jar`、および `.war`ファイルの SHA-1 ハッシュを収集しようとします。

主な特徴

- すべてのコンパイル済みJavaアーティファクトの SHA-1 ハッシュを生成します

SHA-1 アーティファクトの例

SHA-1 アーティファクトの例を次に示します。

```
"@ampproject/remapping@npm:^2.2.0":
version: 2.2.0
resolution: "@ampproject/remapping@npm:2.2.0"
dependencies:
"@jridgewell/gen-mapping": ^0.1.0
"@jridgewell/trace-mapping": ^0.3.9
checksum:
 d74d170d06468913921d72430259424b7e4c826b5a7d39ff839a29d547efb97dc577caa8ba3fb5cf023624e9af9d09
languageName: node
linkType: hard

"@babel/code-frame@npm:^7.0.0, @babel/code-frame@npm:^7.12.13, @babel/code-
frame@npm:^7.18.6, @babel/code-frame@npm:^7.21.4":
version: 7.21.4
resolution: "@babel/code-frame@npm:7.21.4"
dependencies:
"@babel/highlight": ^7.18.6
checksum:
 e5390e6ec1ac58dcef01d4f18eaf1fd2f1325528661ff6d4a5de8979588b9f5a8e852a54a91b923846f7a5c681b217
languageName: node
linkType: hard
```

> **Note**
>
> このアーティファクトは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定

し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

.NET 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
.NET	.NET Core	*.deps.json	該当なし	該当なし	該当なし	該当なし	はい
	Nuget	Packages.config	該当なし	該当なし	該当なし	該当なし	あり
	Nuget	packages.lock.json	該当なし	該当なし	はい	該当なし	あり
	.NET	.csproj	該当なし	該当なし	該当なし	該当なし	はい

Packages.config

`Packages.config` ファイルは、プロジェクトの依存関係を管理するNugetために の古いバージョンで使用される XML ファイルです。特定のバージョンを含む、プロジェクトによって参照されるすべてのパッケージが一覧表示されます。

主な特徴

- XML 構造を解析してパッケージ IDsとバージョンを抽出する

例

次は、`Packages.config` ファイルの例です。

```
<?xml version="1.0" encoding="utf-8"? >
<packages>
<package id="FluentAssertions" version="5.4.1" targetFramework="net461" />
<package id="Newtonsoft.Json" version="11.0.2" targetFramework="net461" />
<package id="SpecFlow" version="2.4.0" targetFramework="net461" />
<package id="SpecRun.Runner" version="1.8.0" targetFramework="net461" />
<package id="SpecRun.SpecFlow" version="1.8.0" targetFramework="net461" />
<package id="SpecRun.SpecFlow.2-4-0" version="1.8.0" targetFramework="net461" />
<package id="System.ValueTuple" version="4.5.0" targetFramework="net461" />
</packages>
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

*.deps.json

`*.deps.json` ファイルは.NET Coreプロジェクトによって生成され、パス、バージョン、ランタイム依存関係など、すべての依存関係に関する詳細情報が含まれています。このファイルは、ランタイムに正しいバージョンの依存関係をロードするために必要な情報があることを確認します。

主な特徴

- JSON 構造を解析して、包括的な依存関係の詳細を取得します
- `libraries` リスト内のパッケージ名とバージョンを抽出します。

`.deps.json` ファイルの例

次は、`.deps.json` ファイルの例です。

```
{
"runtimeTarget": {
```

```
    "name": ".NETCoreApp,Version=v7.0",
    "signature": ""
},
"libraries": {
    "sample-Nuget/1.0.0": {
        "type": "project",
        "serviceable": false,
        "sha512": ""
    },
    "Microsoft.EntityFrameworkCore/7.0.5": {
        "type": "package",
        "serviceable": true,
        "sha512": "sha512-
RXbRLHHWP2Z3pq8qcL5nQ6LPeoOyp8hasM5bd0Te8PiQi3RjWQR4tcbdY5XMqQ+oTO9wA8/RLhZRn/
hnxlTDnQ==",
        "path": "microsoft.entityframeworkcore/7.0.5",
        "hashPath": "microsoft.entityframeworkcore.7.0.5.nupkg.sha512"
    },
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

packages.lock.json

`packages.lock.json` このファイルは、新しいバージョンの によって使用されNuget、.NETプロジェクトの依存関係の正確なバージョンをロックして、異なる環境で同じバージョンが一貫して使用されるようにします。

主な特徴

- JSON 構造を解析して、ロックされた依存関係を一覧表示します
- 直接依存関係と推移依存関係の両方をサポート
- パッケージ名と解決済みバージョンを抽出します

packages.lock.json ファイルの例

次は、packages.lock.json ファイルの例です。


```
{
"version": 1,
"dependencies": {
"net7.0": {
  "Microsoft.EntityFrameworkCore": {
    "type": "Direct",
    "requested": "[7.0.5, )",
    "resolved": "7.0.5",
    "contentHash": "RXbRLHHWP2Z3pq8qcL5nQ6LPeoOyp8hasM5bd0Te8PiQi3RjWQR4tcbdY5XMqQ
+oTO9wA8/RLhZRn/hnxlTDnQ==",
    "dependencies": {
      "Microsoft.EntityFrameworkCore.Abstractions": "7.0.5",
      "Microsoft.EntityFrameworkCore.Analyzers": "7.0.5",
      "Microsoft.Extensions.Caching.Memory": "7.0.0",
      "Microsoft.Extensions.DependencyInjection": "7.0.0",
      "Microsoft.Extensions.Logging": "7.0.0"
    }
  },
  "Newtonsoft.Json": {
    "type": "Direct",
    "requested": "[13.0.3, )",
    "resolved": "13.0.3",
    "contentHash": "HrC5BXdl00IP9zeV+0Z848QWPAoCr9P3bDEZguI+gkLcBKAOxix/tLEAAHC
+UvDNPv4a2d18lOReHMOagPa+zQ=="
  },
  "Microsoft.Extensions.Primitives": {
    "type": "Transitive",
    "resolved": "7.0.0",
    "contentHash": "um1KU5kxcRp3CNuI8o/GrZtD4AIOXDk
+RLsytjZ9QPok3ttLUelLKpilVPuaFT3TFjOhSibUAso0odbOaCDj3Q=="
  }
}
}
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

.csproj

`.csproj` ファイルは XML で記述され、プロジェクトの.NETプロジェクトファイルとして記述されます。これには、Nugetパッケージ、プロジェクトプロパティ、ビルド設定への参照が含まれます。

主な特徴

- XML を解析してパッケージ参照を抽出する構造

`.csproj` ファイルの例

次は、`.csproj` ファイルの例です。

```
<Project Sdk="Microsoft.NET.Sdk">
<PropertyGroup>
<TargetFramework>net7.0</TargetFramework>
<RootNamespace>sample_Nuget</RootNamespace>
<ImplicitUsings>enable</ImplicitUsings>
<Nullable>enable</Nullable>
<RestorePackagesWithLockFile>true</RestorePackagesWithLockFile>
</PropertyGroup>
<ItemGroup>
</ItemGroup>
<ItemGroup>
<PackageReference Include="Newtonsoft.Json" Version="13.0.3" />
<PackageReference Include="Microsoft.EntityFrameworkCore" Version="7.0.5" />
</ItemGroup>
</Project>
```

`.csproj` ファイルの例

次は、.csproj ファイルの例です。

```
<PackageReference Include="ExamplePackage" Version="6.*" />
<PackageReferencePackageReference Include="ExamplePackage" Version="(4.1.3,)" />
<PackageReference Include="ExamplePackage" Version="(,5.0)" />
<PackageReference Include="ExamplePackage" Version="[1,3)" />
<PackageReference Include="ExamplePackage" Version="[1.3.2,1.5)" />
```

> **Note**
>
> これらのファイルはそれぞれ、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

PHP 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
PHP	Composer	composer.lock /vendor/composer/installed.json	該当なし 該当なし	該当なし 該当なし	はい はい	該当なし 該当なし	はい はい

composer.lock

`composer.lock` ファイルは、コンポーザーのインストールコマンドまたはコンポーザー更新コマンドを実行すると自動的に生成されます。このファイルは、すべての環境に同じバージョンの依存関係がインストールされていることを保証します。これにより、一貫性のある信頼性の高いビルドプロセスが提供されます。

主な特徴

- 構造化データの JSON 形式を解析します
- 依存関係の名前とバージョンを抽出します

`composer.lock` ファイルの例

次は、`composer.lock` ファイルの例です。

```
{
"packages": [
    {
        "name": "nesbot/carbon",
        "version": "2.53.1",
        // TRUNCATED
    },
    {
        "name": "symfony/deprecation-contracts",
        "version": "v3.2.1",
        // TRUNCATED
    },
    {
        "name": "symfony/polyfill-mbstring",
        "version": "v1.27.0",
        // TRUNCATED
    }
]
// TRUNCATED
}
```

> **Note**
>
> これにより、パッケージ URL を含む出力が生成されます。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

/vendor/composer/installed.json

`/vendor/composer/installed.json` ファイルは `vendor/composer` ディレクトリにあり、インストールされているすべてのパッケージとパッケージバージョンの包括的なリストを提供します。

主な特徴

- 構造化データの JSON 形式を解析します
- 依存関係名とバージョンを抽出します

`/vendor/composer/installed.json` ファイルの例

次は、`/vendor/composer/installed.json` ファイルの例です。

```
{
"packages": [
    {
        "name": "nesbot/carbon",
        "version": "2.53.1",
        // TRUNCATED
    },
    {
        "name": "symfony/deprecation-contracts",
        "version": "v3.2.1",
        // TRUNCATED
    },
    {
        "name": "symfony/polyfill-mbstring",
        "version": "v1.27.0",
        // TRUNCATED
    }
]
```

```
// TRUNCATED
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Python 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Python	pip	requirements.txt	該当なし	該当なし	該当なし	該当なし	はい
	Poetry	Poetry.lock	該当なし	該当なし	該当なし	該当なし	あり
	Pipenv	Pipfile.lock	該当なし	該当なし	該当なし	該当なし	あり
	Egg/Wheel	.egg-info/PKG-INFO	該当なし	該当なし	該当なし	該当なし	あり
		.dist-info/METADATA	該当なし	該当なし	該当なし	該当なし	はい

requirements.txt

`requirements.txt` ファイルは、Pythonプロジェクトの依存関係を指定するためにプロジェクトで広く使用されている形式です。このファイルの各行には、バージョン制約を含むパッケージが含まれています。Amazon Inspector SBOM Generator はこのファイルを解析して、依存関係を正確に識別してカタログ化します。

主な特徴

- バージョン指定子をサポート (== および ~=)
- コメントと複雑な依存関係の行をサポート

> **Note**
> バージョン指定子 <= および => はサポートされていません。

`requirements.txt` ファイルの例

次は、`requirements.txt` ファイルの例です。

```
flask==1.1.2
requests==2.24.0
numpy==1.18.5
foo~=1.2.0
# Comment about a dependency
scipy. # invalid
```

> **Note**
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Pipfile.lock

Pipenv は、すべてのパッケージングワールド (バンドル、ピン留め、ピン留めなし) を最大限に活用するツールです。は、決定論的なビルドを容易にするために、依存関係の正確なバージョンを`Pipfile.lock`ロックします。Amazon Inspector SBOM Generator はこのファイルを読み取り、依存関係とその解決済みバージョンを一覧表示します。

主な特徴

- 依存関係解決のための JSON 形式を解析します
- デフォルトと開発の依存関係をサポート

`Pipfile.lock` ファイルの例

次は、`Pipfile.lock` ファイルの例です。

```
{
"default": {
    "requests": {
        "version": "==2.24.0",
        "hashes": [
            "sha256:cc718bb187e53b8d"
        ]
    }
},
"develop": {
    "blinker": {
        "hashes": [
            "sha256:1779309f71bf239144b9399d06ae925637cf6634cf6bd131104184531bf67c01",
            "sha256:8f77b09d3bf7c795e969e9486f39c2c5e9c39d4ee07424be2bc594ece9642d83"
        ],
        "markers": "python_version >= '3.8'",
        "version": "==1.8.2"
    }
}
}
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表を生成するときにソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Poetry.lock

Poetry は、Python の依存関係管理およびパッケージングツールです。このPoetry.lockファイルは、一貫した環境を容易にするために、依存関係の正確なバージョンをロックします。Amazon Inspector SBOM Generator は、このファイルから詳細な依存関係情報を抽出します。

主な特徴

- 構造化データの TOML 形式を解析します
- 依存関係名とバージョンを抽出します

Poetry.lock ファイルの例

次は、Poetry.lock ファイルの例です。

```
[[package]]
name = "flask"
version = "1.1.2"
description = "A simple framework for building complex web applications."
category = "main"
optional = false
python-versions = ">=3.5"
[[package]]
name = "requests"
version = "2.24.0"
description = "Python HTTP for Humans."
category = "main"
optional = false
python-versions = ">=3.5"
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

エッグ/ホイール

グローバルにインストールされた Python パッケージの場合、Amazon Inspector SBOM Generator は `.egg-info/PKG-INFO`および `.dist-info/METADATA` ディレクトリにあるメタデータファイルの解析をサポートしています。これらのファイルは、インストールされたパッケージに関する詳細なメタデータを提供します。

主な特徴

- パッケージ名とバージョンを抽出します
- エッグ形式とホイール形式の両方をサポート

`PKG-INFO/METADATA` ファイルの例

次は、`PKG-INFO/METADATA` ファイルの例です。

```
Metadata-Version: 1.2
Name: Flask
Version: 1.1.2
Summary: A simple framework for building complex web applications.
Home-page: https://palletsprojects.com/p/flask/
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Ruby 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	Toolchain g のサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Ruby	Bundler	Gemfile.lock .gemspec globall installed Gems	該当なし 該当なし 該当なし	該当なし 該当なし 該当なし	はい 該当なし 該当なし	該当なし 該当なし 該当なし	はい あり はい

Gemfile.lock

このGemfile.lockファイルは、すべての依存関係の正確なバージョンをロックして、すべての環境で同じバージョンが使用されていることを確認します。

主な特徴

- Gemfile.lock ファイルを ID 依存関係と依存関係バージョンに解析します
- 詳細なパッケージ名とパッケージバージョンを抽出します

Gemfile.lock ファイルの例

次は、Gemfile.lock ファイルの例です。

```
GEM
remote: https://rubygems.org/
specs:
ast (2.4.2)
awesome_print (1.9.2)
diff-lcs (1.5.0)
json (2.6.3)
```

```
parallel (1.22.1)
parser (3.2.2.0)
nokogiri (1.16.6-aarch64-linux)
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

.gemspec

`.gemspec` ファイルは、gem に関するメタデータを含むRubyGemファイルです。Amazon Inspector SBOM Generator はこのファイルを解析して、gem に関する詳細情報を収集します。

主な特徴

- Gem 名と Gem バージョンを解析して抽出します

> **Note**
>
> リファレンス仕様はサポートされていません。

`.gemspec` ファイルの例

次は、`.gemspec` ファイルの例です。

```
Gem::Specification.new do |s|
s.name        = "generategem"
s.version     = "2.0.0"
s.date        = "2020-06-12"
s.summary     = "generategem"
s.description = "A Gemspec Builder"
s.email       = "edersondeveloper@gmail.com"
s.files       = ["lib/generategem.rb"]
```

```
s.homepage    = "https://github.com/edersonferreira/generategem"
s.license     = "MIT"
s.executables = ["generategem"]
s.add_dependency('colorize', '~> 0.8.1')
end
```

```
# Not supported

Gem::Specification.new do |s|
s.name        = &class1
s.version     = &foo.bar.version
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表を生成するときにソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

グローバルにインストールされた Gem

Amazon Inspector SBOM Generator は、グローバルにインストールされた Gem のスキャンをサポートしています。Gem は、`/usr/local/lib/ruby/gems/<ruby_version>/gems/`Amazon EC2/Amazon ECR や `ruby/gems/<ruby_version>/gems/` Lambda などの標準ディレクトリにあります。これにより、グローバルにインストールされた依存関係がすべて識別され、カタログ化されます。

主な特徴

- 標準ディレクトリにグローバルにインストールされているすべての Gem を識別してスキャンします
- グローバルにインストールされた各 Gem のメタデータとバージョン情報を抽出します

ディレクトリ構造の例

ディレクトリ構造の例を次に示します。

```
.
### /usr/local/lib/ruby/3.5.0/gems/
### actrivesupport-6.1.4
### concurrent-ruby-1.1.9
### i18n-1.8.10
```

> **Note**
>
> この構造は、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Rust 依存関係スキャン

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
Rust	Cargo.toml	Cargo.toml Cargo.lock Rust binary (built with cargo-	該当なし 該当なし はい	該当なし 該当なし 該当なし	該当なし はい 該当なし	該当なし 該当なし 該当なし	はい あり はい

プログラム言語	パッケージマネージャー	サポートされているアーティファクト	ツールチェーンのサポート	開発の依存関係	推移的な依存関係	プライベートフラグ	再帰的
		aud itable)					

Cargo.toml

`Cargo.toml` ファイルはプロジェクトのマニフェストファイルですRust。

主な特徴

- `Cargo.toml` ファイルを解析して抽出し、プロジェクトパッケージ名とバージョンを識別します。

`Cargo.toml` ファイルの例

次は、`Cargo.toml` ファイルの例です。

```
[package]
name = "wait-timeout"
version = "0.2.0"
description = "A crate to wait on a child process with a timeout specified across Unix
 and\nWindows platforms.\n"
homepage = "https://github.com/alexcrichton/wait-timeout"
documentation = "https://docs.rs/wait-timeout"
readme = "README.md"
categories = ["os"]
license = "MIT/Apache-2.0"
repository = "https://github.com/alexcrichton/wait-timeout"
[target."cfg(unix)".dependencies.libc]
version = "0.2"
[badges.appveyor]
repository = "alexcrichton/wait-timeout"
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

Cargo.lock

`Cargo.lock` ファイルは依存関係のバージョンをロックして、プロジェクトが構築されるたびに同じバージョンが使用されていることを確認します。

主な特徴

- `Cargo.lock` ファイルを解析して、すべての依存関係と依存関係のバージョンを識別します。

`Cargo.lock` ファイルの例

次は、`Cargo.lock` ファイルの例です。

```
# This file is automatically @generated by Cargo.
# It is not intended for manual editing.
[[package]]
name = "adler32"
version = "1.0.3"
source = "registry+https://github.com/rust-lang/crates.io-index"

[[package]]
name = "aho-corasick"
version = "0.7.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
```

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に

含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

貨物監査可能な錆びたバイナリ

Amazon Inspector SBOM Generator は、`cargo-auditable`ライブラリで構築されたRustバイナリから依存関係を収集します。これにより、コンパイルされたバイナリから依存関係を抽出できるようにすることで、追加の依存関係情報が提供されます。

主な特徴

- `cargo-auditable` ライブラリで構築されたRustバイナリから依存関係情報を直接抽出します
- バイナリに含まれる依存関係のメタデータとバージョン情報を取得します。

> **Note**
>
> このファイルは、パッケージ URL を含む出力を生成します。この URL を使用して、ソフトウェア部品表の生成時にソフトウェアパッケージに関する情報を指定し、ScanSbom API に含めることができます。詳細については、GitHub ウェブサイトの package-url を参照してください。

サポートされていないアーティファクト

このセクションでは、サポートされていないアーティファクトについて説明します。

Java

Amazon Inspector SBOM Generator ジェネレーターは、メインストリームMavenリポジトリから取得された依存関係の脆弱性検出のみをサポートします。Red Hat Maven や などのプライベートMavenリポジトリやカスタムリポジトリJenkinsはサポートされていません。脆弱性を正確に検出するには、Java依存関係がメインストリームMavenリポジトリからプルされていることを確認します。他のリポジトリからの依存関係は、脆弱性スキャンではカバーされません。

JavaScript

esbuild バンドル

esbuild 最小化されたバンドルの場合、Amazon Inspector SBOM Generator は を使用するプロジェクトの依存関係スキャンをサポートしていませんesbuild。によって生成されたソースマップには、正確なSbomgen生成に必要な十分なメタデータ (依存関係の名前とバージョン) esbuildが含まれていません。信頼性の高い結果を得るには、バンドルプロセス`package-lock.json`の前に、`node_modules/directory`や などの元のプロジェクトファイルをスキャンします。

package.json

Amazon Inspector SBOM Generator は、ルートレベルの package.json ファイルの依存関係情報のスキャンをサポートしていません。このファイルはパッケージ名とバージョン範囲のみを指定しますが、完全に解決されたパッケージバージョンは含まれません。正確なスキャン結果を得るには、`package.json` または解決済みバージョンを含む `pnpm.lock``yarn.lock`や などの他のロックファイルを使用します。

ドットネット

でフローティングバージョンまたはバージョン範囲を使用する場合`PackageReference`、パッケージ解決を実行せずにプロジェクトで使用される正確なパッケージバージョンを特定することが難しくなります。フローティングバージョンとバージョン範囲により、開発者は固定バージョンではなく許容可能なパッケージバージョンの範囲を指定できます。

Go バイナリ

Amazon Inspector SBOM Generator は、ビルド ID Go を除外するように設定されたビルドフラグを使用して構築されたバイナリをスキャンしません。これらのビルドフラグにより、 Bomerman はバイナリを元のソースに正確にマッピングできなくなります。パッケージ情報を抽出できないため、不明瞭なGoバイナリはサポートされていません。正確な依存関係スキャンを行うには、バイナリがビルド ID Go などのデフォルト設定で構築されていることを確認してください。

Rust バイナリ

Amazon Inspector SBOM Generator はRust、バイナリが貨物監査可能なライブラリを使用して構築されている場合にのみバイナリをスキャンします。このライブラリを利用していないRustバイナリには、正確な依存関係抽出に必要なメタデータがありません。Amazon Inspector SBOM Generator は、1.7.3 Rust からコンパイルされたRustツールチェーンバージョンを抽出しますが、Linux環境内のバイナリに対してのみ抽出します。包括的なスキャンを行うには、 で、貨物監査可能な Rust Linuxを使用してバイナリを構築します。

> **Note**
>
> Rust ツールチェーンバージョンが抽出された場合でも、ツールチェーン自体の脆弱性検出はサポートされていません。

Amazon Inspector SBOM Generator の包括的なエコシステムコレクション

Amazon Inspector SBOM Generator は、ソフトウェア部品表 (SBOM) を作成し、オペレーティングシステムやプログラミング言語からサポートされているパッケージの脆弱性スキャンを実行するためのツールです。また、コアオペレーティングシステム以外のさまざまなエコシステムのスキャンをサポートし、インフラストラクチャコンポーネントの堅牢で詳細な分析を行います。SBOM を生成することで、ユーザーは最新のテクノロジースタックの構成を理解し、エコシステムコンポーネントの脆弱性を特定し、サードパーティーのソフトウェアを可視化できます。

サポートされているエコシステム

エコシステムコレクションは、OS パッケージマネージャーを通じてインストールされたパッケージを超えて SBOM 生成を拡張します。これは、手動インストールなどの代替方法でデプロイされたアプリケーションのコレクションを通じて行われます。Amazon Inspector SBOM Generator は、次のエコシステムのスキャンをサポートしています。

エコシステム	アプリケーション
Oracle Java	JDK JRE Amazon Corretto
Apache	httpd tomcat
WordPress	core プラグイン

エコシステム	アプリケーション
	テーマ
Google	Chrome
Node.JS	ノード

Apache エコシステムコレクション

Amazon Inspector SBOM Generator は、プラットフォーム間で共通のApacheインストールパスにあるインストールをスキャンします。

- macOS: `/Library/`
- Linux: `/etc/`, `/usr/share`, `/usr/lib`, `/usr/local`, `/var`, `/opt`

サポートされているアプリケーション

- httpd
- tomcat

主な特徴

- Apache httpd – `/include/ap_release.h`ファイルを解析して、メジャー識別子文字列、マイナー識別子文字列、パッチ識別子文字列を含むインストールマクロを抽出します。
- Apache tomcat – `catalina.jar`ファイルを解凍して、バージョン文字列を含む (`META-INF/MANIFEST.MF`) ファイル内のインストールマクロを抽出します。

`ap_release.h` ファイルの例

以下は、 `ap_release.h` ファイル内のコンテンツの例です。

```
//truncated

#define AP_SERVER_BASEVENDOR "Apache Software Foundation"
#define AP_SERVER_BASEPROJECT "Apache HTTP Server"
```

```
#define AP_SERVER_BASEPRODUCT "Apache"

#define AP_SERVER_MAJORVERSION_NUMBER 2
#define AP_SERVER_MINORVERSION_NUMBER 4
#define AP_SERVER_PATCHLEVEL_NUMBER   1
#define AP_SERVER_DEVBUILD_BOOLEAN    0

//truncated
```

PURL の例

Apache httpd アプリケーションのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/apache/httpd@2.4.1
```

catalina.jar/META-INF/MANIFEST.MF ファイルの例

以下は、 catalina.jar/META-INF/MANIFEST.MF ファイル内のコンテンツの例です。

```
//truncated

Implementation-Title: Apache Tomcat
Implementation-Vendor: Apache Software Foundation
Implementation-Version: 10.1.31

//truncated
```

PURL の例

Apache Tomcat アプリケーションのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/apache/tomcat@10.1.31
```

Java エコシステムコレクション

サポートされているアプリケーション

- Oracle JDK
- Oracle JRE
- Amazon Corretto

主な特徴

- Java インストールの文字列を抽出します。
- Java ランタイムを含むディレクトリパスを識別します。
- ベンダーを Oracle JDK、Oracle JRE、および として識別しますAmazon Corretto。

Amazon Inspector SBOM Generator は、次のJavaインストールパスとプラットフォーム全体のインストールをスキャンします。

- macOS: `/Library/Java/JavaVirtualMachines`
- Linux 32-bit: `/usr/lib/jvm`
- Linux 64-bit: `/usr/lib64/jvm`
- Linux (generic): `/usr/java and /opt/java`

Java バージョン情報の例

以下は、 Oracle Javaリリースの例です。

```
// Amazon Corretto
IMPLEMENTOR="Amazon.com Inc."
IMPLEMENTOR_VERSION="Corretto-17.0.11.9.1"
JAVA_RUNTIME_VERSION="17.0.11+9-LTS"
JAVA_VERSION="17.0.11"
JAVA_VERSION_DATE="2024-04-16"
LIBC="default"
MODULES="java.base java.compiler java.datatransfer java.xml java.prefs java.desktop
 java.instrument java.logging java.management java.security.sasl java.naming
 java.rmi java.management.rmi java.net.http java.scripting java.security.jgss
 java.transaction.xa java.sql java.sql.rowset java.xml.crypto java.se java.smartcardio
```

```
 jdk.accessibility jdk.internal.jvmstat jdk.attach jdk.charsets jdk.compiler
 jdk.crypto.ec jdk.crypto.cryptoki jdk.dynalink jdk.internal.ed jdk.editpad
 jdk.hotspot.agent jdk.httpserver jdk.incubator.foreign jdk.incubator.vector
 jdk.internal.le jdk.internal.opt jdk.internal.vm.ci jdk.internal.vm.compiler
 jdk.internal.vm.compiler.management jdk.jartool jdk.javadoc jdk.jcmd jdk.management
 jdk.management.agent jdk.jconsole jdk.jdeps jdk.jdwp.agent jdk.jdi jdk.jfr jdk.jlink
 jdk.jpackage jdk.jshell jdk.jsobject jdk.jstatd jdk.localedata jdk.management.jfr
 jdk.naming.dns jdk.naming.rmi jdk.net jdk.nio.mapmode jdk.random jdk.sctp
 jdk.security.auth jdk.security.jgss jdk.unsupported jdk.unsupported.desktop
 jdk.xml.dom jdk.zipfs"
OS_ARCH="x86_64"
OS_NAME="Darwin"
SOURCE=".:git:7917f11551e8+"

// JDK
IMPLEMENTOR="Oracle Corporation"
JAVA_VERSION="19"
JAVA_VERSION_DATE="2022-09-20"
LIBC="default"
MODULES="java.base java.compiler java.datatransfer java.xml java.prefs java.desktop
 java.instrument java.logging java.management java.security.sasl java.naming
 java.rmi java.management.rmi java.net.http java.scripting java.security.jgss
 java.transaction.xa java.sql java.sql.rowset java.xml.crypto java.se java.smartcardio
 jdk.accessibility jdk.internal.jvmstat jdk.attach jdk.charsets jdk.zipfs jdk.compiler
 jdk.crypto.ec jdk.crypto.cryptoki jdk.dynalink jdk.internal.ed jdk.editpad
 jdk.hotspot.agent jdk.httpserver jdk.incubator.concurrent jdk.incubator.vector
 jdk.internal.le jdk.internal.opt jdk.internal.vm.ci jdk.internal.vm.compiler
 jdk.internal.vm.compiler.management jdk.jartool jdk.javadoc jdk.jcmd jdk.management
 jdk.management.agent jdk.jconsole jdk.jdeps jdk.jdwp.agent jdk.jdi jdk.jfr jdk.jlink
 jdk.jpackage jdk.jshell jdk.jsobject jdk.jstatd jdk.localedata jdk.management.jfr
 jdk.naming.dns jdk.naming.rmi jdk.net jdk.nio.mapmode jdk.random jdk.sctp
 jdk.security.auth jdk.security.jgss jdk.unsupported jdk.unsupported.desktop
 jdk.xml.dom"
OS_ARCH="x86_64"
OS_NAME="Darwin"
SOURCE=".:git:53b4a11304b0 open:git:967a28c3d85f"
```

PURL の例

以下は、 Oracle Javaリリースのパッケージ URL の例です。

```
Sample PURL:
```

```
# Amazon Corretto
pkg:generic/amazon/amazon-corretto@21.0.3
# Oracle JDK
pkg:generic/oracle/jdk@11.0.16
# Oracle JRE
pkg:generic/oracle/jre@20
```

Google エコシステムコレクション

サポートされているアプリケーション

- Google Chrome

サポートされているアーティファクト

Amazon Inspector は、以下からGoogle Chrome情報を収集します。

- `chrome/VERSION` ファイル (ビルドソース)
- `puppeteer` ファイル (インストール)

Amazon Inspector SBOM Generator は、サポートされている各アーティファクトの対応するバージョンを解析して収集します。

`chrome/VERSION` バージョンファイルの例

`chrome/VERSION` バージョンファイルの例を次に示します。

```
MAJOR=130
MINOR=0
BUILD=6723
PATCH=58
```

PURL の例

`chrome/VERSION` バージョンファイルのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/google/chrome@131.0.6778.87
```

puppeteer バージョンファイルの例

puppeteer バージョンファイルの例を次に示します。

```
{
"name": "puppeteer",
"version": "23.9.0",
"description": "A high-level API to control headless Chrome over the DevTools
 Protocol",
"keywords": [
  "puppeteer",
  "chrome",
  "headless",
  "automation"
]
}
```

PURL の例

puppeteer バージョンファイルのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/google/puppeteer@23.9.0
```

WordPress エコシステムコレクション

サポートされているコンポーネント

- WordPress コア
- WordPress プラグイン
- WordPress テーマ

主な特徴

- WordPress core – `/wp-includes/version.php` ファイルを解析して $wp_version 変数からバージョン値を抽出します。

- WordPress plugins – /wp-content/plugins/<WordPress Plugin>/readme.txt ファイルまたは /wp-content/plugins/<WordPress Plugin>/readme.md ファイルを解析して、バージョン文字列としてStableタグを抽出します。
- WordPress テーマ – /wp-content/themes/<WordPress Theme>/style.cssファイルを解析して、バージョンメタデータからバージョンを抽出します。

version.php ファイルの例

WordPress コアversion.phpファイルの例を次に示します。

```
// truncated

/**
* The WordPress version string.
*
* Holds the current version number for WordPress core. Used to bust caches
* and to enable development mode for scripts when running from the /src directory.
*
* @global string $wp_version
*/
$wp_version = '6.5.5';

// truncated
```

PURL の例

WordPress コアのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/wordpress/core/wordpress@6.5.5
```

readme.txt ファイルの例

WordPress プラグインreadme.txtファイルの例を次に示します。

```
=== Plugin Name ===
```

```
Contributors: (this should be a list of wordpress.org userid's)
Donate link: https://example.com/
Tags: tag1, tag2
Requires at least: 4.7
Tested up to: 5.4
Stable tag: 4.3
Requires PHP: 7.0
License: GPLv2 or later
License URI: https://www.gnu.org/licenses/gpl-2.0.html

// truncated
```

PURL の例

WordPress プラグインのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/wordpress/plugin/exclusive-addons-for-elementor@1.0.0
```

style.css ファイルの例

WordPress テーマ`style.css`ファイルの例を次に示します。

```
/*
Author: the WordPress team
Author URI: https://wordpress.org
Description: Twenty Twenty-Four is designed to be flexible, versatile and applicable
 to any website. Its collection of templates and patterns tailor to different needs,
 such as presenting a business, blogging and writing or showcasing work. A multitude
 of possibilities open up with just a few adjustments to color and typography. Twenty
 Twenty-Four comes with style variations and full page designs to help speed up the
 site building process, is fully compatible with the site editor, and takes advantage
 of new design tools introduced in WordPress 6.4.
Requires at least: 6.4
Tested up to: 6.5
Requires PHP: 7.0
Version: 1.2
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Text Domain: twentytwentyfour
```

```
Tags: one-column, custom-colors, custom-menu, custom-logo, editor-style, featured-
images, full-site-editing, block-patterns, rtl-language-support, sticky-post,
 threaded-comments, translation-ready, wide-blocks, block-styles, style-variations,
 accessibility-ready, blog, portfolio, news
*/
```

PURL の例

WordPress テーマのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/wordpress/theme/avada@1.0.0
```

Node.JS ランタイムコレクション

サポートされているアプリケーション

- のノードランタイムバイナリ Node.JS

サポートされているアーティファクト

- MacOS および Linux – `node`、`asdf`、`fnm`、`nvm`、`volta`または公式Node.JSコンテナでインストールされたバイナリ詳細によるバイナリ検出。

例MacOSとLinuxパス

以下は、 MacOSおよび のパスの例ですLinux。

```
NVM:   ~/.nvm/, /usr/local/nvm
FNM:   ~/.local/share/fnm/
ASDF:  ~/.asdf/
MISE:  ~/.local/share/mise/
VOLTA: ~/.volta/
```

PURL の例

以下は、 のパッケージ URL の例ですNode.JS。

```
Sample PURL: pkg:generic/nodejs/node@20.18.0
```

OpenSSL エコシステムコレクション

サポートされているアプリケーション

OpenSSL ライブラリと開発パッケージのサポートは、公式の OpenSSL for 3.0.0 リリース以降で構築されたソフトウェアに限定されます。また、ソフトウェアはセマンティックバージョニングに従う必要があります。カスタムまたは分岐された OpenSSL バリアントと 3.0.0 より前のバージョンはサポートされていません。

Amazon Inspector SBOM Generator は、インストールされた各 OpenSSL インスタンスのキーパッケージ情報を抽出します。

主な特徴

- OpenSSL ヘッダーファイルから基本 SEMVER バージョン文字列を抽出します
- OpenSSL インストールを含むディレクトリパスを識別します

Amazon Inspector SBOM Generator は、プラットフォーム間で共通のインストールパスにある `openssl v.h` ファイルをスキャンして、OpenSSL のインストールを探します。

Linux/Unix のインストールパスの例

Linux/Unix のインストールパスの例を次に示します。

```
/usr/local/include/openssl/opensslv.h
/usr/local/ssl/include/openssl/opensslv.h
/usr/local/openssl/include/openssl/opensslv.h
/usr/local/opt/openssl/include/openssl/opensslv.h
/usr/include/openssl/opensslv.h
```

Amazon Inspector SBOM Generator は、`opensslv.h`ファイルを解析してバージョン定義を検索することで、バージョン情報を抽出します。

```
# define OPENSSL_VERSION_MAJOR  3
# define OPENSSL_VERSION_MINOR  4
# define OPENSSL_VERSION_PATCH  0
```

PURL の例

OpenSSL バージョンのパッケージ URL の例を次に示します。

```
Sample PURL: pkg:generic/openssl/openssl@3.4.0
```

Amazon Inspector SBOM Generator ライセンスコレクション

Amazon Inspector SBOM Generator は、ソフトウェア部品表 (SBOM) のライセンス情報を追跡するのに役立ちます。オペレーティングシステムとプログラミング言語全体でサポートされているパッケージからライセンス情報を収集します。生成された SBOM で標準化されたライセンス式を使用すると、ライセンス義務を理解できます。

ライセンス情報を収集する

コマンドの例

次の例は、 ディレクトリからライセンス情報を収集する方法を示しています。

```
./inspector-sbomgen directory --path /path/to/your/directory/ --collect-licenses
```

SBOM コンポーネントの例

次の例は、生成された SBOM のコンポーネントエントリを示しています。

```
"components": [
    {
      "bom-ref": "comp-2",
      "type": "application",
      "name": "sample-js-pkg",
      "version": "1.2.3",
      "licenses": [
        {
          "expression": "Apache-2.0 AND (MIT OR GPL-2.0-only)"
        }
      ],
      "purl": "pkg:npm/sample-js-pkg@1.2.3",
    }
  ]
```

サポートされているパッケージ

ライセンスコレクションでは、次のプログラミング言語とオペレーティングシステムパッケージがサポートされています。

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
Alma Linux	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
Amazon Linux	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
CentOS	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
Fedora	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
OpenSUSE	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
Oracle Linux	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
Photon OS	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
RHEL	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
Rocky Linux	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
SLES	RPM	• /usr/lib/sysimage/rpm/rpmdb.sqlite • /usr/lib/sysimage/rpm/Packages • /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	OS
Alpine Linux	APK	`/lib/apk/db/installed`	OS
Chainguard	APK	`/lib/apk/db/installed`	OS

ターゲット	パッケージマネージャー	ライセンス情報ソース	タイプ
Debian	DPKG	`/usr/share/doc/*/copyright`	OS
Ubuntu	DPKG	`/usr/share/doc/*/copyright`	OS
Node.js	Javascript	`node_modules/*/package.json`	プログラミング言語
PHP	Composer パッケージ	• `composer.lock` • `/vendor/composer/installed.json`	プログラミング言語
Go	Go	`LICENSE`	プログラミング言語
Python	Python/Egg/Wheel	• `.dist-info/METADATA` • `.egg-info` • `.egg-info/PKG-INFO`	プログラミング言語
Ruby	RubyGem	`*.gemspec`	プログラミング言語
Rust	crate	`Cargo.toml`	プログラム言語

ライセンス式の標準化

SPDX ライセンス表現形式は、オープンソースソフトウェアにあるライセンス条項を正確に表現します。Amazon Inspector SBOM Generator は、このセクションで説明するルールを使用して、すべてのライセンス情報を SPDX ライセンス式に標準化します。このルールは、ライセンス情報間の一貫性と互換性を提供します。

SPDX ショートフォーム識別子マッピング

すべてのライセンス名は SPDX ショートフォーム識別子にマッピングされます。例えば、`MIT License` は `MIT` に短縮されます。

複数のライセンスの組み合わせ

複数のライセンスを AND演算子と組み合わせることができます。コマンドをフォーマットする方法を示すコマンドの例を次に示します。

```
MIT AND Apache-2.0
```

カスタムライセンスプレフィックス

カスタムライセンスには`LicenseRef`、 などのプレフィックスが付きます`LicenseRef-CompanyPrivate`。

カスタム例外プレフィックス

カスタム例外には`AdditionRef-`、 などのプレフィックスが付きます`AdditionRef-CustomException`。

パッケージ URL とは

パッケージ URL または PURL は、さまざまなパッケージ管理システムのソフトウェアパッケージ、コンポーネント、ライブラリを識別するために使用される標準化された形式です。形式により、特に Software Bill of Materials (SBOMs。

PURL 構造

PURL 構造は URL に似ており、複数のコンポーネントで構成されています。

- `pkg` – リテラルプレフィックス
- `type`– パッケージタイプ
- `namespace` – グループ化
- `name` – パッケージ名
- `version` – パッケージバージョン
- `qualifiers` – 追加のキーと値のペア
- `subpath` – パッケージ内のファイルパス

PURL の例

PURL がどのように表示されるかの例を次に示します。

```
pkg:<type>/<namespace>/<name>@<version>?<qualifiers>#<subpath>
```

汎用 PURL

汎用 PURL は、、npm、 pypiなどの確立されたパッケージエコシステムに適合しないソフトウェアパッケージとコンポーネントを表すために使用されますmaven。ソフトウェアコンポーネントを識別し、特定のパッケージ管理システムと一致しない可能性のあるメタデータをキャプチャします。汎用 PURL は、コンパイルされたバイナリから Apacheや などのプラットフォームまで、さまざまなソフトウェアプロジェクトに役立ちますWordPress。これにより、コンパイルされたバイナリ、ウェブプラットフォーム、カスタムソフトウェアディストリビューションなど、幅広いユースケースに適用できます。

主なユースケース

- コンパイルされたバイナリをサポートし、 Goおよび に役立ちます。 Rust
- パッケージが従来のパッケージマネージャーに関連付けられていない可能性がある WordPressApacheや などのウェブプラットフォームをサポートします。
- 組織が内部で開発されたソフトウェアまたは正式なパッケージがないシステムを参照できるようにすることで、カスタムレガシーソフトウェアをサポートします。

形式例

一般的な PURL 形式の例を次に示します。

```
pkg:generic/<namespace>/<name>@<version>?<qualifiers>
```

汎用 PURL 形式のその他の例

一般的な PURL 形式の追加の例を次に示します。

コンパイルされたGoバイナリ

以下は、 で`inspector-sbomgen binary`コンパイルされた を表しますGo。

```
pkg:generic/inspector-sbomgen?go_toolchain=1.22.5
```

コンパイルされたRustバイナリ

以下は、 でコンパイルされたmyrustappバイナリを表しますRust。

```
pkg:generic/myrustapp?rust_toolchain=1.71.0
```

Apache プロジェクト

以下は、 Apache名前空間の http プロジェクトを指します。

```
pkg:generic/apache/httpd@1.0.0
```

WordPress ソフトウェア

以下は、コアWordPressソフトウェアを指します。

```
pkg:generic/wordpress/core/wordpress@6.0.0
```

WordPress テーマ

カスタムWordPressテーマを以下に示します。

```
pkg:generic/wordpress/theme/mytheme@1.0.0
```

WordPress プラグイン

カスタムWordPressプラグインを次に示します。

```
pkg:generic/wordpress/plugin/myplugin@1.0.0
```

Amazon Inspector SBOM Generator での未解決または非標準バージョン参照の処理

Amazon Inspector SBOM Generator は、ソースファイルから直接依存関係を識別することで、システム内でサポートされているアーティファクトを見つけて解析します。これはパッケージマネージャーではなく、バージョン範囲の解決、動的参照に基づくバージョンの推測、レジストリ検索の処理を行いません。依存関係は、プロジェクトソースアーティファクトで定義されている場合にのみ収集されます。多くの場合、、、 `package.json` `pom.xml`などのパッケージマニフェストの依存関係

はrequirements.txt、未解決のバージョンまたは範囲ベースのバージョンを使用して指定されます。このトピックでは、これらの依存関係がどのように表示されるかの例を示します。

レコメンデーション

Amazon Inspector SBOM Generator はソースアーティファクトから依存関係を抽出しますが、バージョン範囲や動的参照を解決または解釈しません。より正確な脆弱性スキャンと SBOMs を行うには、プロジェクトの依存関係で解決済みのセマンティックバージョン識別子を使用することをお勧めします。

Java

の場合Java、Mavenプロジェクトはバージョン範囲を使用してpom.xmlファイル内の依存関係を定義できます。

```
<dependency>
    <groupId>org.inspector</groupId>
    <artifactId>inspector-api</artifactId>
    <version>(,1.0]</version>
</dependency>
```

範囲は、1.0 までのバージョンが許容されることを指定します。ただし、バージョンが解決済みバージョンでない場合、Amazon Inspector SBOM Generator は特定のリリースにマッピングできないため、そのバージョンを収集しません。

JavaScript

の場合JavaScript、package.jsonファイルには次のようなバージョン範囲を含めることができます。

```
"dependencies": {
    "ky": "^1.2.0",
    "registry-auth-token": "^5.0.2",
    "registry-url": "^6.0.1",
    "semver": "^7.6.0"
}
```

^ 演算子は、指定されたバージョン以上の任意のバージョンが許容されることを指定します。ただし、指定されたバージョンが解決済みバージョンでない場合、Amazon Inspector SBOM Generator はそれを収集しません。収集しないと、脆弱性検出中に誤検出が発生する可能性があります。

Python

の場合Python、`requirements.txt`ファイルにはブール式を持つエントリを含めることができます。

```
requests>=1.0.0
```

>= 演算子は、 以上のバージョン`1.0.0`が許容されることを指定します。この特定の式は正確なバージョンを指定しないため、Amazon Inspector SBOM Generator は脆弱性分析のためにバージョンを確実に収集することはできません。

Amazon Inspector SBOM Generator は、ベータ、最新、スナップショットなどの非標準またはあいまいなバージョン識別子をサポートしていません。

```
pkg:maven/org.example.com/testmaven@1.0.2%20Beta-RC-1_Release
```

> **Note**
>
> などの非標準サフィックスの使用はBeta-RC-1_Release、標準のセマンティックバージョニングに準拠しておらず、Amazon Inspector 検出エンジン内の脆弱性について評価することはできません。

Amazon Inspector での CycloneDX 名前空間の使用

Amazon Inspector は、SBOM で使用できる CycloneDX 名前空間とプロパティ名を提供します。このセクションでは、CycloneDX SBOM のコンポーネントに追加できる可能性のあるすべてのカスタムキー/値プロパティについて説明します。詳細については、GitHub ウェブサイトの「CycloneDX プロパティ分類」を参照してください。

`amazon:inspector:sbom_scanner` 名前空間の分類

Amazon Inspector スキャン API は `amazon:inspector:sbom_scanner` 名前空間を使用し、以下のプロパティを持ちす。

プロパティ	説明
amazon:inspector:sbom_scanner:cisa_kev_date_added	脆弱性が CISA の「Known Exploited Vulnerabilities Catalog」に追加された時期を示します。
amazon:inspector:sbom_scanner:cisa_kev_date_due	CISA の「Known Exploited Vulnerabilities Catalog」に従って、脆弱性の修正期限がいつであるかを示します。
amazon:inspector:sbom_scanner:critical_vulnerabilities	SBOM で見つかった重大度が重大な脆弱性の総数。
amazon:inspector:sbom_scanner:exploit_available	特定の脆弱性に対して、エクスプロイトが利用可能かどうかを示します。
amazon:inspector:sbom_scanner:exploit_last_seen_in_public	特定の脆弱性に対するエクスプロイトが最後に公開された時期を示します。
amazon:inspector:sbom_scanner:fixed_version: *component_bom_ref*	特定の脆弱性に対して、指定されたコンポーネントの修正バージョンを提供します。
amazon:inspector:sbom_scanner:high_vulnerabilities	SBOM で見つかった重大度が高い脆弱性の総数。
amazon:inspector:sbom_scanner:info	特定のコンポーネントのスキャンコンテキストを提供します。例:「コンポーネントがスキャンされました: 脆弱性は見つかりませんでした」
amazon:inspector:sbom_scanner:is_malicious	影響を受けるコンポーネントを OpenSSF が悪意のあるものとして識別するかどうかを示します。
amazon:inspector:sbom_scanner:low_vulnerabilities	SBOM で見つかった重大度が低い脆弱性の総数。
amazon:inspector:sbom_scanner:medium_vulnerabilities	SBOM で見つかった重大度が中程度の脆弱性の総数。

プロパティ	説明
amazon:inspector:sbom_scanner:path	対象パッケージ情報を生成するファイルへのパス。
amazon:inspector:sbom_scanner:priority	特定の脆弱性を修正するための推奨優先度。値は、降順で「IMMEDIATE」、「URGENT」、「MODERATE」、および「STANDARD」です。
amazon:inspector:sbom_scanner:priority_intelligence	特定の脆弱性の優先度を決定するために使用されるインテリジェンスの品質。値には、「VERIFIED」または「UNVERIFIED」が含まれます。
amazon:inspector:sbom_scanner:warning	特定のコンポーネントがスキャンされなかった理由のコンテキストを提供します。例:「コンポーネントがスキップされました: purl が提供されていません」

amazon:inspector:sbom_generator 名前空間の分類

Amazon Inspector SBOM Generator は amazon:inspector:sbom_generator 名前空間を使用し、以下のプロパティを持ちます。

プロパティ	説明
amazon:inspector:sbom_generator:cpu_architecture	インベントリされるシステムの CPU アーキテクチャ (x86_64)。
amazon:inspector:sbom_generator:ec2:instance_id	Amazon EC2 インスタンス ID。
amazon:inspector:sbom_generator:live_patching_enabled	Amazon EC2 Amazon でライブパッチ適用が有効になっているかどうかを示すブール値 Linux。

プロパティ	説明
amazon:inspector:sbom_generator:live_patched_cves	Amazon EC2 Amazon でのライブパッチによってパッチが適用された CVEs のリストLinux。
amazon:inspector:sbom_generator:dockerfile_finding: *inspector_finding_id*	コンポーネントでの Amazon Inspector の検出結果が Dockerfile チェックに関連していることを示します。
amazon:inspector:sbom_generator:image_id	コンテナイメージ設定ファイル (イメージ ID とも呼ばれます) に属するハッシュ。
amazon:inspector:sbom_generator:image_arch	コンテナイメージのアーキテクチャ。
amazon:inspector:sbom_generator:image_author	コンテナイメージの作成者。
amazon:inspector:sbom_generator:image_docker_version	コンテナイメージの構築に使用される Docker バージョン。
amazon:inspector:sbom_generator:is_duplicate_package	対象パッケージが複数のファイルスキャナーによって検出されたことを示します。
amazon:inspector:sbom_generator:duplicate_purl	別のスキャナーによって見つかった重複パッケージ PURL を示します。
amazon:inspector:sbom_generator:kernel_name	インベントリ対象のシステムのカーネル名。
amazon:inspector:sbom_generator:kernel_version	インベントリ対象のシステムのカーネルバージョン。
amazon:inspector:sbom_generator:kernel_component	サブジェクトパッケージがカーネルコンポーネントであるかどうかを示すブール値
amazon:inspector:sbom_generator:running_kernel	サブジェクトパッケージが実行中のカーネルであるかどうかを示すブール値

プロパティ	説明
amazon:inspector:sbom_generator:layer_diff_id	非圧縮コンテナイメージレイヤーのハッシュ。
amazon:inspector:sbom_generator:replaced_by	現在のGoモジュールを置き換える値。
amazon:inspector:sbom_generator:os_hostname	インベントリ対象のシステムのホスト名。
amazon:inspector:sbom_generator:source_file_scanner	パッケージ情報を含むファイルを検出したスキャナー。例: /var/lib/dpkg/status
amazon:inspector:sbom_generator:source_package_collector	特定のファイルからパッケージ名とバージョンを抽出したコレクター。
amazon:inspector:sbom_generator:source_path	対象パッケージ情報が抽出されたファイルへのパス。
amazon:inspector:sbom_generator:file_size_bytes	特定のアーティファクトのファイルサイズを示します。
amazon:inspector:sbom_generator:unresolved_version	パッケージマネージャーによって解決されていないバージョン文字列を示します。
amazon:inspector:sbom_generator:experimental:transitive_dependency	パッケージマネージャーからの間接的な依存関係を示します。

Amazon Inspector スキャンを CI/CD パイプラインに統合する

Amazon Inspector を CI/CD に統合した場合、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を利用してコンテナイメージの脆弱性レポートが作成されます。Amazon Inspector SBOM Generator は、アーカイブ、コンテナイメージ、ディレクトリ、ローカルシステム、コンパイルされた Go および Rust バイナリのソフトウェア部品表 (SBOM) を作成します。Amazon Inspector スキャン API は SBOM をスキャンして、検出された脆弱性に関する詳細を含むレポートを作成します。Amazon Inspector コンテナイメージスキャンを CI/CD パイプラインと統合して、ソフトウェアの脆弱性をスキャンし、脆弱性レポートを生成できます。これにより、デプロイ前にリスクを調査して修正できます。CI/CD 統合を設定するには、プラグインを使用するか、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を使用してカスタム CI/CD 統合を作成できます。

トピック

- プラグインによる統合
- カスタム統合
- Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する
- Amazon Inspector Dockerfile チェック
- Amazon Inspector スキャンを使用したカスタム CI/CD パイプライン統合の作成
- Amazon Inspector Jenkins プラグインを使用する
- Amazon Inspector TeamCity プラグインを使用する
- Amazon Inspector での GitHub アクションの使用
- GitLab コンポーネントでの Amazon Inspector の使用
- Amazon Inspector での CodeCatalyst アクションの使用
- CodePipeline での Amazon Inspector スキャンアクションの使用

プラグインによる統合

Amazon Inspector には、サポートされている CI/CD ソリューション用のプラグインが用意されています。これらのプラグインをそれぞれのマーケットプレイスからインストールし、それらを使用して Amazon Inspector スキャンをパイプラインのビルドステップとして追加できます。プラグインのビ

ルドステップでは、指定したイメージに対して Amazon Inspector SBOM Generator を実行し、生成された SBOM に対して Amazon Inspector スキャン API を実行します。

プラグインによる Amazon Inspector の CI/CD への統合がどのように機能するか、その概要を以下に示します。

1. Amazon Inspector スキャン API へのアクセスを許可する AWS アカウント ように を設定します。手順については、「Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する」を参照してください。
2. マーケットプレイスから Amazon Inspector プラグインをインストールします。
3. Amazon Inspector SBOM Generator バイナリをインストールして設定します。手順については、「Amazon Inspector SBOM Generator」を参照してください。
4. Amazon Inspector スキャンを CI/CD パイプラインのビルドステップとして追加し、スキャンを設定します。
5. ビルドを実行すると、プラグインはコンテナイメージを入力として受け取り、そのイメージに対して Amazon Inspector SBOM Generator を実行して CycloneDX と互換がある SBOM を生成します。
6. そこから、プラグインは生成された SBOM を Amazon Inspector スキャン API のエンドポイントに送信します。このエンドポイントは、各 SBOM コンポーネントの脆弱性を評価します。
7. Amazon Inspector スキャン API のレスポンスは、CSV、SBOM、JSON、および HTML 形式の脆弱性レポートに変換されます。レポートには、Amazon Inspector が検出したあらゆる脆弱性に関する詳細が含まれています。

サポートされている CI/CD ソリューション

Amazon Inspector は現在、以下の CI/CD ソリューションをサポートしています。プラグインを使用して CI/CD の統合をセットアップする詳細な手順については、CI/CD ソリューションに適したプラグインを以下から選択してください。

- Jenkins プラグイン
- TeamCity プラグイン
- GitHub アクション

カスタム統合

Amazon Inspector から CI/CD ソリューション用のプラグインが提供されていない場合は、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を組み合わせて、独自にカスタムした CI/CD 統合を作成できます。カスタム統合では、Amazon Inspector SBOM Generator で利用可能なオプションを使用してスキャンを微調整することもできます。

カスタムによる Amazon Inspector の CI/CD への統合がどのように機能するか、その概要を以下に示します。

1. Amazon Inspector スキャン API へのアクセスを許可する AWS アカウント ように を設定します。手順については、「Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する」を参照してください。
2. Amazon Inspector SBOM Generator バイナリをインストールして設定します。手順については、「Amazon Inspector SBOM Generator」を参照してください。
3. Amazon Inspector SBOM Generator を使用し、コンテナイメージに対して CycloneDX と互換性のある SBOM を生成します。
4. 生成された SBOM に Amazon Inspector スキャン API を使用して、脆弱性レポートを作成します。

カスタム統合を設定する手順については、「Amazon Inspector スキャンを使用したカスタム CI/CD パイプライン統合の作成」を参照してください。

Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する

Amazon Inspector CI/CD 統合を使用するには、 AWS アカウントにサインアップする必要があります。には、CI/CD パイプラインに Amazon Inspector スキャン API へのアクセスを許可する IAM ロール AWS アカウント が必要です。以下のトピックのタスクを完了して、 にサインアップし AWS アカウント、管理者ユーザーを作成し、CI/CD 統合用の IAM ロールを設定します。

> **Note**
>
> に既にサインアップしている場合は AWS アカウント、「」に進むことができますCI/CD への統合のための IAM ロールを設定する。

トピック

- にサインアップする AWS アカウント
- 管理アクセスを持つユーザーを作成する
- CI/CD への統合のための IAM ロールを設定する

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM Identity Center、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 AWS IAM Identity Center ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの結合」を参照してください。

CI/CD への統合のための IAM ロールを設定する

Amazon Inspector のスキャンを CI/CD パイプラインに統合するには、ソフトウェア部品表 (SBOM) をスキャンする Amazon Inspector スキャン API へのアクセスを許可する IAM ポリシーを作成する必要があります。次に、そのポリシーを IAM ロールにアタッチし、アカウントが Amazon Inspector スキャン API を実行できるようにします。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで、[ポリシー]、[ポリシーを作成] の順に選択します。
3. [ポリシーエディタ] で [JSON] を選択し、以下のステートメントを貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "inspector-scan:ScanSbom",
            "Resource": "*"
        }
    ]
}
```

4. [次へ] を選択します。
5. ポリシーに名前を付けて (例えば `InspectorCICDscan-policy`)、必要に応じて説明を入力してから、[ポリシーの作成] を選択します。このポリシーは、次の手順で作成するロールにアタッチされます。
6. IAM コンソールのナビゲーションペインで、[ロール]、[新しいロールの作成] の順に選択します。
7. [信頼されたエンティティを選択] で [カスタム信頼ポリシー] を選択し、以下のポリシーを入力します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{ACCOUNT_ID}:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

8. [次へ] を選択します。
9. [許可を追加] で、前に作成したたポリシーを検索して選択し、[次へ] を選択します。
10. ロールに名前を付けて (例えば `InspectorCICDscan-role`) 、必要に応じて説明を追加して、`Create Role` を選択します。

Amazon Inspector Dockerfile チェック

このセクションでは、Amazon Inspector SBOM Generator を使用して Dockerfiles および Docker コンテナイメージをスキャンし、セキュリティの脆弱性を引き起こす設定ミスがないかどうか確認する方法について説明します。

トピック

- Sbomgen Dockerfile チェックの使用
- サポートされている Dockerfile チェック

Sbomgen Dockerfile チェックの使用

Dockerfile チェックは、`Dockerfile` または `*.Dockerfile` という名前のファイルが検出されたとき、および Docker イメージがスキャンされたときに自動的に実行されます。

`--skip-scanners dockerfile` 引数を使用して Dockerfile チェックを無効にすることができます。また、Dockerfile チェックを OS やサードパーティーパッケージなどの利用可能なスキャナーと組み合わせることもできます。

Docker チェックコマンドの例

次のコマンド例は、Dockerfiles と Docker コンテナイメージ、および OS とサードパーティーパッケージの SBOM を生成する方法を示しています。

```
# generate SBOM only containing Docker checks for Dockerfiles in a local directory
./inspector-sbomgen directory --path ./project/ --scanners dockerfile

# generate SBOM for container image will by default include Dockerfile checks
./inspector-sbomgen container --image image:tag

# generate SBOM only containing Docker checks for specific Dockerfiles and Alpine,
 Debian, and Rhel OS packages in a local directory
/inspector-sbomgen directory --path ./project/ --scanners dockerfile,dpkg,alpine-
apk,rhel-rpm

# generate SBOM only containing Docker checks for specific Dockerfiles in a local
 directory
./inspector-sbomgen directory --path ./project/ --skip-scanners dockerfile
```

ファイルコンポーネントの例

以下は、ファイルコンポーネントの Dockerfile 検出結果の例です。

```
{
     "bom-ref": "comp-2",
      "name": "dockerfile:data/docker/Dockerfile",
      "properties": [
        {
          "name": "amazon:inspector:sbom_scanner:dockerfile_finding:IN-DOCKER-001",
          "value": "affected_lines:27-27"
        }
      ],
      "type": "file"
    },
```

脆弱性レスポンスコンポーネントの例

以下は、脆弱性レスポンスコンポーネントの Dockerfile 検出結果の例です。

```
{
      "advisories": [
        {
          "url": "https://docs.docker.com/develop/develop-images/instructions/"
        }
      ],
      "affects": [
        {
          "ref": "comp-2"
        }
      ],
      "analysis": {
        "state": "in_triage"
      },
      "bom-ref": "vuln-13",
      "created": "2024-03-27T14:36:39Z",
      "description": "apt-get layer caching: Using apt-get update alone in a RUN
 statement causes caching issues and subsequent apt-get install instructions to fail.",
      "id": "IN-DOCKER-001",
      "ratings": [
        {
          "method": "other",
          "severity": "info",
          "source": {
            "name": "AMAZON_INSPECTOR",
            "url": "https://aws.amazon.com/inspector/"
          }
        }
      ],
      "source": {
        "name": "AMAZON_INSPECTOR",
        "url": "https://aws.amazon.com/inspector/"
      },
      "updated": "2024-03-27T14:36:39Z"
    },
```

> **Note**
>
> `--scan-sbom` フラグなしで Sbomgen を呼び出すと、未加工の Dockerfile 検出結果のみを表示できます。

サポートされている Dockerfile チェック

Sbomgen Dockerfile チェックは、以下に対してサポートされています。

- Sudo バイナリパッケージ
- Debian APT ユーティリティ
- ハードコーディングされたシークレット
- ルートコンテナ
- ランタイムの弱体化コマンドフラグ
- ランタイムの弱体化環境変数

これらの各 Dockerfile チェックには、次のトピックの上部に記載されている、対応する重要度評価があります。

> **Note**
> 以下のトピックで説明する推奨事項は、業界のベストプラクティスに基づいています。

Sudo バイナリパッケージ

> **Note**
> このチェックの重要度評価は、参考です。

Sudo バイナリパッケージには、予測不可能な TTY とシグナル転送動作があるため、これをインストールまたは使用することはお勧めしません。詳細については、Docker Docs ウェブサイトの「User」を参照してください。ユースケースで Sudo バイナリパッケージと同様の機能が必要な場合は、Gosu を使用することをお勧めします。

Debian APT ユーティリティ

> **Note**
> このチェックの重要度評価は、高です。

以下は、Debian APT ユーティリティを使用するためのベストプラクティスです。

キャッシュの問題を回避するために 1 つの Run ステートメントに apt-get コマンドを組み合わせる

Docker コンテナ内の単一の RUN ステートメントに `apt-get` コマンドを組み合わせることをお勧めします。単独で `apt-get update` を使用すると、キャッシュの問題が発生し、その後の `apt-get install` の手順が失敗します。詳細については、Docker Docs ウェブサイトの「apt-get」を参照してください。

> **Note**
>
> Docker コンテナソフトウェアが古い場合、説明されているキャッシュ動作は Docker コンテナコンテナ内でも発生する可能性があります。

非インタラクティブ方式での APT コマンドラインユーティリティの使用

APT コマンドラインユーティリティをインタラクティブに使用することをお勧めします。APT コマンドラインユーティリティはエンドユーザーツールとして設計されており、その動作はバージョンごとに変わります。詳細については、Debian ウェブサイトの「Script Usage and differences from other APT tools」を参照してください。

ハードコーディングされたシークレット

> **Note**
>
> このチェックの重要度評価は、緊急です。

Dockerfile の機密情報はハードコーディングされたシークレットと見なされます。以下のハードコーディングされたシークレットは、Sbomgen Docker ファイルチェックを通じて識別できます。

- AWS アクセスキー IDs – `AKIAIOSFODNN7EXAMPLE`
- AWS シークレットキー – `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
- DockerHub 個人用のアクセストークン – `dckr_pat_thisisa27charexample1234567`
- GitHub 個人用のアクセストークン – `ghp_examplev61wY7Pj1YnotrealUoY123456789`
- GitLab 個人用のアクセストークン – `glpat-12345example12345678`

ルートコンテナ

> **Note**
>
> このチェックの重要度マーカーは、参考です。

ルート権限なしで Docker コンテナを実行することをお勧めします。ルート権限なしでは実行できないコンテナ化されたワークロードの場合、最小限の権限の原則を使用してアプリケーションを構築することをお勧めします。詳細については、Docker Docs ウェブサイトの「User」を参照してください。

ランタイムの弱体化環境変数

> **Note**
>
> このチェックの重要度評価は、高です。

いくつかのコマンドラインユーティリティまたはプログラミング言語ランタイムは、安全なデフォルト設定のバイパスをサポートしているため、安全でないメソッドを通じた実行が許可されます。

NODE_TLS_REJECT_UNAUTHORIZED=0

`NODE_TLS_REJECT_UNAUTHORIZED` を `0` に設定して Node.js プロセスを実行すると、TLS 証明書の検証は無効になります。詳細については、Node.js ウェブサイトの「NODE_TLS_REJECT_UNAUTHORIZED=0」を参照してください。

GIT_SSL_NO_VERIFY=*

`GIT_SSL_NO_VERIFY` を設定して git コマンドラインプロセスを実行すると、Git は TLS 証明書の検証をスキップします。詳細については、Git ウェブサイトの「Environment variables」を参照してください。

PIP_TRUSTED_HOST=*

`PIP_TRUSTED_HOST` を設定して Python pip コマンドラインプロセスを実行すると、Pip は指定されたドメインでの TLS 証明書の検証をスキップします。詳細については、Pip ウェブサイトの「--trusted-host」を参照してください。

NPM_CONFIG_STRICT_SSL=false

`NPM_CONFIG_STRICT_SSL` を false に設定して Node.js npm コマンドラインプロセスを実行すると、Node Package Manager (npm) ユーティリティは TLS 証明書を検証せずに NPM レジストリに接続します。詳細については、npm Docs ウェブサイトの「strict-ssl」を参照してください。

ランタイムの弱体化コマンドフラグ

> **Note**
>
> このチェックの重要度評価は、高です。

ランタイムの弱体化環境変数と同様に、いくつかのコマンドラインユーティリティやプログラミング言語ランタイムは、安全なデフォルト設定をバイパスすることをサポートしているため、安全でないメソッドを通じた実行が許可されます。

`npm --strict-ssl=false`

`--strict-ssl=false` フラグを使用して Node.js npm コマンドラインプロセスを実行すると、Node Package Manager (npm) ユーティリティは TLS 証明書を検証せずに NPM レジストリに接続します。詳細については、npm Docs ウェブサイトの「strict-ssl」を参照してください。

`apk --allow-untrusted`

`--allow-untrusted` フラグを使用して Alpine Package Keeper ユーティリティを実行すると、`apk` は、署名がないか信頼できない署名を持つパッケージをインストールします。詳細については、Apline ウェブサイトで以下のリポジトリを参照してください。

`apt-get --allow-unauthenticated`

`--allow-unauthenticated` フラグを使用して Debian `apt-get` パッケージユーティリティを 実行した場合、`apt-get` はパッケージの有効性を確認しません。詳細については、Debian ウェブサイトの「APT-Get(8)」を参照してください。

`pip --trusted-host`

`--trusted-host` フラグを使用して Python pip ユーティリティを実行すると、指定されたホスト名は TLS 証明書の検証をバイパスします。詳細については、Pip ウェブサイトの「--trusted-host」を参照してください。

`rpm --nodigest, --nosignature, --noverify, --nofiledigest`

`--nodigest`、`--nosignature`、`--noverify`、および `--nofiledigest` フラグを使用して RPM ベースのパッケージマネージャー `rpm` を実行すると、RPM パッケージマネージャーはパッケージのインストール時にパッケージヘッダー、署名、またはファイルを検証しません。詳細については、RPM ウェブサイトの「RPM manual page」を参照してください。

`yum-config-manager --setopt=sslverify false`

`--setopt=sslverify` フラグを false に設定して RPM ベースのパッケージマネージャー `yum-config-manager` を実行すると、YUM パッケージマネージャーは TLS 証明書を検証しません。詳細については、Man7 ウェブサイトの「YUM manual page」を参照してください。

`yum --nogpgcheck`

`--nogpgcheck` フラグを使用して RPM ベースのパッケージマネージャー `yum` を実行すると、YUM パッケージマネージャーはパッケージの GPG 署名の確認をスキップします。詳細については、Man7 ウェブサイトの「yum(8)」を参照してください。

`curl --insecure, curl -k`

`--insecure` または `-k` フラグを使用して `curl` を実行すると、TLS 証明書の検証は無効になります。デフォルトでは、`curl` が行うすべての安全な接続は、転送が実行される前に安全であることが検証されます。このオプションでは、`curl` は検証ステップをスキップし、チェックせずに続行します。詳細については、Curl ウェブサイトの「Curl manual page」を参照してください。

`wget --no-check-certificate`

`--no-check-certificate` フラグを使用して `wget` を実行すると、TLS 証明書の検証は無効になります。詳細については、GNU ウェブサイトの「Wget manual page」を参照してください。

コンテナ内の OS パッケージデータベースの削除チェック

> **Note**
>
> このチェックの重要度評価は、参考です。

オペレーティングシステムパッケージデータベースを削除すると、コンテナイメージのソフトウェアの完全なインベントリをスキャンできなくなります。これらのデータベースは、コンテナのビルドステップ中にそのままにしておく必要があります。

OS パッケージデータベースの削除チェックは、次のパッケージマネージャーでサポートされています。

アルパインパッケージキーパー (APK)

インストールされたソフトウェアの APK パッケージマネージャーを利用するコンテナイメージは、ビルド中に APK システムファイルが削除されないようにする必要があります。詳細については、Arch Linuxウェブサイトの APK manpages システムファイルのドキュメントを参照してください。

Debian パッケージマネージャー (DPKG)

Debian、Ubuntu、Distroless ベースのイメージなど、DPKG パッケージマネージャーを利用するコンテナは、コンテナの構築中に DPKG データベースが削除されないようにする必要があります。詳細については、 Ubuntuウェブサイトの「DPKG manpages system files documentation」を参照してください。

RPM パッケージマネージャー (RPM)

Amazon Linux や Red Hat Enterprise Linux など、RPM Package Manager (yum/dnf) を利用するコンテナは、コンテナの構築中に RPM データベースが削除されないようにする必要があります。詳細については、RPM ウェブサイトの「RPM 管理システムファイルのドキュメント」を参照してください。

Amazon Inspector スキャンを使用したカスタム CI/CD パイプライン統合の作成

Amazon Inspector CI/CD プラグインが CI/CD ソリューションで使用できる場合は、Amazon Inspector CI/CD プラグインを使用することをお勧めします。Amazon Inspector CI/CD プラグインが CI/CD ソリューションに利用できない場合は、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API の組み合わせを使用して、カスタム CI/CD 統合を作成できます。次の手順では、Amazon Inspector スキャンとのカスタム CI/CD パイプライン統合を作成する方法について説明します。

> **Tip**
>
> 1 つのコマンドで SBOM を生成してスキャンする場合は、Amazon Inspector SBOM Generator (Sbomgen) を使用してステップ 3 とステップ 4 をスキップできます。

ステップ 1. の設定 AWS アカウント

Amazon Inspector スキャン API へのアクセス AWS アカウント を提供する を設定します。詳細については、「Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する」を参照してください。

ステップ 2. Sbomgen バイナリのインストール

Sbomgen バイナリをインストールして設定します。詳細については、「Sbomgen のインストール」 をご参照ください。

ステップ 3. Sbomgen を使用する

Sbomgen を使用して、スキャンするコンテナイメージの SBOM ファイルを作成します。

次の例を使用できます。*image:id* をスキャンするイメージの名前に置き換えます。*sbom_path.json* を SBOM 出力を保存する場所に置き換えます。

例

```
./inspector-sbomgen container --image image:id -o sbom_path.json
```

ステップ 4. Amazon Inspector スキャン API の呼び出し

`inspector-scan` API を呼び出して、生成された SBOM をスキャンし、脆弱性レポートを提供します。

次の例を使用できます。*sbom_path.json* を有効な CycloneDX 互換 SBOM ファイルの場所に置き換えます。*ENDPOINT* を、現在認証されている の API エンドポイントに置き換え AWS リージョンます。*REGION* を対応するリージョンに置き換えます。

例

```
aws inspector-scan scan-sbom --sbom file://sbom_path.json --endpoint
ENDPOINT-URL --region REGION
```

AWS リージョン および エンドポイントの完全なリストについては、「リージョンとエンドポイント」を参照してください。

(オプション) ステップ 5. 1 つのコマンドでの SBOM の生成とスキャン

> **Note**
>
> このステップを完了するのは、ステップ 3 とステップ 4 をスキップした場合のみです。

`--scan-bom` フラグを使用して、単一のコマンドで SBOM を生成、スキャンします。

次の例を使用できます。*`image:id`* を、スキャンするイメージの名前に置き換えます。*`profile`* を、対応するプロファイルに置き換えます。*`REGION`* を対応するリージョンに置き換えます。*`/tmp/scan.json`* を、tmp ディレクトリ内の scan.json ファイルの場所に置き換えます。

例

```
./inspector-sbomgen container --image image:id --scan-sbom --aws-profile
profile --aws-region REGION -o /tmp/scan.json
```

AWS リージョン および エンドポイントの完全なリストについては、「リージョンとエンドポイント」を参照してください。

API 出力形式

Amazon Inspector スキャン API は、脆弱性レポートを CycloneDX 1.5 形式で出力することも、Amazon Inspector 検出結果 JSON 形式で出力することもできます。デフォルトは `--output-format` フラグを使用して変更できます。

CycloneDX 1.5 形式の出力例

```
{
  "status": "SBOM parsed successfully, 1 vulnerabilities found",
  "sbom": {
    "bomFormat": "CycloneDX",
    "specVersion": "1.5",
    "serialNumber": "urn:uuid:0077b45b-ff1e-4dbb-8950-ded11d8242b1",
    "metadata": {
      "properties": [
        {
          "name": "amazon:inspector:sbom_scanner:critical_vulnerabilities",
          "value": "1"
        },
        {
```

```
          "name": "amazon:inspector:sbom_scanner:high_vulnerabilities",
          "value": "0"
        },
        {
          "name": "amazon:inspector:sbom_scanner:medium_vulnerabilities",
          "value": "0"
        },
        {
          "name": "amazon:inspector:sbom_scanner:low_vulnerabilities",
          "value": "0"
        }
      ],
      "tools": [
        {
          "name": "CycloneDX SBOM API",
          "vendor": "Amazon Inspector",
          "version": "empty:083c9b00:083c9b00:083c9b00"
        }
      ],
      "timestamp": "2023-06-28T14:15:53.760Z"
    },
    "components": [
      {
        "bom-ref": "comp-1",
        "type": "library",
        "name": "log4j-core",
        "purl": "pkg:maven/org.apache.logging.log4j/log4j-core@2.12.1",
        "properties": [
          {
            "name": "amazon:inspector:sbom_scanner:path",
            "value": "/home/dev/foo.jar"
          }
        ]
      }
    ],
    "vulnerabilities": [
      {
        "bom-ref": "vuln-1",
        "id": "CVE-2021-44228",
        "source": {
          "name": "NVD",
          "url": "https://nvd.nist.gov/vuln/detail/CVE-2021-44228"
        },
        "references": [
```

```
          {
            "id": "SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720",
            "source": {
              "name": "SNYK",
              "url": "https://security.snyk.io/vuln/SNYK-JAVA-
ORGAPACHELOGGINGLOG4J-2314720"
            }
          },
          {
            "id": "GHSA-jfh8-c2jp-5v3q",
            "source": {
              "name": "GITHUB",
              "url": "https://github.com/advisories/GHSA-jfh8-c2jp-5v3q"
            }
          }
        ],
        "ratings": [
          {
            "source": {
              "name": "NVD",
              "url": "https://www.first.org/cvss/v3-1/"
            },
            "score": 10.0,
            "severity": "critical",
            "method": "CVSSv31",
            "vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
          },
          {
            "source": {
              "name": "NVD",
              "url": "https://www.first.org/cvss/v2/"
            },
            "score": 9.3,
            "severity": "critical",
            "method": "CVSSv2",
            "vector": "AC:M/Au:N/C:C/I:C/A:C"
          },
          {
            "source": {
              "name": "EPSS",
              "url": "https://www.first.org/epss/"
            },
            "score": 0.97565,
            "severity": "none",
```

```
            "method": "other",
            "vector": "model:v2023.03.01,date:2023-06-27T00:00:00+0000"
          },
          {
            "source": {
              "name": "SNYK",
              "url": "https://security.snyk.io/vuln/SNYK-JAVA-
ORGAPACHELOGGINGLOG4J-2314720"
            },
            "score": 10.0,
            "severity": "critical",
            "method": "CVSSv31",
            "vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:H"
          },
          {
            "source": {
              "name": "GITHUB",
              "url": "https://github.com/advisories/GHSA-jfh8-c2jp-5v3q"
            },
            "score": 10.0,
            "severity": "critical",
            "method": "CVSSv31",
            "vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
          }
        ],
        "cwes": [
          400,
          20,
          502
        ],
        "description": "Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security
 releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages,
 and parameters do not protect against attacker controlled LDAP and other JNDI related
 endpoints. An attacker who can control log messages or log message parameters can
 execute arbitrary code loaded from LDAP servers when message lookup substitution is
 enabled. From log4j 2.15.0, this behavior has been disabled by default. From version
 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely
 removed. Note that this vulnerability is specific to log4j-core and does not affect
 log4net, log4cxx, or other Apache Logging Services projects.",
        "advisories": [
          {
            "url": "https://www.intel.com/content/www/us/en/security-center/advisory/
intel-sa-00646.html"
          },
```

```
          {
            "url": "https://support.apple.com/kb/HT213189"
          },
          {
            "url": "https://msrc-blog.microsoft.com/2021/12/11/microsofts-response-to-
cve-2021-44228-apache-log4j2/"
          },
          {
            "url": "https://logging.apache.org/log4j/2.x/security.html"
          },
          {
            "url": "https://www.debian.org/security/2021/dsa-5020"
          },
          {
            "url": "https://cert-portal.siemens.com/productcert/pdf/ssa-479842.pdf"
          },
          {
            "url": "https://www.oracle.com/security-alerts/alert-cve-2021-44228.html"
          },
          {
            "url": "https://www.oracle.com/security-alerts/cpujan2022.html"
          },
          {
            "url": "https://cert-portal.siemens.com/productcert/pdf/ssa-714170.pdf"
          },
          {
            "url": "https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/M5CSVUNV4HWZZXGOKNSK6L7RPM7BOKIB/"
          },
          {
            "url": "https://cert-portal.siemens.com/productcert/pdf/ssa-397453.pdf"
          },
          {
            "url": "https://cert-portal.siemens.com/productcert/pdf/ssa-661247.pdf"
          },
          {
            "url": "https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/VU57UJDCFIASIO35GC55JMKSRXJMCDFM/"
          },
          {
            "url": "https://www.oracle.com/security-alerts/cpuapr2022.html"
          },
          {
            "url": "https://twitter.com/kurtseifried/status/1469345530182455296"
```

```
          },
          {
            "url": "https://tools.cisco.com/security/center/content/
CiscoSecurityAdvisory/cisco-sa-apache-log4j-qRuKNEbd"
          },
          {
            "url": "https://lists.debian.org/debian-lts-announce/2021/12/msg00007.html"
          },
          {
            "url": "https://www.kb.cert.org/vuls/id/930724"
          }
        ],
        "created": "2021-12-10T10:15:00Z",
        "updated": "2023-04-03T20:15:00Z",
        "affects": [
          {
            "ref": "comp-1"
          }
        ],
        "properties": [
          {
            "name": "amazon:inspector:sbom_scanner:exploit_available",
            "value": "true"
          },
          {
            "name": "amazon:inspector:sbom_scanner:exploit_last_seen_in_public",
            "value": "2023-03-06T00:00:00Z"
          },
          {
            "name": "amazon:inspector:sbom_scanner:cisa_kev_date_added",
            "value": "2021-12-10T00:00:00Z"
          },
          {
            "name": "amazon:inspector:sbom_scanner:cisa_kev_date_due",
            "value": "2021-12-24T00:00:00Z"
          },
          {
            "name": "amazon:inspector:sbom_scanner:fixed_version:comp-1",
            "value": "2.15.0"
          }
        ]
      }
    ]
  }
```

```
}
```

Inspector 形式の出力例

```
                        {
  "status": "SBOM parsed successfully, 1 vulnerability found",
  "inspector": {
    "messages": [
      {
        "name": "foo",
        "purl": "pkg:maven/foo@1.0.0", // Will not exist in output if missing in sbom
        "info": "Component skipped: no rules found."
      }
    ],
    "vulnerability_count": {
      "critical": 1,
      "high": 0,
      "medium": 0,
      "low": 0
    },
    "vulnerabilities": [
      {
        "id": "CVE-2021-44228",
        "severity": "critical",
        "source": "https://nvd.nist.gov/vuln/detail/CVE-2021-44228",
        "related": [
          "SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720",
          "GHSA-jfh8-c2jp-5v3q"
        ],
        "description": "Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security
 releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages,
 and parameters do not protect against attacker controlled LDAP and other JNDI related
 endpoints. An attacker who can control log messages or log message parameters can
 execute arbitrary code loaded from LDAP servers when message lookup substitution is
 enabled. From log4j 2.15.0, this behavior has been disabled by default. From version
 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely
 removed. Note that this vulnerability is specific to log4j-core and does not affect
 log4net, log4cxx, or other Apache Logging Services projects.",
        "references": [
          "https://www.intel.com/content/www/us/en/security-center/advisory/intel-
sa-00646.html",
          "https://support.apple.com/kb/HT213189",
```

```
          "https://msrc-blog.microsoft.com/2021/12/11/microsofts-response-to-
cve-2021-44228-apache-log4j2/",
          "https://logging.apache.org/log4j/2.x/security.html",
          "https://www.debian.org/security/2021/dsa-5020",
          "https://cert-portal.siemens.com/productcert/pdf/ssa-479842.pdf",
          "https://www.oracle.com/security-alerts/alert-cve-2021-44228.html",
          "https://www.oracle.com/security-alerts/cpujan2022.html",
          "https://cert-portal.siemens.com/productcert/pdf/ssa-714170.pdf",
          "https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/M5CSVUNV4HWZZXGOKNSK6L7RPM7BOKIB/",
          "https://cert-portal.siemens.com/productcert/pdf/ssa-397453.pdf",
          "https://cert-portal.siemens.com/productcert/pdf/ssa-661247.pdf",
          "https://lists.fedoraproject.org/archives/list/package-
announce@lists.fedoraproject.org/message/VU57UJDCFIASIO35GC55JMKSRXJMCDFM/",
          "https://www.oracle.com/security-alerts/cpuapr2022.html",
          "https://twitter.com/kurtseifried/status/1469345530182455296",
          "https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-
sa-apache-log4j-qRuKNEbd",
          "https://lists.debian.org/debian-lts-announce/2021/12/msg00007.html",
          "https://www.kb.cert.org/vuls/id/930724"
        ],
        "created": "2021-12-10T10:15:00Z",
        "updated": "2023-04-03T20:15:00Z",
        "properties": {
          "cisa_kev_date_added": "2021-12-10T00:00:00Z",
          "cisa_kev_date_due": "2021-12-24T00:00:00Z",
          "cwes": [
            400,
            20,
            502
          ],
          "cvss": [
            {
              "source": "NVD",
              "severity": "critical",
              "cvss3_base_score": 10.0,
              "cvss3_base_vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H",
              "cvss2_base_score": 9.3,
              "cvss2_base_vector": "AC:M/Au:N/C:C/I:C/A:C"
            },
            {
              "source": "SNYK",
              "severity": "critical",
              "cvss3_base_score": 10.0,
```

```
              "cvss3_base_vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:H"
            },
            {
              "source": "GITHUB",
              "severity": "critical",
              "cvss3_base_score": 10.0,
              "cvss3_base_vector": "AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
            }
          ],
          "epss": 0.97565,
          "exploit_available": true,
          "exploit_last_seen_in_public": "2023-03-06T00:00:00Z"
        },
        "affects": [
          {
            "installed_version": "pkg:maven/org.apache.logging.log4j/log4j-
core@2.12.1",
            "fixed_version": "2.15.0",
            "path": "/home/dev/foo.jar"
          }
        ]
      }
    ]
  }
}
```


Amazon Inspector Jenkins プラグインを使用する

この Jenkins プラグインは、Amazon Inspector SBOM Generator バイナリと Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを作成します。これにより、デプロイ前にリスクを調査して修正できます。Amazon Inspector Jenkins プラグインを使用すると、Amazon Inspector の脆弱性スキャンを Jenkins パイプラインに追加することができます。検出された脆弱性の数と重大度に基づいて、パイプラインの実行に成功または失敗するように Amazon Inspector 脆弱性スキャンを設定できます。Jenkins プラグインの最新バージョンは、Jenkins のマーケットプレイス (https://plugins.jenkins.io/amazon-inspector-image-scanner/) で確認できます。次の手順は、Amazon Inspector Jenkins プラグインを設定する方法を示しています。

> **Important**
>
> 次の手順を完了する前に、プラグインを実行するために Jenkins をバージョン 2.387.3 以降にアップグレードする必要があります。

ステップ 1. のセットアップ AWS アカウント

Amazon Inspector スキャン API へのアクセスを許可する IAM ロール AWS アカウント を使用してを設定します。手順については、「Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する」を参照してください。

ステップ 2. Amazon Inspector Jenkins プラグインのインストール

次の手順は、Jenkins ダッシュボードから Amazon Inspector Jenkins プラグインをインストールする方法を示しています。

1. Jenkins ダッシュボードから、[Jenkins の管理] を選択し、[プラグインの管理] を選択します。
2. [利用可能] を選択します。
3. [利用可能] タブから [Amazon Inspector スキャン] を検索し、プラグインをインストールします。

(オプション) ステップ 3. Jenkins への Docker 認証情報の追加

> **Note**
>
> Docker イメージがプライベートリポジトリにある場合にのみ、Docker 認証情報を追加します。それ以外の場合は、この手順をスキップしてください。

次の手順は、Jenkins ダッシュボードから Jenkins に Docker 認証情報を追加する方法を示しています。

1. Jenkins ダッシュボードから、[Jenkins の管理]、[認証情報]、[システム] の順に選択します。
2. [グローバル認証情報]、[認証情報を追加] の順に選択します。
3. [種類] で、[パスワード付きのユーザー名] を選択します。

4. [スコープ] で、[グローバル (Jenkins、ノード、項目、すべての子項目など)] を選択します。
5. 詳細を入力し、[OK] を選択します。

(オプション) ステップ 4. AWS 認証情報を追加する

> **Note**
> IAM ユーザーに基づいて認証する場合にのみ、 AWS 認証情報を追加します。それ以外の場合は、この手順をスキップしてください。

次の手順では、Jenkinsダッシュボードから AWS 認証情報を追加する方法について説明します。

1. Jenkins ダッシュボードから、[Jenkins の管理]、[認証情報]、[システム] の順に選択します。
2. [グローバル認証情報]、[認証情報を追加] の順に選択します。
3. [種類] で、[AWS 認証情報] を選択します。
4. [アクセスキー ID] や [シークレットアクセスキー] などの詳細を入力し、[OK] を選択します。

ステップ 5. Jenkins スクリプトへの CSS サポートの追加

次の手順は、Jenkins スクリプトに CSS サポートを追加する方法を示しています。

1. Jenkins を再起動します。
2. ダッシュボードから、[Jenkins の管理]、[ノード]、[組み込みノード] [スクリプトコンソール] の順に選択します。
3. テキストボックスに行 `System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")` を追加し、[実行] を選択します。

ステップ 6。Amazon Inspector スキャンのビルドへの追加

Amazon Inspector スキャンをビルドに追加するには、プロジェクトにビルドステップを追加するか、Jenkins 宣言型パイプラインを使用します。

プロジェクトでのビルドステップの追加による Amazon Inspector スキャンのビルドへの追加

1. 設定ページで [構築ステップ] まで下にスクロールし、[構築ステップを追加] を選択します。次に、[Amazon Inspector スキャン] を選択します。
2. inspector-sbomgen のインストール方法は、自動 と 手動 の 2 つから選択します。自動オプションを使用すると、プラグインは最新バージョンをダウンロードできます。また、常に最新の機能、セキュリティ更新、バグ修正を確実に行うことができます。

   a. (オプション 1) [自動]を選択して、最新バージョンの inspector-sbomgen をダウンロードします。このオプションは、現在使用中のオペレーティングシステムと CPU アーキテクチャを自動的に検出します。

   b. (オプション 2) スキャン用に Amazon Inspector SBOM Generator バイナリを設定する場合は、[手動]を選択します。この方法を選択した場合は、以前にダウンロードしたバージョンの inspector-sbomgen への完全なパスを必ず指定してください。

   詳細については、「Amazon Inspector SBOM Generator」の「Amazon Inspector SBOM Generator (Sbomgen) のインストール」を参照してください。
3. Amazon Inspector スキャンのビルドステップの設定を完了するには、以下を実行します。

   a. [Image Id] を入力します。イメージはローカル、リモート、アーカイブされたもののいずれでもかまいません。イメージ名は Docker の命名規則に従う必要があります。エクスポートされたイメージを分析する場合は、予想される tar ファイルへのパスを指定します。イメージ ID のパスの例については、以下を参照してください。

      i. ローカルコンテナまたはリモートコンテナの場合: `NAME[:TAG|@DIGEST]`

      ii. tar ファイルの場合: `/path/to/image.tar`

   b. [AWS リージョン] を選択して、スキャンリクエストを送信します。

   c. (オプション) レポートアーティファクト名に、ビルドプロセス中に生成されたアーティファクトのカスタム名を入力します。これにより、それらを一意に識別および管理できます。

   d. (オプション) スキップファイルには、スキャンから除外するディレクトリを 1 つ以上指定します。サイズが原因でスキャンする必要のないディレクトリの場合は、このオプションを検討してください。

   e. (オプション) [Docker credentials] で Docker ユーザー名を選択します。これは、コンテナイメージがプライベートリポジトリにある場合にのみ行ってください。

f. (オプション) 以下のサポートされている AWS 認証方法を指定できます。

   i. (オプション) [IAM ロール] で、ロール ARN (arn:aws:iam::*AccountNumber*:role/*RoleName*) を指定します。

   ii. (オプション) AWS 認証情報の場合は、IAM ユーザーに基づいて認証する AWS 認証情報を指定します。

   iii. (オプション) [AWS プロファイル名] で、プロファイル名を使用して認証するプロファイルの名前を指定します。

g. (オプション) 脆弱性しきい値を有効にするを選択します。このオプションを使用すると、スキャンされた脆弱性が値を超えた場合にビルドが失敗するかどうかを判断できます。すべての値が `0` に等しい場合、スキャンされる脆弱性の数に関係なく、ビルドは成功します。EPSS スコアの値は 0 ~ 1 です。スキャンされた脆弱性が値を超えると、ビルドは失敗し、値を超える EPSS スコアを持つすべての CVEs がコンソールに表示されます。

4. [保存] を選択します。

Jenkins 宣言型パイプラインを使用した Amazon Inspector スキャンのビルドへの追加

Jenkins 宣言型パイプラインを使用して、自動または手動で Amazon Inspector スキャンをビルドに追加できます。

SBOMGen 宣言パイプラインを自動的にダウンロードするには

- Amazon Inspector スキャンをビルドに追加するには、次の構文例を使用します。Amazon Inspector SBOM Generator ダウンロードの任意の OS アーキテクチャに基づいて、*SBOMGEN_SOURCE* を linuxAmd64 または linuxArm64 に置き換えます。プライベートリポジトリを使用している場合は、*IMAGE_PATH* をイメージへのパス (*alpine:latest* など) に置き換え、*IAM_ROLE* をステップ 1 で設定した IAM ロールの ARN に置き換え、*ID* をDocker 認証情報 ID に置き換えます。オプションで脆弱性のしきい値を有効にし、重大度ごとに値を指定できます。

```
pipeline {
   agent any
   stages {
       stage('amazon-inspector-image-scanner') {
           steps {
               script {
               step([
```

```
                $class:
 'com.amazon.inspector.jenkins.amazoninspectorbuildstep.AmazonInspectorBuilder',
                sbomgenSource: 'SBOMGEN_SOURCE', // this can be linuxAmd64 or linuxArm64
                archivePath: 'IMAGE_PATH',
                awsRegion: 'REGION',
                iamRole: 'IAM ROLE',
                credentialId: 'Id', // provide empty string if image not in private
 repositories
                awsCredentialId: ''AWS ID;',
                awsProfileName: 'Profile Name',
                isThresholdEnabled: false,
                countCritical: 0,
                countHigh: 0,
                countLow: 10,
                countMedium: 5,
               ])
            }
        }
      }
   }
 }
```

SBOMGen 宣言パイプラインを手動でダウンロードするには

- Amazon Inspector スキャンをビルドに追加するには、次の構文例を使用します。*SBOMGEN_PATH* をステップ 3 でインストールした Amazon Inspector SBOM Generator へのパスに置き換え、*IMAGE_PATH* をイメージへのパスに置き換え (*alpine:latest* など)、*IAM_ROLE* をステップ 1 で設定した IAM ロールの ARN に置き換え、プライベートリポジトリを使用している場合は *ID* を Docker の認証情報 ID に置き換えます。オプションで脆弱性のしきい値を有効にし、重大度ごとに値を指定できます。

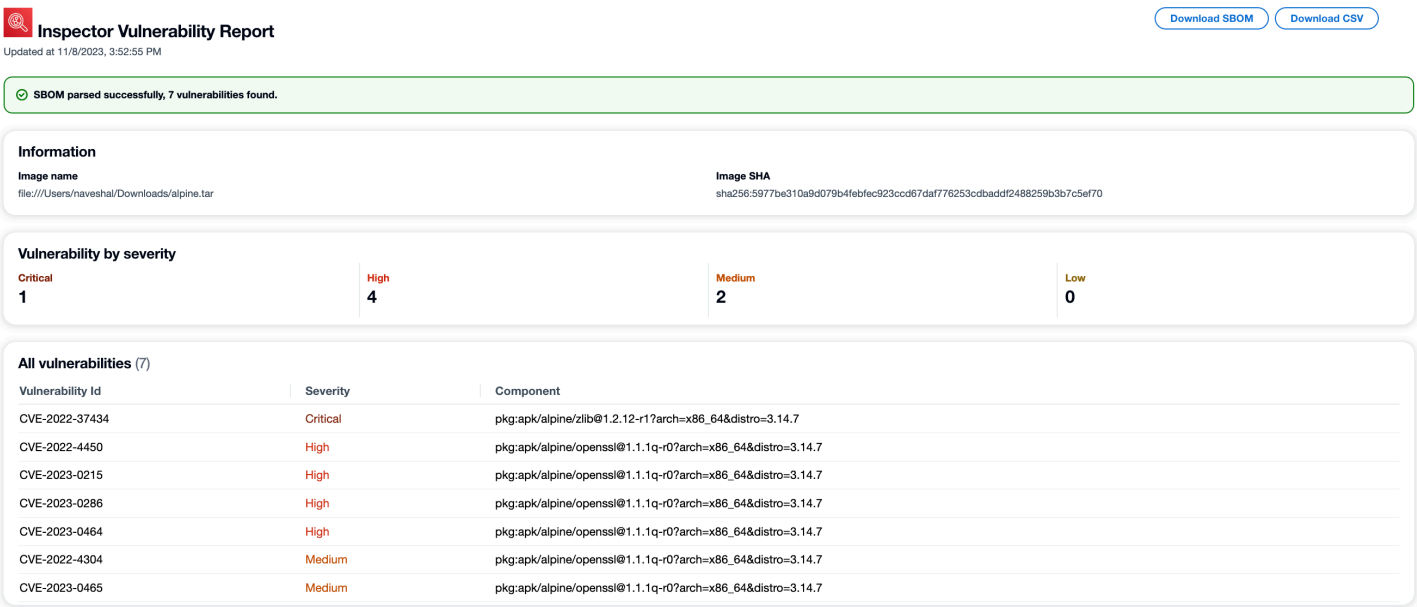
> **Note**
>
> Jenkins ディレクトリに Sbomgen を配置し、プラグインで Jenkins ディレクトリへのパスを指定します (*/opt/folder/arm64/inspector-sbomgen* など)。

```
pipeline {
    agent any
    stages {
```

```
        stage('amazon-inspector-image-scanner') {
            steps {
                script {
                step([
                $class:
 'com.amazon.inspector.jenkins.amazoninspectorbuildstep.AmazonInspectorBuilder',
                sbomgenPath: 'SBOMGEN_PATH',
                archivePath: 'IMAGE_PATH',
                awsRegion: 'REGION',
                iamRole: 'IAM ROLE',
                awsCredentialId: ''AWS ID;',
                credentialId: 'Id;', // provide empty string if image not in private
 repositories
                awsProfileName: 'Profile Name',
                isThresholdEnabled: false,
                countCritical: 0,
                countHigh: 0,
                countLow: 10,
                countMedium: 5,
               ])
            }
        }
      }
   }
 }
```

ステップ 7。Amazon Inspector の脆弱性レポートの確認

1. プロジェクトの新しいビルドを完了します。
2. ビルドが完了したら、結果から出力形式を選択します。HTML を選択した場合、JSON SBOM または CSV バージョンのレポートをダウンロードするオプションがあります。HTML レポートの例を以下に示します。



トラブルシューティング

Jenkins 向け Amazon Inspector スキャンプラグインを使用する際に発生する可能性がある一般的なエラーを以下に示します。

認証情報の読み込み失敗または sts 例外エラー

エラー:

```
InstanceProfileCredentialsProvider(): Failed to load credentials or sts
exception.
```

解決策

AWS アカウントの `aws_access_key_id`と `aws_secret_access_key` を取得します。`~/.aws/credentials` の `aws_access_key_id` と`aws_secret_access_key` をセットアップします。

ターボール、ローカル、またはリモートソースからイメージの読み込み失敗

エラー:

```
2024/10/16 02:25:17 [ImageDownloadFailed]: failed to load image from
tarball, local, or remote sources.
```

> **Note**
>
> このエラーが発生する可能性があるのは、Jenkins プラグインがコンテナイメージを読み取れず、コンテナイメージが Docker エンジンに見つからない場合、およびコンテナイメージがリモートコンテナレジストリに見つからない場合です。

解決策:

以下について確認してください。

- Jenkins プラグインユーザーには、スキャンするイメージに対する読み取りアクセス許可がある。
- スキャンするイメージは Docker エンジンにある。
- リモートイメージ URL が正しい。
- リモートレジストリに対して認証されている (該当する場合)。

Inspector-sbomgen パスエラー

エラー:

```
Exception:com.amazon.inspector.jenkins.amazoninspectorbuildstep.exception.Sbomge
There was an issue running inspector-sbomgen, is /opt/inspector/inspector-
sbomgen the correct path?
```

解決策:

この問題を解決するには、次の手順を完了します。

1. 正しい OS アーキテクチャ Inspector-sbomgen を Jenkins ディレクトリに配置します。詳細については、「Amazon Inspector SBOM Generator」を参照してください。
2. 以下のコマンドを使用して、バイナリに実行権限を付与します: `chmod +x inspector-sbomgen`。
3. `/opt/folder/arm64/inspector-sbomgen` など、プラグインで正しい Jenkins マシンパスを指定します。
4. 設定を保存し、Jenkins ジョブを実行します。

Amazon Inspector TeamCity プラグインを使用する

Amazon Inspector TeamCity プラグインは、Amazon Inspector SBOM Generator バイナリと Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを作成します。これにより、デプロイ前にリスクを調査して修正できます。Amazon Inspector TeamCity プラグインを使用すると、Amazon Inspector の脆弱性スキャンを TeamCity パイプラインに追加することができます。検出された脆弱性の数と重大度に基づいて、パイプラインの実行に成功または失敗するように Amazon Inspector 脆弱性スキャンを設定できます。Amazon Inspector TeamCity プラグインの最新バージョンは、TeamCity マーケットプレイス (https://plugins.jetbrains.com/plugin/23236-amazon-inspector-scanner) で確認できます。Amazon Inspector スキャンを CI/CD パイプラインに統合する方法については、「Amazon Inspector スキャンを CI/CD パイプラインに統合する」を参照してください。Amazon Inspector がサポートするオペレーティングシステムとプログラミング言語のリストについては、「サポートされているオペレーティングシステムとプログラミング言語」を参照してください。次の手順は、Amazon Inspector TeamCity プラグインを設定する方法を示しています。

1. をセットアップします AWS アカウント。
   - Amazon Inspector スキャン API へのアクセスを許可する IAM ロール AWS アカウント を使用して を設定します。手順については、「Amazon Inspector CI/CD 統合を使用するように AWS アカウントを設定する」を参照してください。
2. Amazon Inspector TeamCity プラグインをインストールする。
   a. ダッシュボードから、[Administration]、[Plugins] の順に移動します。
   b. [Amazon Inspector Scans] を検索します。
   c. プラグインをインストールします。
3. Amazon Inspector SBOM Generator をインストールする。
   - Amazon Inspector SBOM Generator バイナリを Teamcity サーバーディレクトリにインストールします。手順については、「Sbomgen のインストール」を参照してください。
4. Amazon Inspector スキャンのビルドステップをプロジェクトに追加する。
   a. 設定ページで [構築ステップ] まで下にスクロールし、[構築ステップを追加] を選択して [Amazon Inspector スキャン] を選択します。
   b. 以下の詳細を入力して、Amazon Inspector スキャンのビルドステップを設定します。
      - [ステップ名] を追加します。

- Amazon Inspector SBOM Generator のインストール方法は、[自動] と [手動] の 2 つから選択します。
  - [自動] の場合、システムと CPU アーキテクチャに基づいて最新バージョンの Amazon Inspector SBOM Generator を自動的にダウンロードします。
  - [手動] の場合、以前にダウンロードしたバージョンの Amazon Inspector SBOM Generator への完全なパスを指定する必要があります。

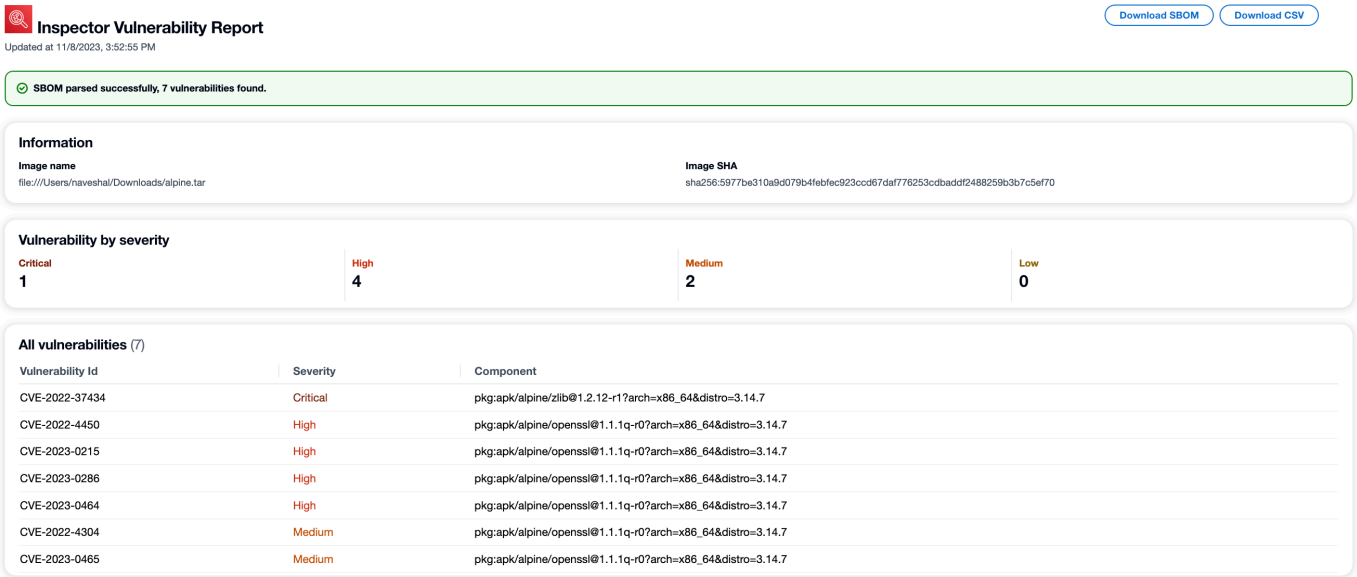
  詳細については、「Amazon Inspector SBOM Generator」の「Amazon Inspector SBOM Generator (Sbomgen) のインストール」を参照してください。
- [Image Id] を入力します。イメージはローカル、リモート、アーカイブされたもののいずれでもかまいません。イメージ名は Docker の命名規則に従う必要があります。エクスポートされたイメージを分析する場合は、予想される tar ファイルへのパスを指定します。イメージ ID のパスの例については、以下を参照してください。
  - ローカルコンテナまたはリモートコンテナの場合: `NAME[:TAG|@DIGEST]`
  - tar ファイルの場合: `/path/to/image.tar`
- [IAM ロール] には、手順 1 で設定したロールの ARN を入力します。
- [AWS リージョン] を選択して、スキャンリクエストを送信します。
- (オプション) [Docker Authentication] では、[Docker Username] と [Docker Password] を入力します。これは、コンテナイメージがプライベートリポジトリにある場合にのみ行ってください。
- (オプション) AWS 認証には、 AWS アクセスキー ID と AWS シークレットキーを入力します。これは、 AWS 認証情報に基づいて認証する場合にのみ実行します。
- (オプション) 重大度ごとに [Vulnerability thresholds] を指定します。スキャン中に指定した数を超えると、イメージのビルドは失敗します。値がすべて `0` の場合、見つかった脆弱性の数に関係なくビルドは成功します。

c. [Save] を選択します。

5. Amazon Inspector の脆弱性レポートを確認する。

   a. プロジェクトの新しいビルドを完了します。

   b. ビルドが完了したら、結果から出力形式を選択します。HTML を選択した場合、JSON SBOM または CSV バージョンのレポートをダウンロードするオプションがあります。HTML レポートの例を以下に示します。



Amazon Inspector での GitHub アクションの使用

GitHub actions で Amazon Inspector を使用して、Amazon Inspector 脆弱性スキャンを GitHub ワークフローに追加できます。これは、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを作成します。これにより、デプロイ前にリスクを調査して修正できます。Amazon Inspector の脆弱性スキャンは、検出された脆弱性の数と重要度に基づいてワークフローに合格するか、不合格になるよう設定できます。Amazon Inspector アクションの最新バージョンは、GitHub ウェブサイトで確認できます。Amazon Inspector スキャンを CI/CD パイプラインに統合する方法については、「Amazon Inspector スキャンを CI/CD パイプラインに統合する」を参照してください。Amazon Inspector がサポートするオペレーティングシステムとプログラミング言語のリストについては、「サポートされているオペレーティングシステムとプログラミング言語」を参照してください。

GitLab コンポーネントでの Amazon Inspector の使用

GitLab CI/CD コンポーネントで Amazon Inspector を使用して、GitLabプロジェクトに Amazon Inspector 脆弱性スキャンを追加できます。これは、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを作成します。これにより、デプロイ前にリスクを調査して修正できます。Amazon Inspector の脆弱性スキャンは、検出された脆弱性の数と重要度に基づいてワークフローに合格するか、不合格になるよう設定できます。Amazon Inspector コンポーネントの最新バージョンは、 GitLabウェブサイトで確認できます。Amazon

Inspector スキャンを CI/CD パイプラインに統合する方法については、「Amazon Inspector スキャンを CI/CD パイプラインに統合する」を参照してください。Amazon Inspector がサポートするオペレーティングシステムとプログラミング言語のリストについては、「サポートされているオペレーティングシステムとプログラミング言語」を参照してください。

Amazon Inspector での CodeCatalyst アクションの使用

Amazon Inspector と Amazon CodeCatalyst を使用して、Amazon Inspector の脆弱性スキャンを CodeCatalyst ワークフローに追加できます。これは、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを作成します。これにより、デプロイ前にリスクを調査して修正できます。Amazon Inspector の脆弱性スキャンは、検出された脆弱性の数と重要度に基づいてワークフローに合格するか、不合格になるよう設定できます。Amazon Inspector スキャンを CI/CD パイプラインに統合する方法については、「Amazon Inspector スキャンを CI/CD パイプラインに統合する」を参照してください。Amazon Inspector がサポートするオペレーティングシステムとプログラミング言語のリストについては、「サポートされているオペレーティングシステムとプログラミング言語」を参照してください。

CodePipeline での Amazon Inspector スキャンアクションの使用

ワークフローに脆弱性スキャンを追加 AWS CodePipeline することで、 で Amazon Inspector を使用できます。この統合では、Amazon Inspector SBOM Generator と Amazon Inspector スキャン API を活用して、ビルドの最後に詳細なレポートを生成します。統合は、デプロイ前にリスクを調査して修正するのに役立ちます。`InspectorScan` アクションは、オープンソースコードのセキュリティ脆弱性の検出と修正を自動化する CodePipeline のマネージドコンピューティングアクションです。このアクションは、GitHub や Bitbucket Cloud などのサードパーティーリポジトリのアプリケーションソースコード、またはコンテナアプリケーションのイメージで使用できます。詳細については、「 AWS CodePipeline ユーザーガイド」のInspectorScan invoke action reference」を参照してください。

AWS 環境の Amazon Inspector カバレッジの評価

Amazon Inspector コンソールのアカウント管理画面から AWS 環境の Amazon Inspector カバレッジを評価できます。これにより、アカウントとリソースの Amazon Inspector スキャンのステータスに関する詳細と統計が表示されます。

> **Note**
>
> 組織の委任管理者である場合、組織内のすべてのアカウントの詳細と統計情報を表示できます。

次の手順は、Amazon Inspector 環境のカバレッジを評価する方法を示しています。

AWS 環境の Amazon Inspector カバレッジを評価するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ナビゲーションペインから、[アカウント管理] を選択します。
3. カバレッジを確認するには、次のいずれかのタブを選択します。
   - [アカウント] を選択して、アカウントレベルのカバレッジを確認します。
   - [インスタンス] を選択して、Amazon Elastic Compute Cloud (Amazon EC2)インスタンスのカバレッジを確認します。
   - [コンテナレポジトリ] を選択して、Amazon Elastic Container Registry (Amazon ECR)リポジトリのカバレッジを確認します。
   - [コンテナイメージ] を選択して、Amazon ECR コンテナイメージのカバレッジを確認します。
   - [Lambda 関数] を選択して、Lambda 関数のカバレッジを確認します。

以下のトピックでは、これらの各タブで提供される情報について説明します。

トピック

- アカウントレベルのカバレッジを評価する
- Amazon EC2 インスタンスのカバレッジを評価する
- Amazon ECR リポジトリのカバレッジを評価する

- Amazon ECR コンテナイメージのカバレッジを評価する
- AWS Lambda 関数のカバレッジの評価

アカウントレベルのカバレッジを評価する

アカウントが組織の一部ではない場合、または組織の委任された Amazon Inspector 管理者アカウントではない場合、[アカウント] タブには、アカウントに関する情報と、アカウントのリソーススキャンのステータスが表示されます。このタブでは、アカウントのすべてまたは特定のタイプのリソースのみのスキャンをアクティブ化または非アクティブ化にできます。詳細については、「Amazon Inspector の自動スキャンタイプ」を参照してください。

アカウントが組織の委任された Amazon Inspector 管理者アカウントの場合、[アカウント] タブには組織内のアカウントの自動アクティベーション設定が表示され、組織内のすべてのアカウントが一覧表示されます。アカウントごとに、そのアカウントで Amazon Inspector がアクティブ化になっているかどうかを表示し、その場合は、そのアカウントでアクティブ化になっているリソーススキャンタイプがリストに表示されます。委任された管理者は、このタブを使用して組織の自動アクティベーション設定を変更できます。また、個々のメンバーアカウントの特定のタイプのリソーススキャンをアクティブ化または非アクティブ化することもできます。詳細については、「メンバーアカウントの Amazon Inspector スキャンをアクティブ化する」を参照してください。

Amazon EC2 インスタンスのカバレッジを評価する

[インスタンス] タブには、 AWS 環境内の Amazon EC2 インスタンスが表示されます。リストは次のタブにグループ分けされています。

- すべて — 環境内のすべてのインスタンスを表示します。ステータス列には、インスタンスの現在のスキャンステータスが表示されます。
- スキャン — Amazon Inspector が環境でアクティブにモニタリングおよびスキャンしているすべてのインスタンスを表示します。
- スキャンしない — Amazon Inspector が環境でモニタリングおよびスキャンしていないすべてのインスタンスを表示します。理由列には、Amazon Inspector がインスタンスをモニタリングおよびスキャンしていない理由が表示されます。

  EC2 インスタンスが [スキャンしない] タブに表示される理由はいくつかあります。Amazon Inspector は AWS Systems Manager (SSM) と SSM エージェントを使用して、EC2 インスタンスの脆弱性を自動的にモニタリングおよびスキャンします。インスタンスで SSM Agent が実行されていない場合、Systems Manager をサポートする AWS Identity and Access Management (IAM)

ロールがない場合、またはサポートされているオペレーティングシステムまたはアーキテクチャを実行していない場合、Amazon Inspector はインスタンスをモニタリングおよびスキャンできません。詳細については、「Amazon EC2 インスタンススキャン」を参照してください。

各タブで、Account 列はインスタンスを所有 AWS アカウント する を指定します。

[EC2 インスタンスタグ] — この列には、インスタンスに関連付けられているタグが表示され、インスタンスがタグによるスキャンから除外されているかどうかを判断できます。

[オペレーティングシステム] — この列には、オペレーティングシステムの種類 (`WINDOWS`、`MAC`、`LINUX`、または `UNKNOWN`) が表示されます。

[使用中の監視] – この列には、このインスタンスで Amazon Inspector がエージェントベースのスキャン方式を使用しているか、エージェントレスのスキャン方式を使用しているかが表示されます。

[最終スキャン日] - この列には、Amazon Inspector がそのリソースの脆弱性を最後にチェックした日時が表示されます。Amazon Inspector がスキャンを実行する頻度は、インスタンスのスキャンに使用されているスキャン方式によって異なります。

EC2 インスタンスに関するその他の詳細を確認するには、EC2 インスタンス列のリンクを選択します。次に、Amazon Inspector はインスタンスに関する詳細と、そのインスタンスに関する現在の検出結果を表示します。特定の検出結果の詳細を確認するには、[タイトル] 列のリンクを選択します。その詳細については、「Amazon Inspector 検出結果の詳細の表示」を参照してください。

Amazon EC2 インスタンスのステータス値のスキャン

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの場合、[ステータス] の値は以下のようになります：

- アクティブにモニタリング中 — Amazon Inspector ではインスタンスを継続的にモニタリングおよびスキャンしています。
- エージェントレスインスタンスストレージの制限の超過 – インスタンスにアタッチされたすべてのボリュームの合計サイズが 1200 GB を超える場合、またはインスタンスにアタッチされたボリュームが 8 個を超える場合、Amazon Inspector はこのステータスを使用します。
- エージェントレスインスタンス収集時間の制限の超過 – インスタンスでエージェントレススキャンを実行しようとして、Amazon Inspector がタイムアウトします。
- EC2 インスタンスが停止 — インスタンスが停止状態になったため、Amazon Inspector はインスタンスのスキャンを一時停止しました。既存の検出結果はすべて、インスタンスが終了するまで保

持されます。インスタンスが再起動されると、Amazon Inspector はインスタンスのスキャンを自動的に再開します。

- 内部エラー — Amazon Inspector がインスタンスをスキャンしようとしたときに内部エラーが発生しました。Amazon Inspector は自動的にエラーに対処し、できるだけ早くスキャンを再開します。
- インベントリなし — Amazon Inspector は、インスタンスをスキャンするソフトウェアアプリケーションインベントリを見つけることができませんでした。インスタンスの Amazon Inspector の関連付けが削除されているか、実行に失敗した可能性があります。

  この問題を修正するには、 を使用して、`InspectorInventoryCollection-do-not-delete`関連付けが存在し、その関連付けステータスが成功し AWS Systems Manager ていることを確認します。さらに、 AWS Systems Manager Fleet Manager を使用してインスタンスのソフトウェアアプリケーションインベントリを確認します。
- 保留中の無効化 — Amazon Inspector はインスタンスのスキャンを停止しました。インスタンスは無効になっており、クリーンアップタスクの完了を待っています。
- 保留中の初期スキャン — Amazon Inspector は初期スキャンのためにインスタンスをキューに入れました。
- リソースが終了しました - インスタンスが終了しました。Amazon Inspector は現在、インスタンスの既存の検出結果とカバレッジデータをクリーンアップしています。
- 古いインベントリ — Amazon Inspector は、過去 7 日以内にインスタンスについてキャプチャされた更新されたソフトウェアアプリケーションインベントリを収集できませんでした。

  この問題を修正するには、 を使用して AWS Systems Manager 、必要な Amazon Inspector の関連付けが存在し、インスタンスに対して実行されていることを確認します。さらに、 AWS Systems Manager Fleet Manager を使用してインスタンスのソフトウェアアプリケーションインベントリを確認します。
- アンマネージド型 EC2 インスタンス — Amazon Inspector はインスタンスをモニタリングまたはスキャンしていません。インスタンスは AWS Systems Managerによって管理されていません。

  この問題を修正するには、 AWS Systems Manager Automation AWSSupport-TroubleshootManagedInstance runbookが提供する を使用できます。インスタンスを管理する AWS Systems Manager ように を設定すると、Amazon Inspector は自動的にインスタンスの継続的なモニタリングとスキャンを開始します。
- サポートされていない OS — Amazon Inspector はインスタンスをモニタリングまたはスキャンしていません。インスタンスは、Amazon Inspector がサポートしていないオペレーティングシステムまたはアーキテクチャを使用しています。Amazon Inspector がサポートしているオペレーティ

ングシステムのリストについては、「Amazon EC2 インスタンスのステータス値」を参照してください。

- アクティブにモニタリングして、部分的なエラーが見つかりました — このステータスは、EC2 スキャンはアクティブですが、Linux ベースの Amazon EC2 インスタンス向け Amazon Inspector 詳細検査 に関連するエラーがあることを意味します。詳細検査で発生する可能性のあるエラーは次のとおりです。
  - 詳細検査パッケージコレクションの制限の超過 – インスタンスが Amazon Inspector 詳細検査の制限である 5000 パッケージを超えました。このインスタンスの詳細検査を再開するには、アカウントに関連付けられているカスタムパスの調整を試みます。
  - 詳細検査の 1 日あたりの SSM インベントリ制限の超過 – SSM Agent がインベントリを Amazon Inspector に送信できませんでした。これは、既にこのインスタンスで 1 日あたりインスタンスごとに収集されるインベントリデータの SSM クォータに達しているためです。詳細については、「Amazon EC2 Systems Manager エンドポイントとクォータ」を参照してください。
  - 詳細検査コレクション時間制限の超過 – パッケージのコレクション時間が最大しきい値の 15 分を超えているため、Amazon Inspector はパッケージインベントリの抽出に失敗しました。
  - Deep inspection にはインベントリがありません – Amazon Inspector SSM プラグインは、このインスタンスのパッケージのインベントリをまだ収集していません。これは通常、保留中のスキャンの結果ですが、6 時間経ってもこの状態が続く場合は、Amazon EC2 Systems Manager を使用して、必要な Amazon Inspector 関連付けインスタンスで存在し、実行されていることを確認してください。

EC2 インスタンスのスキャン設定の詳細については、「Amazon EC2 インスタンススキャン」を参照してください。

Amazon ECR リポジトリのカバレッジを評価する

[リポジトリ] タブには、 AWS 環境内の Amazon ECR リポジトリが表示されます。リストは以下のタブでグループにまとめられています。

- すべて — 環境内のすべてのリポジトリを表示します。[ステータス] 列には、レポジトリの現在のスキャンステータスが表示されます。
- アクティブ化 — Amazon Inspector が環境でモニタリングおよびスキャンするように設定されているすべてのリポジトリを表示します。[ステータス] 列には、レポジトリの現在のスキャンステータスが表示されます。

- アクティブ化されていません — Amazon Inspector が環境でモニタリングおよびスキャンしていないすべてのリポジトリを表示します。[理由] 列には、Amazon Inspector がインスタンスをモニタリングおよびスキャンしていない理由が表示されます。

各タブで、アカウント列はリポジトリを所有 AWS アカウント する を指定します。

リポジトリに関するその他の詳細を確認するには、リポジトリの名前を選択します。次に、Amazon Inspector はリポジトリ内のコンテナイメージのリストと各イメージの詳細を表示します。詳細には、イメージタグ、イメージダイジェスト、スキャンステータスが含まれます。また、イメージの緊急の検出結果の数など、主要な検出結果の統計情報も含まれます。検出結果の裏付けとなるデータを掘り下げて確認するには、イメージの [イメージ] タグを選択します。

Amazon ECR リポジトリのステータス値のスキャン

Amazon Elastic Container Registry (Amazon ECR) リポジトリの場合、[ステータス] の値は以下のようになります：

- アクティブ化 (継続) – リポジトリの場合、Amazon Inspector はこのリポジトリ内のイメージを継続的にモニタリングします。リポジトリの拡張スキャン設定は継続的スキャンに設定されています。Amazon Inspector は、プッシュされたときに新しいイメージを最初にスキャンし、そのイメージに関連する新しい CVE が発行されたときにイメージを再スキャンします。Amazon Inspector は、設定した Amazon ECR 再スキャン期間は、このリポジトリ内のイメージを継続的にモニタリングします。
- アクティブ化 (プッシュ時) – Amazon Inspector は新しいイメージがプッシュされると、リポジトリ内の個々のコンテナイメージを自動的にスキャンします。拡張スキャンは、レポジトリに対してアクティブ化され、プッシュ時にスキャンするように設定されています。
- アクセス拒否 — Amazon Inspector は、リポジトリまたはリポジトリ内のコンテナイメージへのアクセスを許可されていません。

  この問題を修正するには、リポジトリの AWS Identity and Access Management (IAM) ポリシーで Amazon Inspector がリポジトリにアクセスできることを確認します。
- 非アクティブ化 (手動) — Amazon Inspector はリポジトリ内のコンテナイメージをモニタリングまたはスキャンしていません。リポジトリの Amazon ECR スキャン設定は、基本的な手動スキャンに設定されています。

  Amazon Inspector を使用してリポジトリ内のイメージのスキャンを開始するには、リポジトリのスキャン設定を拡張スキャンに変更し、イメージを継続的にスキャンするか、新しいイメージがプッシュされたときにのみスキャンするかを選択します。

- アクティブ化 (プッシュ時) – Amazon Inspector は新しいイメージがプッシュされると、リポジトリ内の個々のコンテナイメージを自動的にスキャンします。リポジトリの拡張スキャン設定は、プッシュ時にスキャンするように設定されています。
- 内部エラー – Amazon Inspector がリポジトリをスキャンしようとしたときに内部エラーが発生しました。Amazon Inspector は自動的にエラーに対処し、できるだけ早くスキャンを再開します。

リポジトリのスキャン設定の詳細については、「Amazon ECR コンテナイメージスキャン」を参照してください。

Amazon ECR コンテナイメージのカバレッジを評価する

[イメージ] タブには、環境内の Amazon ECR コンテナイメージが表示されます。 AWS リストは次のタブでグループにまとめられています。

- すべて — 環境内のすべてのコンテナイメージを表示します。[ステータス] 列には、イメージの現在のスキャンステータスが表示されます。
- スキャン — Amazon Inspector が環境でアクティブにモニタリングおよびスキャンしているすべてのコンテナイメージを表示します。[ステータス] 列には、イメージの現在のスキャンステータスが表示されます。
- スキャンしない — Amazon Inspector が環境でモニタリングおよびスキャンしていないすべてのコンテナイメージを表示します。[理由] 列には、Amazon Inspector がイメージをモニタリングおよびスキャンしていない理由が表示されます。

  コンテナイメージが [アクティブ化されていません] タブに表示される理由はいくつかあります。Amazon Inspector のスキャンがアクティブ化されていないリポジトリにイメージが保存されているか、Amazon ECR フィルタリングルールによってそのリポジトリがスキャンされない場合があります。または、[ECR 再スキャン期間] に設定されている日数内にイメージがプッシュまたはプルされていません。詳細については、「Amazon ECR 再スキャン期間の設定」を参照してください。

各タブの [リポジトリ名] 列には、コンテナイメージを格納するリポジトリの名前を指定します。Account 列は、リポジトリを所有 AWS アカウント する を指定します。[最終スキャン日] 列には、Amazon Inspector がそのリソースの脆弱性を最後にチェックした日時が表示されます。これには、メタデータの検出結果が更新されたとき、リソースのアプリケーションインベントリが更新されたとき、または新しい CVE に応じて再スキャンが行われたときのチェックが含まれます。詳細については、「Amazon ECR スキャンのスキャン動作」を参照してください。

コンテナイメージに関するその他の詳細を確認するには、[ECR コンテナイメージ] 列のリンクを選択します。次に、Amazon Inspector はイメージに関する詳細と、そのイメージに関する現在の検出結果を表示します。特定の検出結果の詳細を確認するには、[タイトル] 列のリンクを選択します。その詳細については、「Amazon Inspector 検出結果の詳細の表示」を参照してください。

Amazon ECR コンテナイメージのステータス値のスキャン

Amazon Elastic Container Registry コンテナイメージの場合、[ステータス] の値は以下のようになります：

- アクティブモニタリング (継続的) – Amazon Inspector は継続的にモニタリングしており、関連する新しい CVE が公開されるたびに、イメージと新しいスキャンが実行されます。イメージの Amazon ECR 再スキャン期間は、イメージがプッシュまたはプルされるたびに更新されます。イメージを保存するリポジトリでは拡張スキャンが有効になっており、リポジトリの拡張スキャン設定は継続的スキャンに設定されています。
- アクティブ化 (プッシュ時) – Amazon Inspector は、新しいイメージがプッシュされるたびにイメージを自動的にスキャンします。イメージを保存するリポジトリでは拡張スキャンがアクティブ化になり、リポジトリの拡張スキャン設定はプッシュ時にスキャンするように設定されます。
- 内部エラー – Amazon Inspector がコンテナイメージをスキャンしようとしたときに内部エラーが発生しました。Amazon Inspector は自動的にエラーに対処し、できるだけ早くスキャンを再開します。
- 保留中の初期スキャン – Amazon Inspector は初期スキャンのためにインスタンスをキューに入れました。
- スキャンの適格性の有効期限切れ (継続) – Amazon Inspector はイメージのスキャンを一時停止しました。リポジトリ内のイメージを自動再スキャンするように指定した期間内に、イメージが更新されていません。イメージをプッシュまたはプルしてスキャンを再開できます。
- スキャンの適格性の有効期限切れ (プッシュ時) – Amazon Inspector はイメージのスキャンを中断しました。リポジトリ内のイメージを自動再スキャンするように指定した期間内に、イメージが更新されていません。イメージをプッシュしてスキャンを再開できます。
- スキャン頻度 (手動) — Amazon Inspector は Amazon ECR コンテナイメージをスキャンしません。イメージを保存するリポジトリの Amazon ECR スキャン設定は、基本の手動スキャンに設定されています。Amazon Inspector で自動的にイメージのスキャンを開始するには、リポジトリの設定を拡張スキャンに変更し、イメージを継続的にスキャンするか、新しいイメージがプッシュされたときにのみスキャンするかを選択します。

- サポートされていない OS – Amazon Inspector はインスタンスをモニタリングまたはスキャンしていません。イメージは、Amazon Inspector がサポートしていないオペレーティングシステムに基づいているか、Amazon Inspector がサポートしていないメディアタイプを使用しています。

  Amazon Inspector がサポートしているオペレーティングシステムのリストについては、「サポートされているオペレーティングシステム: Amazon Inspector による Amazon ECR スキャン」を参照してください。Amazon Inspector がサポートするメディアタイプのリストについては、「サポートされているメディアタイプ」を参照してください。

リポジトリとイメージのスキャン設定の詳細については、「Amazon ECR コンテナイメージスキャン」を参照してください。

AWS Lambda 関数のカバレッジの評価

Lambda タブには、 AWS 環境内の Lambda 関数が表示されます。このページには 2 つのテーブルがあります。1 つは Lambda 標準スキャンの関数カバレッジの詳細を示し、もう 1 つは Lambda コードスキャンの関数カバレッジの詳細を示しています。以下のタブに基づいて関数をグループ化できます。

- すべて — 環境内のすべての Lambda 関数を表示します。[ステータス] 列には、Lambda 関数の現在のスキャンステータスが表示されます。
- スキャン — Amazon Inspector がスキャンするように設定されている Lambda 関数を表示します。[ステータス] 列には、各 Lambda 関数の現在のスキャンステータスが表示されます。
- スキャンしない — Amazon Inspector がスキャンするように設定されていない Lambda 関数を表示します。[理由] 列には、Amazon Inspector が関数をモニタリングおよびスキャンしていない理由が表示されます。

  Lambda 関数が [スキャンしない] タブに表示される理由はいくつかあります。Lambda 関数が Amazon Inspector に追加されていないアカウントに属しているか、フィルタリングルールによりこの関数がスキャンされない可能性があります。詳細については、「Lambda 関数のスキャン」を参照してください。

各タブの [関数名] 列には、Lambda 関数の名前を指定します。Account 列は、関数を所有 AWS アカウント する を指定します。[ランタイム] には、関数のランタイムを指定します。[ステータス] 列には、各 Lambda 関数の現在のスキャンステータスが表示されます。[リソース] タグは、関数に適用されたタグを表示します。[最終スキャン日] 列には、Amazon Inspector がそのリソースの脆弱性を最後にチェックした日時が表示されます。これには、メタデータの検出結果が更新されたとき、リソー

スのアプリケーションインベントリが更新されたとき、または新しい CVE に応じて再スキャンが行われたときのチェックが含まれます。詳細については、「Lambda 関数スキャンのスキャン動作」を参照してください。

AWS Lambda 関数のステータス値のスキャン

Lambda 関数の場合、指定できるステータス値は次のとおりです。

- アクティブにモニタリング中 — Amazon Inspector は Lambda 関数を継続的にモニタリングおよびスキャンしています。継続的スキャンには、新しい関数がリポジトリにプッシュされたときの初回スキャンと、関数が更新されたときや新しい共通脆弱性識別子 (CVE) がリリースされたときの関数の自動再スキャンが含まれます。
- タグで除外済み — この関数はタグによるスキャンから除外されているため、Amazon Inspector はスキャンしていません。
- スキャンの適格性が失効しました — Amazon Inspector は、前回呼び出されたり更新されたりしてから 90 日以上が経過しているため、この関数をモニタリングしていません。
- 内部エラー — Amazon Inspector が関数をスキャンしようとしたときに内部エラーが発生しました。Amazon Inspector は自動的にエラーに対処し、できるだけ早くスキャンを再開します。
- 保留中の初期スキャン — Amazon Inspector は初期スキャンの関数をキューに入れました。
- サポートなし — Lambda 関数のランタイムはサポートされていません。

を使用した Amazon Inspector での複数のアカウントの管理 AWS Organizations

Amazon Inspector を使用して、組織内の複数のアカウントを管理できます。これを行うには、 AWS Organizations 管理アカウントで Amazon Inspector をアクティブ化し、委任管理者を指定する必要があります。委任管理者は、組織の Amazon Inspector を管理し、組織に代わってタスクを実行できます。以下のトピックでは、委任管理者アカウントとメンバーアカウントの違い、委任管理者を指定および削除する方法、メンバーアカウントを管理する方法について説明します。

トピック

- Amazon Inspector の委任管理者アカウントとメンバーアカウントについて
- Amazon Inspector 用の委任管理者アカウントの指定

Amazon Inspector の委任管理者アカウントとメンバーアカウントについて

複数アカウント環境で Amazon Inspector を使用する場合、委任管理者アカウントは特定のメタデータにアクセスできます。メタデータには、Amazon EC2、Amazon ECR、Lambda の標準スキャン、および Lambda コードスキャンが含まれます。また、メンバーアカウントのセキュリティ検出結果も含まれます。このセクションでは、委任管理者アカウントが実行できるアクションと、メンバーアカウントが実行できるアクションに関する情報を提供します。

委任管理者のアクション

通常、委任管理者が自分のアカウントに設定を適用すると、その設定は組織内の他のすべてのアカウントに適用されます。委任管理者は、自分のアカウントや関連するメンバーの情報を表示および取得することもできます。Amazon Inspector の委任管理者アカウントは、以下のアクションを実行できます。

- 委任された管理者を指定および削除できるのは AWS Organizations 管理アカウントのみです。
- 委任管理者を指定するときは、管理するメンバーアカウントと同じ組織に所属している必要があります。
- Amazon Inspector のアクティブ化や非アクティブ化など、関連するアカウントの Amazon Inspector のステータスを表示および管理できます。

- 組織内のすべてのメンバーアカウントのスキャンタイプをアクティブ化または非アクティブ化します。
- 組織全体の集約された検出結果データと、組織内のすべてのメンバーアカウントの検出結果の詳細が表示されます。
- 組織内のすべてのアカウントの検出結果に適用される抑制ルールを作成および管理します。
- 組織のすべてのメンバーに対して Amazon ECR 拡張スキャンをアクティブ化します。
- 組織全体のリソースカバレッジを表示します。
- 組織内のすべてのメンバーアカウントの ECR コンテナイメージを自動的に再スキャンする期間を定義します。委任された管理者のスキャン期間設定は、メンバーアカウントが以前に設定した設定よりも優先されます。組織内のすべてのアカウントは、委任管理者の Amazon ECR 自動再スキャン期間を共有します。個別のアカウントに異なる再スキャン期間を設定することはできません。
- 組織内のすべてのアカウントで使用される Amazon Inspector の Amazon EC2 向け詳細検査に 5 つのカスタムパスを指定します。これは、委任された管理者個々のアカウントに設定できる 5 つのカスタムパスに追加されます。詳細検査カスタムパスの設定の詳細については、「Amazon Inspector 詳細検査のカスタムパス」を参照してください。
- メンバーアカウントの Amazon Inspector 詳細検査をアクティブ化または非アクティブ化します。
- 組織内のあらゆるメンバーアカウントの SBOM をエクスポートします。
- 組織内のすべてのメンバーアカウントの Amazon EC2 スキャンモードを設定します。詳細については、「スキャンモードの管理」を参照してください。
- メンバーアカウントによって作成されたスキャン設定を除き、組織内のすべてのアカウントの CIS スキャン設定を作成および管理します。

> **Note**
>
> メンバーアカウントが組織を離れると、委任管理者は、そのアカウントによってスケジュールされたスキャン設定を表示できなくなります。

- 組織内のすべてのアカウントの CIS スキャン結果を表示します。

メンバーアカウントのアクション

メンバーアカウントは Amazon Inspector でアカウントに関する情報を表示および取得できますが、アカウントの設定は委任管理者によって管理されます。組織内のメンバーアカウントは Amazon Inspector で以下のアクションを実行できます。

- メンバー自身のアカウントで Amazon Inspector をアクティベートします。
- メンバー自身のアカウントのリソースカバレッジを表示します。
- メンバー自身のアカウントの検出結果の詳細を表示します。
- メンバー自身のアカウントの ECR コンテナイメージ自動再スキャン期間設定を表示します。
- Amazon Inspector の EC2 向け詳細検査に、個々のアカウントで使用される 5 つのカスタムパスを指定します。これらのパスは、委任管理者が組織用に指定したカスタムパスに加えてスキャンされます。詳細検査パスの設定についての詳細は、「Amazon Inspector 詳細検査のカスタムパス」を参照してください。
- Amazon Inspector 詳細検査向けに委任管理者によって設定されたカスタムパスを表示します。
- アカウントに関連付けられているあらゆるリソースの SBOM をエクスポートします。
- アカウントのスキャンモードを表示します。
- アカウントの CIS スキャン設定を作成および管理します。
- 委任管理者によってスケジュールされたリソースを含む、アカウント内のリソースの CIS スキャン結果を表示します。

> **Note**
>
> アクティベーション後、Amazon Inspector は委任された管理者アカウントでのみ非アクティブ化できます。

Amazon Inspector 用の委任管理者アカウントの指定

委任管理者は、組織のサービスを管理するアカウントです。このトピックでは、Amazon Inspector の委任管理者を指定する方法について説明します。

考慮事項

委任管理者を指定する前に、次の点に注意してください。

委任管理者は、最大 10,000 人のメンバーを管理できます。

> メンバーアカウントが 10,000 を超えると、Amazon CloudWatch Personal Health Dashboard を通じて通知が送信され、委任管理者アカウントに E メールが送信されます。

委任管理者はリージョン別です。

Amazon Inspector はリージョナルサービスです。Amazon Inspector を使用する AWS リージョン予定のすべての の手順の手順を繰り返す必要があります。

組織は、委任された管理者を 1 名だけ持つことができます。

あるアカウントでアカウントを委任管理者として指定する場合 AWS リージョン、そのアカウントは他のすべてのアカウントで委任管理者である必要があります AWS リージョン。

委任された管理者を変更しても、メンバーアカウントの Amazon Inspector は非アクティブ化になりません。

委任管理者を削除しても、メンバーアカウントはスタンドアロンアカウントになり、スキャン設定は影響を受けません。

Organization では、すべての機能がアクティブ化されている AWS 必要があります。

これは のデフォルト設定です AWS Organizations。アクティブ化になっていない場合は、「Activating all features in your organization」を参照してください。

委任された管理者の指定に必要な許可

Amazon Inspector をアクティブ化し、Amazon Inspector に委任された管理者を指定するアクセス許可が必要です。IAM ポリシーの最後に次のステートメントを追加して、これらのアクセス許可を付与します。詳細については、「IAM ポリシーを管理する」を参照してください。

```
{
    "Sid": "PermissionsForInspectorAdmin",
    "Effect": "Allow",
    "Action": [
        "inspector2:EnableDelegatedAdminAccount",
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization"
    ],
    "Resource": "*"
```

```
}
```

AWS 組織の委任管理者の指定

次の手順では、組織の委任管理者を指定する方法について説明します。手順を完了する前に、委任された管理者が管理するメンバーアカウントと同じ組織に属していることを確認してください。

> **Note**
>
> この手順を完了するには、 AWS Organizations 管理アカウントを使用する必要があります。AWS Organizations 管理アカウントのみが委任管理者を指定できます。委任された管理者を指定するには、アクセス許可が必要になる場合があります。詳細については、「委任された管理者の指定に必要な許可」を参照してください。

Amazon Inspector を初めてアクティブ化すると、Amazon Inspector は`AWSServiceRoleForAmazonInspector`アカウントのサービスにリンクされたロールを作成します。Amazon Inspector がサービスにリンクされたロールを使用する方法については、「」を参照してくださいAmazon Inspector でのサービスにリンクされたロールの使用。

Console

Amazon Inspector 用の委任された管理者の指定

1. AWS Organizations 管理アカウントにサインインし、https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. AWS リージョン セレクターを使用して、委任管理者を指定する AWS リージョン を指定します。
3. ナビゲーションペインから、[全般設定] を選択します。
4. 委任された管理者に、委任された管理者として AWS アカウント 指定する の 12 桁の ID を入力します。
5. Delegate を選択し、もう一度 Delegate を選択します。

委任管理者を指定すると、デフォルトでアカウントのすべてのスキャンタイプがアクティブ化されます。 AWS Organizations 管理アカウントの Amazon Inspector をアクティブ化する場合は、次の手順を実行します。

AWS Organizations 管理アカウントの Amazon Inspector をアクティブ化するには

1. 委任管理者アカウントにサインインし、https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. ナビゲーションペインから、[アカウント管理] を選択します。
3. アカウントで管理 AWS Organizations アカウントを選択し、アクティブ化を選択します。
4. AWS Organizations 管理アカウントに対してアクティブ化するスキャンタイプを選択し、送信を選択します。

API

API を使用して委任管理者を指定する

- Organizations 管理アカウントの の認証情報を使用して、EnableDelegatedAdminAccount API AWS アカウント オペレーションを実行します。これを行う AWS Command Line Interface には、次の CLI コマンドを実行します`aws inspector2 enable-delegated-admin-account --delegated-admin-account-id 11111111111`。

> **Note**
> Amazon Inspector の委任管理者にしたいアカウントのアカウント ID を必ず指定してください。

メンバーアカウントの Amazon Inspector スキャンをアクティブ化する

組織の委任管理者である場合は、組織内のメンバーアカウントの Amazon EC2 および Amazon ECR スキャンをアクティブ化できます。メンバーアカウントのスキャンをアクティブ化すると、Amazon Inspector がそのアカウントに対して自動的にアクティブ化され、アカウントは委任管理者アカウントと関連付けられます。Amazon Inspector スキャンタイプの詳細については、「Amazon Inspector の自動スキャンタイプ」を参照してください。このセクションでは、メンバーアカウントのスキャンをアクティブ化する方法について説明します。

メンバーアカウントのスキャンのアクティブ化

メンバーアカウントのスキャンは、さまざまな方法でアクティブ化できます。以下の手順は、委任管理者としてすべてのメンバーアカウントと特定のメンバーアカウントのスキャンをアクティブ化する方法と、メンバーアカウントとしてスキャンをアクティブ化する方法を示しています。

すべてのメンバーアカウントのスキャンを自動的にアクティブ化するには

1. 委任管理者アカウントの認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. リージョンセレクタを使用して、すべてのメンバーアカウントのスキャンをアクティブ化する AWS リージョン を選択します。
3. ナビゲーションペインから、[アカウント管理] を選択します。アカウントタブには、 AWS Organizations 管理アカウントに関連付けられているすべてのメンバーアカウントが表示されます。
4. [組織] で、[アカウント番号] の横にあるボックスを選択します。次に、[アクティブ化] を選択して、メンバーアカウントに適用するスキャンオプションを選択します。以下のスキャンタイプを選択できます。

   - Amazon EC2 スキャン
   - Amazon ECR スキャン
   - Lambda 標準スキャン
   - Lambda コードスキャン

   - 任意のスキャンタイプを選択したら、[保存] を選択します。

   > **Note**
   > アカウントのページが複数ある場合は、各ページでこのステップを繰り返す必要があります。歯車アイコンを選択すると、各ページに表示されるアカウントの数を変更できます。

5. [新しいメンバーアカウントの Inspector を自動的にアクティブ化] 設定をオンにし、組織に追加された新規メンバーアカウントに適用するスキャンオプションを選択します。以下のスキャンタイプを選択できます。

   - Amazon EC2 スキャン
   - Amazon ECR スキャン
   - Lambda 標準スキャン
   - Lambda コードスキャン

- 任意のスキャンタイプを選択したら、[アクティブ化] を選択します。

> **Note**
>
> [新しいメンバーアカウントの Inspector を自動的にアクティブ化] 設定により、組織の今後のメンバー全員に対して Amazon Inspector がアクティブ化されます。
> メンバーアカウント数が 5,000 を超えると、この設定は自動的にオフになります。メンバーアカウント数が 5,000 未満になると、この設定は自動的に再度アクティブ化されます。

6. (推奨) メンバーアカウントのスキャンをアクティブ化 AWS リージョン する各 で、これらの各ステップを繰り返します。

特定メンバーアカウントのスキャンをアクティブ化するには

1. 委任管理者アカウントの認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. リージョンセレクタを使用して、すべてのメンバーアカウントのスキャンをアクティブ化する AWS リージョン を選択します。
3. ナビゲーションペインから、[アカウント管理] を選択します。アカウントタブには、 AWS Organizations 管理アカウントに関連付けられているすべてのメンバーアカウントが表示されます。
4. [組織] で、スキャンをアクティブ化する各メンバーアカウント番号の横にあるボックスを選択します。次に、[アクティブ化] を選択して、メンバーアカウントに適用するスキャンオプションを選択します。以下のスキャンタイプを選択できます。

   - Amazon EC2 スキャン
   - Amazon ECR スキャン
   - Lambda 標準スキャン
   - Lambda コードスキャン

   - 任意のスキャンタイプを選択したら、[保存] を選択します。

> **Note**
>
> アカウントのページが複数ある場合は、各ページでこのステップを繰り返す必要があります。歯車アイコンを選択すると、各ページに表示されるアカウントの数を変更できます。

5. (推奨) 特定のメンバーのスキャンをアクティブ化 AWS リージョン する各 で、これらの各ステップを繰り返します。

メンバーアカウントとしてスキャンをアクティブ化するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. リージョンセレクタを使用して、すべてのメンバーアカウントのスキャンをアクティブ化する AWS リージョン を選択します。
3. ナビゲーションペインから、[アカウント管理] を選択します。アカウントタブには、 AWS Organizations 管理アカウントに関連付けられているすべてのメンバーアカウントが表示されます。
4. [組織] で、アカウント番号の横にあるボックスを選択します。次に、[アクティブ化] を選択して、適用するスキャンオプションを選択します。以下のスキャンタイプを選択できます。

   - Amazon EC2 スキャン
   - Amazon ECR スキャン
   - Lambda 標準スキャン
   - Lambda コードスキャン

   - 任意のスキャンタイプを選択したら、[保存] を選択します。
5. (推奨) メンバーアカウントのスキャンをアクティブ化するリージョンごとに、これらの手順を繰り返します。

> **Note**
>
> AWS Organizations 管理アカウントに Amazon Inspector の委任管理者アカウントがある場合は、アカウントをメンバーアカウントとしてアクティブ化してスキャンの詳細を表示できます。

Amazon Inspector でのメンバーアカウントの関連付けを解除する

委任管理者として、アカウントからメンバーアカウントの関連付けを解除しなければならない場合があります。メンバーアカウントの関連付けを解除しても、Amazon Inspector はアカウントで引き続きアクティブ化され、アカウントはスタンドアロンアカウントになります。また、アカウントの Amazon Inspector を管理するアクセス許可もなくなります。ただし、以前に関連付けを解除したメンバーアカウントは、いつでもアカウントに関連付けることができます。このセクションでは、委任管理者としてメンバーアカウントの関連付けを解除する方法について説明します。

Console

コンソールを使用して、メンバーアカウントの関連付けを解除するには

1. 委任管理者アカウントの認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. リージョンセレクタを使用して、メンバーアカウントの関連付けを解除する AWS リージョン を選択します。
3. ナビゲーションペインから、[アカウント管理] を選択します。
4. [組織] で、関連付けを解除する各アカウント番号の横にあるボックスをオンにします。
5. [アクション] メニューを選択してから、[アカウントの関連付けを解除] を選択します。

API

API を使用してメンバーアカウントの関連付けを解除するには

DisassociateMember API オペレーションを実行します。リクエストで、関連付けを解除するアカウント ID を指定します。

Amazon Inspector での委任管理者の削除

Amazon Inspector 委任管理者アカウントを削除しなければならない場合があります。これは、 AWS Organizations 管理アカウントから実行できます。Amazon Inspector 委任管理者アカウントを削除すると、Amazon Inspector はアカウントとそのすべてのメンバーアカウントで引き続きアクティブ化されます。委任管理者アカウントとそのすべてのメンバーアカウントはスタンドアロンアカウントになり、元のスキャン設定を保持します。このセクションでは、委任管理者アカウントを削除する方法について説明します。

Amazon Inspector 委任管理者の削除

次の手順は、Amazon Inspector の委任管理者を削除する方法と、委任管理者アカウントからメンバーアカウントを関連付ける方法を示しています。

Amazon Inspector 委任管理者を割り当てる方法については、「Amazon Inspector 用の委任された管理者アカウントの指定」を参照してください。

> **Note**
>
> Amazon Inspector 委任管理者を割り当てた後、Amazon Inspector 委任管理者はメンバーアカウントを手動で関連付ける必要があります。

委任された管理者を削除するには

1. AWS Organizations 管理アカウント AWS Management Console を使用して にサインインします。
2. https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
3. リージョンセレクタを使用して、委任管理者 AWS リージョン を削除する を選択します。
4. ナビゲーションペインから、[全般設定] を選択します。
5. [委任された管理者] セクションで [削除] を選択し、アクションを確認します。

メンバーを新しい委任された管理者と関連付けるには

1. 委任管理者アカウントの認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。

2. リージョンセレクタを使用して、メンバーを関連付ける AWS リージョン を選択します。
3. ナビゲーションペインから、[アカウント管理] を選択します。
4. [組織] で、[アカウント番号] の横にあるボックスを選択します。
5. [アクション]、[メンバーを追加] の順に選択します。

Amazon Inspector リソースのタグ付け

タグは、 AWS リソースに追加するラベルです。タグは、特定の基準に基づいて AWS リソースを分類するのに役立ちます。タグはキーと値のペアで構成されます。タグキーは一般的なラベルです。タグ値は、タグキーの説明です。Amazon Inspector を使用すると、抑制ルールと CIS スキャン設定にタグを付けることができます。Amazon Inspector リソースごとに最大 50 個のタグを追加できます。

タグ付けの基本

タグは、キーと値のペアから構成されます。タグキーは一般的なラベルです。タグ値は、タグキーの説明です。このトピックでは、Amazon Inspector リソースのタグ付けの基本について説明します。Amazon Inspector リソースにタグを付けるときは、次の点を考慮してください。

- 抑制ルールと CIS スキャン設定にタグを付けることができます。
- Amazon Inspector リソースごとに最大 50 個のタグを追加できます。
- タグ キーは一意である必要があります。
- タグキーには 1 つのタグ値しか使用できません。
- タグキーとタグ値は、最大 128 UTF-8 文字です。文字には、文字、数字、スペース、または記号 `_ . : / = + -` を使用できます`@`。
- `aws` プレフィックスは、どのタグでも使用したり、このプレフィックスでタグを変更したりすることはできません。`aws` プレフィックス が付いたタグは、 が使用するために予約されています AWS。
- Amazon Inspector リソースに割り当てられたタグは、 AWS アカウントとそれらを作成した AWS リージョン でのみ使用できます。
- リソースを削除すると、リソースに関連付けられているすべてのタグも削除されます。

タグの詳細については、「 AWS リソースのタグ付けとタグエディタユーザーガイド」の「ベストプラクティスと戦略」を参照してください。

> **Note**
>
> タグは、機密情報または機密情報を保存することを目的としたものではありません。このタイプのデータを保存するためにタグを使用しないでください。タグは、他の AWS サービスからアクセスできます。

タグの追加

Amazon Inspector リソースにタグを追加できます。これらのリソースには、抑制ルールと CIS スキャン設定が含まれます。タグは、特定の基準に基づいて AWS リソースを分類するのに役立ちます。このトピックでは、Amazon Inspector リソースにタグを追加する方法について説明します。

Amazon Inspector リソースへのタグの追加

抑制ルールと CIS スキャン設定にタグを付けることができます。次の手順では、コンソールと Amazon Inspector API でタグを追加する方法について説明します。

コンソールでのタグの追加

コンソールで Amazon Inspector リソースにタグを追加できます。

抑制ルールへのタグの追加

作成時に抑制ルールにタグを追加できます。詳細については、「抑制ルールの作成」を参照してください。

抑制ルールを編集してタグを含めることもできます。詳細については、「抑制ルールの編集」を参照してください。

CIS スキャン設定へのタグの追加

作成時に CIS スキャン設定にタグを追加できます。詳細については、「CIS スキャン設定の作成」を参照してください。

CIS スキャン設定を編集してタグを含めることもできます。詳細については、「CIS スキャン設定の編集」を参照してください。

Amazon Inspector API を使用したタグの追加

Amazon Inspector API を使用して Amazon Inspector リソースにタグを追加できます。

Amazon Inspector リソースへのタグの追加

`TagResource` API を使用して Amazon Inspector リソースにタグを追加します。コマンドには、リソースの ARN とタグのキーと値のペアを含める必要があります。次のコマンド例では、サプレッションフィルターに空のリソース ARN を使用します。キーは `CostAllocation`で、値は ですdev。Amazon Inspector のリソースタイプの詳細については、「サービス認可リファレンス」の「Amazon Inspector2 のアクション、リソース、および条件キー」を参照してください。

```
aws inspector2 tag-resource \
--resource-arn "arn:${Partition}:inspector2:${Region}:${Account}:owner/${OwnerId}/
filter/${FilterId}" \
--tags CostAllocation=dev \
--region us-west-2
```

作成時の抑制ルールへのタグの追加

CreateFilter API を使用して、作成時に抑制ルールにタグを追加します。

```
aws inspector2 create-filter \
--name "ExampleSuppressionRuleECR" \
--action SUPPRESS \
--filter-criteria 'resourceType=[{comparison="EQUALS", value="AWS_ECR_IMAGE"}]' \
--tags Owner=ApplicationSecurity \
--region us-west-2
```

CIS スキャン設定へのタグの追加

CreateCisScanConfiguration API を使用して、CIS スキャン設定にタグを追加します。

```
aws inspector2 create-cis-scan-configuration \
--scan-name "CreateConfigWithTagsSample" \
--security-level LEVEL_2 \
--targets accountIds=SELF,targetResourceTags={InspectorCisScan=True} \
--schedule 'daily={startTime={timeOfDay=11:10,timezone=UTC}}' \
--tags Owner=SecurityEngineering \
--region us-west-2
```

タグの削除

Amazon Inspector リソースからタグを削除できます。これらのリソースには、抑制ルールと CIS スキャン設定が含まれます。タグは、特定の基準に基づいて AWS リソースを分類するのに役立ちます。このトピックでは、Amazon Inspector リソースからタグを削除する方法について説明します。

Amazon Inspector リソースからのタグの削除

抑制ルールと CIS スキャン設定からタグを削除できます。次の手順では、コンソールと Amazon Inspector API でタグを削除する方法について説明します。

コンソールでのタグの削除

コンソールで Amazon Inspector リソースからタグを削除できます。

抑制ルールからのタグの削除

抑制ルールを編集してタグを含めないようにすることで、抑制ルールからタグを削除できます。詳細については、「抑制ルールの編集」を参照してください。

CIS スキャン設定からのタグの削除

CIS スキャン設定を編集してタグを含めないようにすることで、CIS スキャン設定からタグを削除できます。詳細については、「CIS スキャン設定の編集」を参照してください。

Amazon Inspector API を使用したタグの削除

Amazon Inspector API を使用してAmazon Inspector リソースからタグを削除できます。

Amazon Inspector リソースからのタグの削除

`UntagResource` API を使用して、Amazon Inspector リソースからタグを削除します。

次のスニペットは、 を使用して Amazon Inspector リソースからタグを削除する方法の例を示しています`UntagResource`。コマンドには、リソースの ARN とタグのキーを含める必要があります。次の例では、サプレッションフィルターに空のリソース ARN を使用します。キーは、`CostAllocation` です。Amazon Inspector のリソースタイプの詳細については、「サービス認可リファレンス」の「Amazon Inspector2 のアクション、リソース、および条件キー」を参照してください。

```
aws inspector2 untag-resource \
--resource-arn "arn:${Partition}:inspector2:${Region}:${Account}:owner/${OwnerId}/cis-configuration/${CISScanConfigurationId}" \
--tag-keys CostAllocation \
--region us-west-2
```

Amazon Inspector での使用状況とコストのモニタリング

Amazon Inspector コンソールと API を使用して、ご使用の環境での Amazon Inspector の月額料金を予測できます。マルチアカウント環境の Amazon Inspector 管理者であれば、環境の総コストと、すべてのメンバーアカウントのコストメトリクスを表示できます。このセクションでは、使用状況統計にアクセスし、使用コストを計算する方法について説明します。

コンソール使用状況を使用する

Amazon Inspector の使用状況と予測コストをコンソールから評価できます。

使用統計にアクセスするには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ページの右上隅にある AWS リージョン セレクターを使用して、コストをモニタリングするリージョンを選択します。
3. ナビゲーションペインで 使用状況を選択します。

[アカウント別] タブには、[アカウント使用量] に表示されている 30 日間の期間に基づく予測総コストが表示されます。[予測コスト] 列の表で値を選択すると、そのアカウントのスキャンタイプ別の使用量の内訳が表示されます。この詳細ウィンドウでは、そのアカウントで無料トライアルがアクティブになっているスキャンタイプも確認できます。

あなたが組織の委任された管理者である場合は、組織内の各アカウントの表に 1 行が表示されます。組織内のアカウントの関連付けが解除されると、コンソールにはそのアカウントの予測コストが - として表示されます。

[スキャンタイプ別] タブには、現在の 30 日間の実際の使用量の内訳がスキャンのタイプごとに表示されます。この情報は、[アカウント別] タブの予測コストの計算に使用されます。

組織の委任された管理者の場合は、組織内の各アカウントの使用状況を確認できます。

このタブでは、以下のいずれかのペインを展開して使用状況統計を表示できます。

Amazon EC2 スキャン

Amazon Inspector 使用状況コンソールは、エージェントベースのスキャンとエージェントレススキャンの以下のメトリクスを追跡します。

- インスタンス (Avg) — Amazon Inspector は、カバレッジ時間を使用して EC2 インスタンススキャンの平均リソース数を計算します。平均は、合計カバレッジ時間を 720 時間 (30 日間の時間数) で割ったものです。
- カバレッジ時間 — Amazon EC2 スキャンの場合、Amazon Inspector がアカウント内の各 EC2 インスタンスに対してアクティブカバレッジを提供した過去 30 日間の合計時間数です。EC2 インスタンスの場合、カバレッジ時間とは、Amazon Inspector がインスタンスを発見してから、インスタンスが終了または停止されるまで、またはタグによってスキャンから除外されるまでの時間です (停止したインスタンスを再起動するか、除外タグを削除すると、Amazon Inspector カバレッジを再開し、そのインスタンスのカバレッジ時間は引き続き加算されます) 。

CIS インスタンススキャン – アカウント内のインスタンスに対して実行された CIS スキャンの合計数。

Amazon ECR スキャン

初回スキャン — 過去 30 日間にアカウント内のイメージを初めてスキャンした合計回数。

再スキャン — 過去 30 日以内にアカウント内のイメージを再スキャンした合計回数。再スキャンとは、Amazon Inspector が以前にスキャンした ECR イメージに対して実行されるスキャンです。ECR リポジトリを継続的スキャン用に設定している場合、Amazon Inspector がデータベースに新しい共通脆弱性識別子 (CVE) を追加すると、再スキャンが自動的に行われます。

Lambda スキャン

Amazon Inspector 使用状況コンソールは、Lambda 標準スキャンと Lambda コードスキャンについて以下のメトリクスを追跡します。

- Lambda 関数の数 (Avg) – Amazon Inspector は、カバレッジ時間を使用して Lambda 関数スキャンの関数の平均数を計算します。平均は、カバレッジ時間の合計を 720 時間 (30 日間の時間数) で割ったものです。
- カバレッジ時間 — Lambda 関数スキャンの場合、Amazon Inspector がアカウント内の各 Lambda 関数に対してアクティブカバレッジを提供した過去 30 日間の合計時間数です。 AWS Lambda 関数については、Amazon Inspector が関数を検出した時点から、その関数が削除されるか、スキャンから除外されるまでの期間で、対応時間が計算されます。除外された関数が再び含まれた場合でも、その関数のカバレッジ時間は引き続き加算されます。

Amazon Inspector での使用コストの計算方法について

Amazon Inspector が提供するコストは実際のコストではなく見積りであるため、 AWS Billing コンソールのコストとは異なる場合があります。

Amazon Inspector が「料金表」ページでコストを計算する方法について、次の点に注意してください。

- 使用コストには現在のリージョンのみが反映されます。スキャンタイプごとの料金は AWS リージョンによって異なります。リージョンごとの正確な料金を確認するには、Amazon Inspector の料金表」を参照してください。
- 予測される使用量はすべて、最も近い米ドルに四捨五入されています。
- 割引は、予測コストには含まれません。
- 予測コストは、スキャンタイプごとの 30 日間の使用期間の合計料金です。アカウントの使用日数が 30 日未満の場合、Amazon Inspector は、現在対象となっているリソースが 30 日間の残りの期間も引き続き適用されるものとして、30 日後のコストを予測します。
- スキャンタイプごとのコストは、以下に基づいて計算されます。
  - EC2 スキャン: コストは、過去 30 日間に Amazon Inspector の対象となった EC2 インスタンスの平均数を反映しています。
  - ECR コンテナスキャン: コストは、過去 30 日間の初回イメージスキャンとイメージ再スキャンの合計が反映されます。
  - Lambda 標準スキャン: コストは、過去 30 日間に Amazon Inspector の対象となった Lambda 関数の平均数を反映しています。
  - Lambda コードスキャン: コストは、過去 30 日間に Amazon Inspector の対象となった Lambda 関数の平均数を反映しています。

Amazon Inspector の無料トライアルについて

Amazon Inspector では、各スキャンタイプに無料トライアルがあります。スキャンタイプをアクティブ化すると、そのスキャンタイプの 15 日間の無料トライアルに自動的に登録されます。無料トライアルが開始されると、スキャンタイプを非アクティブ化した場合でも、15 日後に自動的に期限切れになります。

> **Note**
>
> 無料トライアルは、CIS スキャンには適用されません。

Amazon Inspector 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からメリットを得られます。

セキュリティは、 AWS 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説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任があります AWS クラウド。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。Amazon Inspector に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon Inspector の使用時に責任共有モデルがどのように適用されるかを理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンス上の目的を達成するように Amazon Inspector を設定する方法について説明します。また、Amazon Inspector リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon Inspector におけるデータ保護
- Amazon Inspector のための Identity and Access Management
- Amazon Inspector のモニタリング
- Amazon Inspector のコンプライアンス検証
- Amazon Inspector の耐障害性
- Amazon Inspector のインフラストラクチャセキュリティ
- Amazon Inspector でのインシデントへの対応
- Amazon Inspector インターフェイスエンドポイント (AWS PrivateLink

Amazon Inspector におけるデータ保護

責任 AWS 共有モデル、Amazon Inspector でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。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と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。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に投稿された AWS 責任共有モデルおよび 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、認証情報を保護し AWS アカウント 、 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 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。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。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- 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 AWS を介して 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、または SDK を使用して Amazon Inspector AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 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。

トピック

- 保管中の暗号化
- 転送中の暗号化

保管中の暗号化

デフォルトでは、Amazon Inspector は AWS 暗号化ソリューションを使用して保管中のデータを保存します。Amazon Inspector は、次のようなデータを暗号化します。

- で収集されたリソースインベントリ AWS Systems Manager。
- Amazon Elastic Container Registry イメージから解析されたリソースインベントリ
- から AWS 所有の暗号化キーを使用して生成されたセキュリティ検出結果 AWS Key Management Service

AWS 所有キーを管理、使用、または表示することはできません。ただし、データを暗号化するキーを保護するために何らかの操作を行ったり、プログラムを変更したりする必要はありません。詳細については、「AWS 所有キー」を参照してください。

Amazon Inspector を無効にすると、収集したインベントリやセキュリティ検出結果など、Amazon Inspector が保存または管理するすべてのリソースが永久に削除されます。

検出結果のコードの保管時の暗号化

Amazon Inspector Lambda コードスキャンの場合、Amazon Inspector は CodeGuru と提携してコードの脆弱性をスキャンします。脆弱性が検出されると、CodeGuru は脆弱性を含むコードのスニペットを抽出し、Amazon Inspector がアクセスをリクエストするまでそのコードを保存します。デフォルトでは、CodeGuru は AWS 所有キーを使用して抽出されたコードを暗号化しますが、暗号化に独自のカスタマーマネージド AWS KMS キーを使用するように Amazon Inspector を設定できます。

次のワークフローでは、Amazon Inspector が、設定したキーを使用してコードを暗号化する方法を説明しています。

1. Amazon Inspector UpdateEncryptionKey API を使用して Amazon Inspector に AWS KMS キーを指定します。
2. Amazon Inspector は、 AWS KMS キーに関する情報を CodeGuru に転送します。CodeGuru は将来の使用のために情報を保存します。

3. CodeGuru は、Amazon Inspector で設定したキー AWS KMS の許可を にリクエストします。
4. CodeGuru は、キーから暗号化されたデータ AWS KMS キーを作成し、保存します。このデータキーは、CodeGuru によって保存されたコードデータを暗号化するために使用されます。
5. Amazon Inspector がコードスキャンからデータをリクエストするたびに、CodeGuru はグラントを使用して暗号化されたデータキーを復号化し、次にそのキーを使用してデータを復号化して取得できるようにします。

Lambda コードスキャンを無効にすると、CodeGuru はグラントを廃止し、関連付けられたデータキーを削除します。

カスタマーマネージドキーによるコード暗号化のアクセス許可

暗号化を使用するには、 AWS KMS アクションへのアクセスを許可するポリシーと、Amazon Inspector と CodeGuru にこれらのアクションを条件キーで使用するためのアクセス許可を付与するステートメントが必要です。

アカウントの暗号化キーを設定、更新、またはリセットする場合は、AWS マネージドポリシー: AmazonInspector2FullAccess などの Amazon Inspector 管理者ポリシーを使用する必要があります。また、暗号化対象として選択したキーに関する検出結果またはデータからコードスニペットを取得する必要がある読み取り専用ユーザーには、次のアクセス許可を付与する必要があります。

KMS の場合、ポリシーは以下のアクションを実行できるようにする必要があります。

- `kms:CreateGrant`
- `kms:Decrypt`
- `kms:DescribeKey`
- `kms:GenerateDataKeyWithoutPlainText`
- `kms:Encrypt`
- `kms:RetireGrant`

ポリシーで正しい AWS KMS アクセス許可があることを確認したら、Amazon Inspector と CodeGuru が暗号化にキーを使用できるようにするステートメントをアタッチする必要があります。以下のポリシーステートメントをアタッチします。

> **Note**
>
> Region を、Amazon Inspector Lambda コードスキャンが有効になっている AWS リージョンに置き換えます。

```
{
            "Sid": "allow CodeGuru Security to request a grant for a AWS KMS key",
          "Effect": "Allow",
          "Action": "kms:CreateGrant",
          "Resource": "*",
          "Condition": {
           "ForAllValues:StringEquals": {
            "kms:GrantOperations": [
             "GenerateDataKey",
             "GenerateDataKeyWithoutPlaintext",
             "Encrypt",
             "Decrypt",
             "RetireGrant",
             "DescribeKey"
            ]
           },
           "StringEquals": {
            "kms:ViaService": [
             "codeguru-security.Region.amazonaws.com"
            ]
           }
          }
         },
         {
          "Sid": "allow Amazon Inspector and CodeGuru Security to use your AWS KMS key",
          "Effect": "Allow",
          "Action": [
           "kms:Encrypt",
           "kms:Decrypt",
           "kms:RetireGrant",
           "kms:DescribeKey",
           "kms:GenerateDataKeyWithoutPlaintext"
          ],
          "Resource": "*",
          "Condition": {
```

```
          "StringEquals": {
           "kms:ViaService": [
            "inspector2.Region.amazonaws.com",
            "codeguru-security.Region.amazonaws.com"
           ]
          }
         }
        }
```

> **Note**
>
> ステートメントをポリシーに追加するときに、構文が有効であることを確認します。ポリシーは JSON 形式を使用します。これは、ステートメントをポリシーに追加する場所に応じて、ステートメントの前後にカンマを追加する必要があることを意味します。ステートメントを最後のステートメントとして追加する場合は、前のステートメントの右中括弧の後にカンマを追加します。最初のステートメントとして追加するか、既存の 2 つのステートメントの間に追加する場合は、右中括弧の後にカンマを追加します。

カスタマーマネージドキーによる暗号化の設定

カスタマーマネージドキーを使用してアカウントの暗号化を設定するには、カスタマーマネージドキーによるコード暗号化のアクセス許可 で説明されているアクセス許可を持つ Amazon Inspector 管理者である必要があります。さらに、検出結果と同じ AWS リージョンに AWS KMS キー、またはマルチリージョンキーが必要になります。アカウントで既存の対称キーを使用するか、 AWS マネジメントコンソールまたは AWS KMS APIs を使用して対称カスタマーマネージドキーを作成できます。詳細については、「 AWS KMS ユーザーガイド」の「対称暗号化 AWS KMS キーの作成」を参照してください。

> **Note**
>
> 2025 年 6 月 13 日以降、コードスニペットの暗号化/復号中に CloudTrail に記録された AWS KMS リクエストのサービスプリンシパルは「codeguru-reviewer」から「q」に変更されます。

Amazon Inspector APL を使用して暗号化を設定する

暗号化用のキーを設定するには、Amazon Inspector 管理者としてサインインしているときに Amazon Inspector API の UpdateEncryptionKey オペレーションを実行します。API リクエストで、`kmsKeyId`フィールドを使用して、使用する AWS KMS キーの ARN を指定します。`scanType` に `CODE` を、`resourceType` に `AWS_LAMBDA_FUNCTION` を入力します。

UpdateEncryptionKey API を使用して、Amazon Inspector が暗号化に使用している AWS KMS キーを確認できます。

> **Note**
>
> カスタマーマネージドキーを設定していない`GetEncryptionKey`ときに を使用しようとすると、オペレーションは`ResourceNotFoundException`エラーを返します。これは、AWS 所有キーが暗号化に使用されていることを意味します。

キーを削除したり、Amazon Inspector や CodeGuru へのアクセスを拒否するようにポリシーを変更したりすると、コードの脆弱性の検出結果にアクセスできなくなり、アカウントの Lambda コードスキャンが失敗します。

`ResetEncryptionKey` を使用して、 AWS 所有キーの使用を再開し、Amazon Inspector の検出結果の一部として抽出されたコードを暗号化できます。

転送中の暗号化

AWS は、 AWS 内部システムと他の AWS サービスの間で転送されるすべてのデータを暗号化します。 AWS Systems Manager は、評価のために Transport Layer Security (TLS) で保護されたチャネル AWS を介して に送信する顧客所有の EC2 インスタンスからテレメトリデータを収集します。Security Hub に送信される Amazon ECR および AWS Lambda 関数スキャンの検出結果は、TLS で保護されたチャネルを使用して暗号化されます。詳細については、「Systems Manager でのデータ保護」を参照して、SSM が転送中のデータを暗号化する方法を理解してください。

Amazon Inspector のための Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソース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 Amazon

Inspector リソースの使用を承認 (許可を付与) するかを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon Inspector と IAM の連携
- Amazon Inspector アイデンティティベースのポリシーの例
- AWS Amazon Inspector の マネージドポリシー
- Amazon Inspector でのサービスにリンクされたロールの使用
- Amazon Inspector アイデンティティとアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon Inspector で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon Inspector サービスを使用する場合は、管理者から必要な認証情報と許可が与えられます。さらに多くの Amazon Inspector 機能を使用して作業を行う場合は、追加の許可が必要になることがあります。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。Amazon Inspector の機能 にアクセスできない場合は、「Amazon Inspector アイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon Inspector リソースを担当している場合は、Amazon Inspector に対する完全なアクセス権があると思われます。サービスのユーザーがどの Amazon Inspector の機能や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権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で Amazon Inspector と IAM を併用する方法の詳細については、「Amazon Inspector と IAM の連携」を参照してください。

IAM 管理者 – IAM 管理者は、Amazon Inspector 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。IAM で使用できる Amazon Inspector 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、「Amazon Inspector アイデンティティベースのポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして 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によって、認証 (にサインイン AWS) される必要があります。

ID ソースを介し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 Identity Center （ IAM Identity Center) 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、フェデレーション 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場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。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 Management Console 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「 へのサインイン AWS アカウント方法」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、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエスト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 Identity Center 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「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。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は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。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、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティ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アイデンティティがアクセスすると 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 Identity Center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、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

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 ID 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保有する IAM 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、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、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、IAM ユーザーの集団を指定するアイデンティティです。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容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

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 ユーザーに関するユースケース」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 IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次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 – フェデレーティッド 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。フェデレーティッド 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ロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」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。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center 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 User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、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における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 でのクロスアカウントのリソースへのアクセス」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。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、または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。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して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。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、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービ

ス へのリクエストをリクエストする と組み合わせて使用します。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。

- サービスロール - 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、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」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。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、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、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、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、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」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。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。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 Management Console、、 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

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、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。

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ています。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「管理ポリシーとインラインポリシーのいずれかを選択する」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、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制御できます。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は、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 を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

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、または ロール) がリソースにアクセスするための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。

Amazon S3、および Amazon VPC は AWS WAF、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ポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限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。結果として得られる権限は、エンティティのアイデンティティベース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 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations 「 ユーザーガイド AWS のサービス 」の「リソースコントロールポリシー (RCPs」を参照してください。 RCPs

- 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ションユーザーの一時的な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。結果としてセッションの権限は、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になります。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権限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。

Amazon Inspector と IAM の連携

IAM を使用して Amazon Inspector へのアクセスを管理する前に、Amazon Inspectorで使用できる IAM 機能について理解しておく必要があります。

Amazon Inspector で使用できる IAM の機能

IAM 機能	Amazon Inspector のサポート
アイデンティティベースポリシー	はい
リソースベースのポリシー	いいえ
ポリシーアクション	はい
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	いいえ
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
プリンシパル権限	はい

IAM 機能	Amazon Inspector のサポート
サービスロール	いいえ
サービスリンクロール	はい

Amazon Inspector およびその他の がほとんどの IAM 機能と AWS のサービス 連携する方法の概要については、IAM ユーザーガイドのAWS のサービス 「IAM と連携する 」を参照してください。

Amazon Inspector アイデンティティベースのポリシー

アイデンティティ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 ユーザーグループ、ユーザーのグループ、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。ID ベースの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、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付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 ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon Inspector アイデンティティベースのポリシーの例

Amazon Inspector アイデンティティベースのポリシーの例は、「Amazon Inspector アイデンティティベースのポリシーの例」を参照してください。

Amazon Inspector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制御できます。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は、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 を含めることができます AWS のサービス。

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場合 AWS アカウント、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、アイデンティティ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権限を付与します。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。

Amazon Inspector のポリシーアクション

ポリシーアクションのサポート:あり

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。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。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。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

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。

Amazon Inspector アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon Inspector で定義されるアクション」を参照してください。

Amazon Inspector 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。

```
inspector2
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "inspector2:action1",
      "inspector2:action2"
         ]
```

Amazon Inspector アイデンティティベースのポリシーの例は、「Amazon Inspector アイデンティティベースのポリシーの例」を参照してください。

Amazon Inspector のポリシーリソース

ポリシーリソースのサポート: あり

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。

`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、リソースレベルの権限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示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon Inspectorリソースのタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon Inspector で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon Inspector で定義されるアクション」を参照してください。

Amazon Inspector アイデンティティベースのポリシーの例は、「Amazon Inspector アイデンティティベースのポリシーの例」を参照してください。

Amazon Inspector 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 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。

`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な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。ステートメントの権限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。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーを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。

Amazon Inspector の条件キーの一覧については、「サービス認可リファレンス」の「Amazon Inspector の条件キー」を参照してください。どのアクションやリソースで条件キーを使用できるかについては、「Amazon Inspector で定義されるアクション」を参照してください。

Amazon Inspector アイデンティティベースのポリシーの例は、「Amazon Inspector アイデンティティベースのポリシーの例」でご確認ください。

Amazon Inspector の ACL

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。

Amazon Inspector と ABAC

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon Inspector 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

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する場合などの詳細については、IAM ユーザーガイドAWS のサービス の「IAM と連携する 」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 AWS Management Console を使用して にサインインする場合、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。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。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して アクセスできます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon Inspector のクロスサービスプリンシパル許可

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 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。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、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。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。

Amazon Inspector 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 のサービスに許可を委任するロールを作成する」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon Inspectorの機能が破損する可能性があります。Amazon Inspector が指示する場合以外は、サービスロールを編集しないでください。

Amazon Inspector で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、 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、サービスリンクロールの権限を表示できますが、編集することはできません。

サービスリンクロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の中から、[Service-linked role (サービスリンクロール)] 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon Inspector 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、ユーザーおよびロールにはAmazon Inspectorリソースを作成または変更する許可はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、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、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらサンプルの 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「IAM ポリシーを作成する (コンソール)」を参照してください。

Amazon Inspector が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「Amazon Inspector のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- Amazon Inspector コンソールの使用
- 自分の権限の表示をユーザーに許可する
- すべての Amazon Inspector リソースへの読み取り専用アクセスを許可する
- すべての Amazon Inspector リソースへのフルアクセスを許可する

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、ユーザーのアカウント内で誰かが Amazon Inspectorリソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。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 マネージドポリシー」または「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な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「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで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、アクションやリソースへのアクセスを制限できます。例えば、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 AWS 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、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で 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 AWS アカウント、MFA をオンにしてセキュリティを強化します。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 を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon Inspector コンソールの使用

Amazon Inspector コンソールにアクセスするには、許可の最小限のセットが必要です。アクセス許可により、 AWS アカウントの Amazon Inspector リソースの詳細をリストおよび表示できます。最小限必要な許可よりも制限が厳しい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

シーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

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、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き Amazon Inspector コンソールを使用できるようにするには、エンティティに Amazon Inspector *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
```

```
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

すべての Amazon Inspector リソースへの読み取り専用アクセスを許可する

この例では、すべての Amazon Inspector リソースへの読み取り専用アクセスを許可するポリシーを示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "inspector2:Describe*",
                "inspector2:Get*",
                "inspector2:BatchGet*",
                "inspector2:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

すべての Amazon Inspector リソースへのフルアクセスを許可する

この例では、すべての Amazon Inspector リソースへのフルアクセスを許可するポリシーを示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "inspector2:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "inspector2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:RegisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Amazon Inspector の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、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るように、多くの一般的なユースケースにアクセス許可を提供するように設計されています。

AWS 管理ポリシーは、すべての AWS 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。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、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 が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS マネージドポリシーを更新する可能性が高くなります。

詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS 管理ポリシー: AmazonInspector2FullAccess_v2

`AmazonInspector2FullAccess` ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、Amazon Inspector へのフルアクセスと、他の関連サービスへのアクセスを許可します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `inspector2` – Amazon Inspector APIs への完全なアクセスを許可します。
- `codeguru-security` – 管理者がアカウントのセキュリティ検出結果と設定を取得できるようにします。
- `iam` – Amazon Inspector がサービスにリンクされたロール `AWSServiceRoleForAmazonInspector2`と を作成できるようにします`AWSServiceRoleForAmazonInspector2Agentless`。 `AWSServiceRoleForAmazonInspector2`はAmazon Inspector が Amazon EC2 インスタン

ス、Amazon ECR リポジトリ、Amazon ECR コンテナイメージに関する情報の取得などのオペレーションを実行するために必要です。また、 AWS KMS キーで暗号化された Amazon EBS スナップショットを復号することも必要です。詳細については、「Amazon Inspector でのサービスにリンクされたロールの使用」を参照してください。

- organizations – AllowServicePrincipalBasedAccessToOrganizationApis は、サービスにリンクされたロールの作成 AWS アカウント、組織の委任管理者 AWS アカウント としての の登録、および組織の委任管理者の一覧表示のみをサービスプリンシパルに許可します。 AllowOrganizationalBasedAccessToOrganizationApis は、ポリシー所有者が組織単位に関する情報、特にリソースレベルの ARNs を取得できるようにします。 AllowAccountsBasedAccessToOrganizationApis は、ポリシー所有者が に関する情報、特にリソースレベルの ARNs を取得できるようにします AWS アカウント。 AllowAccessToOrganizationApis は、ポリシー所有者が組織および組織情報と AWS のサービス 統合された を表示できるようにします。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "AllowFullAccessToInspectorApis",
      "Effect" : "Allow",
      "Action" : "inspector2:*",
      "Resource" : "*"
    },
    {
      "Sid" : "AllowAccessToCodeGuruApis",
      "Effect" : "Allow",
      "Action" : [
        "codeguru-security:BatchGetFindings",
        "codeguru-security:GetAccountConfiguration"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "AllowAccessToCreateSlr",
      "Effect" : "Allow",
      "Action" : "iam:CreateServiceLinkedRole",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "iam:AWSServiceName" : [
```

```
            "agentless.inspector2.amazonaws.com",
            "inspector2.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid" : "AllowServicePrincipalBasedAccessToOrganizationApis",
      "Effect" : "Allow",
      "Action" : [
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:ListDelegatedAdministrators"
      ],
      "Resource" : "*",
      "Condition": {
        "StringEquals": {
          "organizations:ServicePrincipal": [
            "inspector2.amazonaws.com",
            "agentless.inspector2.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid" : "AllowOrganizationalBasedAccessToOrganizationApis",
      "Effect" : "Allow",
      "Action" : [
        "organizations:DescribeOrganizationalUnit"
      ],
      "Resource" : "arn:*:organizations::*:ou/o-*/ou-*"
    },
    {
      "Sid" : "AllowAccountsBasedAccessToOrganizationApis",
      "Effect" : "Allow",
      "Action" : [
        "organizations:DescribeAccount"
      ],
      "Resource" : "arn:*:organizations::*:account/o-*/*"
    },
    {
      "Sid" : "AllowAccessToOrganizationApis",
      "Effect" : "Allow",
      "Action" : [
```

```
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:DescribeOrganization"
      ],
      "Resource" : "*"
    }
  ]
}
```

AWS マネージドポリシー: AmazonInspector2FullAccess

`AmazonInspector2FullAccess` ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、Amazon Inspector へのフルアクセスを許可する管理許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `inspector2` – Amazon Inspector 機能へのフルアクセスを許可します。
- `iam` – Amazon Inspector がサービスにリンクされたロール `AWSServiceRoleForAmazonInspector2` と `AWSServiceRoleForAmazonInspector2Agentless` を作成できるようにします。`AWSServiceRoleForAmazonInspector2` は、Amazon Inspector が Amazon EC2 インスタンス、Amazon ECR リポジトリ、コンテナイメージに関する情報の取得などのオペレーションを実行するために必要です。また、Amazon Inspector は、VPC ネットワークを分析し、組織に関連付けられているアカウントを記述する必要もあります。`AWSServiceRoleForAmazonInspector2Agentless` は、Amazon Inspector が Amazon EC2 インスタンスや Amazon EBS スナップショットに関する情報の取得などのオペレーションを実行するために必要です。また、 AWS KMS キーで暗号化された Amazon EBS スナップショットを復号することも必要です。詳細については、「Amazon Inspector でのサービスにリンクされたロールの使用」を参照してください。
- `organizations` — 管理者による AWS Organizationsの組織への Amazon Inspector の使用を許可します。で Amazon Inspector の信頼されたアクセスをアクティブ化すると AWS Organizations、委任管理者アカウントのメンバーは設定を管理し、組織全体の結果を表示できます。

- `codeguru-security` – 管理者は Amazon Inspector を使用して、CodeGuru Security が保存しているコードの情報コードスニペットを取得し、暗号化設定を変更できます。詳細については、「検出結果のコードの保管時の暗号化」を参照してください。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowFullAccessToInspectorApis",
   "Effect": "Allow",
   "Action": "inspector2:*",
   "Resource": "*"
  },
  {
   "Sid": "AllowAccessToCodeGuruApis",
   "Effect": "Allow",
   "Action": [
    "codeguru-security:BatchGetFindings",
    "codeguru-security:GetAccountConfiguration"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowAccessToCreateSlr",
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "iam:AWSServiceName": [
      "agentless.inspector2.amazonaws.com",
      "inspector2.amazonaws.com"
     ]
    }
   }
  },
  {
   "Sid": "AllowAccessToOrganizationApis",
   "Effect": "Allow",
   "Action": [
    "organizations:EnableAWSServiceAccess",
    "organizations:RegisterDelegatedAdministrator",
```

```
    "organizations:ListDelegatedAdministrators",
    "organizations:ListAWSServiceAccessForOrganization",
    "organizations:DescribeOrganizationalUnit",
    "organizations:DescribeAccount",
    "organizations:DescribeOrganization"
   ],
   "Resource": "*"
  }
 ]
}
```

AWS 管理ポリシー: AmazonInspector2ReadOnlyAccess

`AmazonInspector2ReadOnlyAccess` ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、Amazon Inspector への読み取り専用アクセスを可能にする許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `inspector2` – Amazon Inspector 機能への読み取り専用アクセスを許可します。
- `organizations` – の組織の Amazon Inspector カバレッジの詳細を表示 AWS Organizations できるようにします。
- `codeguru-security` — CodeGuru Security からコードスニペットを取得できるようにします。また、CodeGuru Security に保存されているコードの暗号化設定を表示することもできます。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "organizations:ListDelegatedAdministrators",
    "organizations:ListAWSServiceAccessForOrganization",
    "organizations:DescribeOrganizationalUnit",
    "organizations:DescribeAccount",
```

```
    "organizations:DescribeOrganization",
    "inspector2:BatchGet*",
    "inspector2:List*",
    "inspector2:Describe*",
    "inspector2:Get*",
    "inspector2:Search*",
    "codeguru-security:BatchGetFindings",
    "codeguru-security:GetAccountConfiguration"
   ],
   "Resource": "*"
  }
 ]
}
```

AWS 管理ポリシー: AmazonInspector2ManagedCisPolicy

IAM エンティティに `AmazonInspector2ManagedCisPolicy` ポリシーをアタッチできます。このポリシーは、Amazon EC2 インスタンスにインスタンスの CIS スキャンの実行を許可するアクセス許可を与えるロールにアタッチする必要があります。IAM ロールを使用して、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、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、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、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `inspector2` – CIS スキャンの実行に使用されるアクションへのアクセスを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
            "inspector2:StartCisSession",
            "inspector2:StopCisSession",
```

```
            "inspector2:SendCisSessionTelemetry",
            "inspector2:SendCisSessionHealth"
            ],
            "Resource": "*",
        }
    ]
}
```

AWS 管理ポリシー: AmazonInspector2ServiceRolePolicy

IAM エンティティに `AmazonInspector2ServiceRolePolicy` ポリシーをアタッチすることはできません。このポリシーは、Amazon Inspector がユーザーに代わってアクションを実行することを許可するサービスリンクロールにアタッチされます。詳細については、「Amazon Inspector でのサービスにリンクされたロールの使用」を参照してください。

AWS 管理ポリシー: AmazonInspector2AgentlessServiceRolePolicy

IAM エンティティに `AmazonInspector2AgentlessServiceRolePolicy` ポリシーをアタッチすることはできません。このポリシーは、Amazon Inspector がユーザーに代わってアクションを実行することを許可するサービスリンクロールにアタッチされます。詳細については、「Amazon Inspector でのサービスにリンクされたロールの使用」を参照してください。

Amazon Inspector による AWS マネージドポリシーの更新

このサービスがこれらの変更の追跡を開始してからの Amazon Inspector の AWS マネージドポリシーの更新に関する詳細を表示します。このページへの変更に関する自動アラートを受け取るには、Amazon Inspector のドキュメント履歴ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonInspector2FullAccess_v2 – 新しいポリシー	Amazon Inspector に、Amazon Inspector へのフルアクセスと他の関連サービスへのアクセスを提供する新しい管理ポリシーが追加されました。	2025 年 7 月 3 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector が IP アドレスとインターネットゲートウェイを記述できるようにする新しいアクセス許可を追加しました。	2025 年 4 月 29 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector に、Amazon ECS および Amazon EKS アクションへの読み取り専用アクセスを許可する新しいアクセス許可が追加されました。	2025 年 3 月 25 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector が AWS Lambdaで関数タグを返すことを許可する新しいアクセス許可を追加しました。	2024 年 7 月 31 日
AmazonInspector2FullAccess — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector がサービスにリンクされたロール `AWSServiceRoleForAmazonInspector2Agentless` を作成できるようにするアクセス許可を追加しました。これにより、ユーザーは Amazon Inspector を有効にするときにエージェントベースのスキャンとエージェントレススキャンを実行できます。	2024 年 4 月 24 日

変更	説明	日付
AmazonInspector2ManagedCisPolicy – 新しいポリシー	Amazon Inspector は、インスタンスプロファイルの一部として使用してインスタンスの CIS スキャンを許可する新しい管理ポリシーを追加しました。	2024 年 1 月 23 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector にターゲットインスタンスで CIS スキャンを開始できるようにする新しいアクセス許可を追加しました。	2024 年 1 月 23 日
AmazonInspector2AgentlessServiceRolePolicy – 新しいポリシー	Amazon Inspector には、EC2 インスタンスのエージェントレススキャンを可能にする、サービスリンクロールの新しいポリシーが追加されました。	2023 年 11 月 27 日
AmazonInspector2ReadOnlyAccess — 既存のポリシーの更新	Amazon Inspector には、読み取り専用ユーザーがパッケージの脆弱性検出結果の脆弱性インテリジェンスの詳細を取得できる新しいアクセス許可が追加されました。	2023 年 9 月 22 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector には、Elastic Load Balancing ターゲットグループの一部である Amazon EC2 インスタンスのネットワーク設定をスキャンできるようにする新しいアクセス許可が追加されました。	2023 年 8 月 31 日
AmazonInspector2ReadOnlyAccess - 既存のポリシーの更新	Amazon Inspector には、読み取り専用ユーザーがリソースのソフトウェア部品表 (SBOM) をエクスポートできる新しいアクセス許可が追加されました。	2023 年 6 月 29 日
AmazonInspector2ReadOnlyAccess - 既存のポリシーの更新	Amazon Inspector には、読み取り専用ユーザーが自分のアカウントの Lambda コードスキャン検出結果の暗号化設定の詳細を取得できるようにする新しいアクセス許可が追加されました。	2023 年 6 月 13 日
AmazonInspector2FullAccess — 既存のポリシーの更新	Amazon Inspector には、Lambda コードスキャンの検出結果に含まれるコードを暗号化するようにカスタマーマネージドキー KMS キーをユーザーが設定できる新しいアクセス許可が追加されました。	2023 年 6 月 13 日

変更	説明	日付
AmazonInspector2ReadOnlyAccess - 既存のポリシーの更新	Amazon Inspector には、読み取り専用ユーザーが自分のアカウントの Lambda コードスキャンのステータスと検出結果の詳細を取得できる新しいアクセス許可が追加されました。	2023 年 5 月 2 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Lambda スキャンをアクティブ化するときに Amazon Inspector がアカウントに AWS CloudTrail サービスにリンクされたチャネルを作成できるようにする新しいアクセス許可を追加しました。これにより、Amazon Inspector はアカウントの CloudTrail イベントをモニタリングできます。	2023 年 4 月 30 日
AmazonInspector2FullAccess — 既存のポリシーの更新	Amazon Inspector には、ユーザーが Lambda コードスキャンから得られたコード脆弱性の検出結果を取得できる新しいアクセス許可が追加されました。	2023 年 4 月 21 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon EC2 詳細検査向けにカスタマーが定義したカスタムパスに関する情報を Amazon EC2 Systems Manager に送信できるようにする新しいアクセス許可を追加しました。	2023 年 4 月 17 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Lambda スキャンをアクティブ化するときに Amazon Inspector がアカウントに AWS CloudTrail サービスにリンクされたチャネルを作成できるようにする新しいアクセス許可を追加しました。これにより、Amazon Inspector はアカウントの CloudTrail イベントをモニタリングできます。	2023 年 4 月 30 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector が AWS Lambda 関数内の開発者コードのスキャンをリクエストし、Amazon CodeGuru Security からスキャンデータを受信できるようにする新しいアクセス許可を追加しました。さらに、Amazon Inspector には、IAM ポリシーを確認するためのアクセス許可が追加されました。Amazon Inspector はこの情報を使用して Lambda 関数のコード脆弱性をスキャンします。	2023 年 2 月 28 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、 AWS Lambda Amazon Inspector が関数が最後に呼び出された日時に関する情報を CloudWatch から取得できるようにする新しいステートメントを追加しました。Amazon Inspector はこの情報を使用して、過去 90 日間にアクティブだった環境内の Lambda 関数にスキャンの対象を絞ります。	2023 年 2 月 20 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector が各 AWS Lambda 関数に関連付けられている各レイヤーバージョンを含む関数に関する情報を取得できるようにする新しいステートメントを追加しました。Amazon Inspector はこの情報を使用して Lambda 関数のセキュリティ脆弱性をスキャンします。	2022 年 11 月 28 日
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector には、Amazon Inspector が SSM 関連付け実行を記述できるようにする新しいアクションが追加されました。さらに、Amazon Inspector では、`AmazonInspector2` 所有の SSM ドキュメントとの SSM 関連付けを作成、更新、削除、および開始できるように、リソーススコープが追加されました。	2022 年 8 月 31 日
AmazonInspector2ServiceRolePolicy 既存のポリシーの更新	Amazon Inspector は、Amazon Inspector が他の AWS パーティションでソフトウェアインベントリを収集できるように、ポリシーのリソーススコープを更新しました。	2022 年 8 月 12 日

変更	説明	日付
AmazonInspector2ServiceRolePolicy — 既存のポリシーの更新	Amazon Inspector はアクションのリソーススコープを再構築し、Amazon Inspector が SSM 関連付けを作成、削除、および更新できるようにしました。	2022 年 8 月 10 日
AmazonInspector2ReadOnlyAccess — 新しいポリシー	Amazon Inspector には、機能への読み取り専用アクセスを許可する新しいポリシーが追加されました。	2022 年 1 月 21 日
AmazonInspector2FullAccess — 新しいポリシー	Amazon Inspector には、機能へのフルアクセスを許可する新しいポリシーが追加されました。	2021 年 11 月 29 日
AmazonInspector2ServiceRolePolicy — 新しいポリシー	Amazon Inspector には、Amazon Inspector がお客様に代わって他のサービスでアクションを実行できるようになりました。	2021 年 11 月 29 日
Amazon Inspector が変更の追跡を開始しました。	Amazon Inspector は AWS 、管理ポリシーの変更の追跡を開始しました。	2021 年 11 月 29 日

Amazon Inspector でのサービスにリンクされたロールの使用

Amazon Inspector は、 という名前の AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します`AWSServiceRoleForAmazonInspector2`。このサービスリンクロールは、Amazon Inspector に直接リンクされた IAM ロールです。これは Amazon Inspector によって事前定義されており、Amazon Inspector が AWS のサービス ユーザーに代わって他の を呼び出すために必要なすべてのアクセス許可が含まれています。

必要な許可を手動で追加する必要がないため、サービスリンクロールは Amazon Inspector のセットアップを容易にします。Amazon Inspector はサービスリンクロールのアクセス許可を定義し、他に定義されていない限り、Amazon Inspector のみがそのロールを引き受けることができます。定義される許可は信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (グループまたはロールなど) に許可するには、アクセス許可を設定する必要があります。詳細についてはIAM ユーザーガイドの「サービスにリンクされた役割のアクセス許可」を参照してください。サービスリンクロールを削除するには、その関連リソースを削除します。これにより、リソースへの意図しないアクセスによる許可の削除が防止され、 Amazon Inspector リソースは保護されます。

サービスにリンクされたロールをサポートする他のサービスの詳細については、AWS 「IAM と連携するサービス」を参照し、「サービスにリンクされたロール」列で「はい」を持つサービスを探します。サービスにリンクされたロールに関するドキュメントをサービスで確認するには、[はい] リンクを選択します。

Amazon Inspector のサービスにリンクされたロールの許可

Amazon Inspector は、 という名前の管理ポリシーを使用しますAWSServiceRoleForAmazonInspector2。このサービスにリンクされたロールは、ロールを引き受ける上で `inspector2.amazonaws.com` サービスを信頼します。

AmazonInspector2ServiceRolePolicy という名前のロールのアクセス許可ポリシーにより、Amazon Inspector は次のようなタスクを実行することが許可されます。

- Amazon Elastic Compute Cloud (Amazon EC2) のアクションを使用して、インスタンスとネットワークパスに関する情報を取得します。
- AWS Systems Manager アクションを使用して Amazon EC2 インスタンスからインベントリを取得し、カスタムパスからサードパーティーパッケージに関する情報を取得します。
- アクションを使用して、ターゲットインスタンスの CIS スキャンを AWS Systems Manager `SendCommand`呼び出します。
- Amazon Elastic Container Registry アクションを使用して、コンテナイメージに関する情報を取得します。
- AWS Lambda アクションを使用して、Lambda 関数に関する情報を取得します。
- AWS Organizations アクションを使用して、関連付けられたアカウントを記述します。
- CloudWatch アクションを使用して、Lambda関数が最後に呼び出されたときの情報を取得します。

- 選択した IAM アクションを使用して、Lambda コードにセキュリティ脆弱性を生じさせる可能性のある IAM ポリシーに関する情報を取得します。
- CodeGuru Security アクションを使用して、Lambda 関数のコードのスキャンを実行します。Amazon Inspector では次の CodeGuru Security を使用します。
  - codeguru-security:CreateScan – CodeGuru Security スキャンを作成するアクセス許可を付与します。
  - codeguru-security:GetScan – CodeGuru Security スキャンメタデータを取得するための許可を付与します。
  - codeguru-security:ListFindings – CodeGuru Security によって生成された検出結果を取得するための許可を付与します。
  - codeguru-security:DeleteScansByCategory – CodeGuru Security が Amazon Inspector によって開始されたスキャンを削除するアクセス許可を付与します。
  - codeguru-security:BatchGetFindings – CodeGuru Security によって生成された特定の検出結果をバッチ取得するための許可を付与します。
- 選択した Elastic Load Balancing アクションを使用して、Elastic Load Balancing ターゲットグループの一部である EC2 インスタンスのネットワークスキャンを実行します。
- Amazon ECS および Amazon EKS アクションを使用して、クラスターとタスクを表示し、タスクを記述するための読み取り専用アクセスを許可します。

ロールは、次のアクセス許可ポリシーを使用して設定されます。

```
    {
"Version": "2012-10-17",
"Statement": [
 {
  "Sid": "TirosPolicy",
  "Effect": "Allow",
  "Action": [
   "directconnect:DescribeConnections",
   "directconnect:DescribeDirectConnectGatewayAssociations",
   "directconnect:DescribeDirectConnectGatewayAttachments",
   "directconnect:DescribeDirectConnectGateways",
   "directconnect:DescribeVirtualGateways",
   "directconnect:DescribeVirtualInterfaces",
   "ec2:DescribeAddresses",
   "ec2:DescribeAvailabilityZones",
```

```
    "ec2:DescribeCustomerGateways",
    "ec2:DescribeEgressOnlyInternetGateways",
    "ec2:DescribeInstances",
    "ec2:DescribeInternetGateways",
    "ec2:DescribeManagedPrefixLists",
    "ec2:DescribeNatGateways",
    "ec2:DescribeNetworkAcls",
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribePrefixLists",
    "ec2:DescribeRegions",
    "ec2:DescribeRouteTables",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeTransitGatewayAttachments",
    "ec2:DescribeTransitGatewayConnects",
    "ec2:DescribeTransitGatewayPeeringAttachments",
    "ec2:DescribeTransitGatewayRouteTables",
    "ec2:DescribeTransitGatewayVpcAttachments",
    "ec2:DescribeTransitGateways",
    "ec2:DescribeVpcEndpointServiceConfigurations",
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeVpcPeeringConnections",
    "ec2:DescribeVpcs",
    "ec2:DescribeVpnConnections",
    "ec2:DescribeVpnGateways",
    "ec2:GetManagedPrefixListEntries",
    "ec2:GetTransitGatewayRouteTablePropagations",
    "ec2:SearchTransitGatewayRoutes",
    "elasticloadbalancing:DescribeListeners",
    "elasticloadbalancing:DescribeLoadBalancerAttributes",
    "elasticloadbalancing:DescribeLoadBalancers",
    "elasticloadbalancing:DescribeRules",
    "elasticloadbalancing:DescribeTags",
    "elasticloadbalancing:DescribeTargetGroups",
    "elasticloadbalancing:DescribeTargetGroupAttributes",
    "elasticloadbalancing:DescribeTargetHealth",
    "network-firewall:DescribeFirewall",
    "network-firewall:DescribeFirewallPolicy",
    "network-firewall:DescribeResourcePolicy",
    "network-firewall:DescribeRuleGroup",
    "network-firewall:ListFirewallPolicies",
    "network-firewall:ListFirewalls",
    "network-firewall:ListRuleGroups",
    "tiros:CreateQuery",
```

```
    "tiros:GetQueryAnswer"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Sid": "PackageVulnerabilityScanning",
   "Effect": "Allow",
   "Action": [
    "ecr:BatchGetImage",
    "ecr:BatchGetRepositoryScanningConfiguration",
    "ecr:DescribeImages",
    "ecr:DescribeRegistry",
    "ecr:DescribeRepositories",
    "ecr:GetAuthorizationToken",
    "ecr:GetDownloadUrlForLayer",
    "ecr:GetRegistryScanningConfiguration",
    "ecr:ListImages",
    "ecr:PutRegistryScanningConfiguration",
    "organizations:DescribeAccount",
    "organizations:DescribeOrganization",
    "organizations:ListAccounts",
    "ssm:DescribeAssociation",
    "ssm:DescribeAssociationExecutions",
    "ssm:DescribeInstanceInformation",
    "ssm:ListAssociations",
    "ssm:ListResourceDataSync"
   ],
   "Resource": "*"
  },
  {
   "Sid": "LambdaPackageVulnerabilityScanning",
   "Effect": "Allow",
   "Action": [
    "lambda:ListFunctions",
    "lambda:GetFunction",
    "lambda:GetLayerVersion",
    "lambda:ListTags",
    "cloudwatch:GetMetricData"
   ],
   "Resource": "*"
  },
  {
```

```
    "Sid": "GatherInventory",
    "Effect": "Allow",
    "Action": [
     "ssm:CreateAssociation",
     "ssm:StartAssociationsOnce",
     "ssm:UpdateAssociation"
    ],
    "Resource": [
     "arn:aws:ec2:*:*:instance/*",
     "arn:aws:ssm:*:*:document/AmazonInspector2-*",
     "arn:aws:ssm:*:*:document/AWS-GatherSoftwareInventory",
     "arn:aws:ssm:*:*:managed-instance/*",
     "arn:aws:ssm:*:*:association/*"
    ]
   },
   {
    "Sid": "GatherInventoryDeleteAssociation",
    "Effect": "Allow",
    "Action": [
     "ssm:DeleteAssociation"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:association/*"
    ]
   },
   {
    "Sid": "DataSyncCleanup",
    "Effect": "Allow",
    "Action": [
     "ssm:CreateResourceDataSync",
     "ssm:DeleteResourceDataSync"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:resource-data-sync/InspectorResourceDataSync-do-not-delete"
    ]
   },
   {
    "Sid": "ManagedRules",
    "Effect": "Allow",
    "Action": [
     "events:PutRule",
     "events:DeleteRule",
     "events:DescribeRule",
     "events:ListTargetsByRule",
```

```
    "events:PutTargets",
    "events:RemoveTargets"
   ],
   "Resource": [
    "arn:aws:events:*:*:rule/DO-NOT-DELETE-AmazonInspector*ManagedRule"
   ]
  },
  {
   "Sid": "LambdaCodeVulnerabilityScanning",
   "Effect": "Allow",
   "Action": [
    "codeguru-security:CreateScan",
    "codeguru-security:GetAccountConfiguration",
    "codeguru-security:GetFindings",
    "codeguru-security:GetScan",
    "codeguru-security:ListFindings",
    "codeguru-security:BatchGetFindings",
    "codeguru-security:DeleteScansByCategory"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Sid": "CodeGuruCodeVulnerabilityScanning",
   "Effect": "Allow",
   "Action": [
    "iam:GetRole",
    "iam:GetRolePolicy",
    "iam:GetPolicy",
    "iam:GetPolicyVersion",
    "iam:ListAttachedRolePolicies",
    "iam:ListPolicies",
    "iam:ListPolicyVersions",
    "iam:ListRolePolicies",
    "lambda:ListVersionsByFunction"
   ],
   "Resource": [
    "*"
   ],
   "Condition": {
    "ForAnyValue:StringEquals": {
     "aws:CalledVia": [
      "codeguru-security.amazonaws.com"
```

```
      ]
     }
    }
   },
   {
    "Sid": "Ec2DeepInspection",
    "Effect": "Allow",
    "Action": [
     "ssm:PutParameter",
     "ssm:GetParameters",
     "ssm:DeleteParameter"
    ],
    "Resource": [
     "arn:aws:ssm:*:*:parameter/inspector-aws/service/inspector-linux-application-paths"
    ],
    "Condition": {
     "StringEquals": {
      "aws:ResourceAccount": "${aws:PrincipalAccount}"
     }
    }
   },
   {
    "Sid": "AllowManagementOfServiceLinkedChannel",
    "Effect": "Allow",
    "Action": [
     "cloudtrail:CreateServiceLinkedChannel",
     "cloudtrail:DeleteServiceLinkedChannel"
    ],
    "Resource": [
     "arn:aws:cloudtrail:*:*:channel/aws-service-channel/inspector2/*"
    ],
    "Condition": {
     "StringEquals": {
      "aws:ResourceAccount": "${aws:PrincipalAccount}"
     }
    }
   },
   {
    "Sid": "AllowListServiceLinkedChannels",
    "Effect": "Allow",
    "Action": [
     "cloudtrail:ListServiceLinkedChannels"
    ],
    "Resource": [
```

```
      "*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceAccount": "${aws:PrincipalAccount}"
      }
    }
  },
  {
    "Sid": "AllowToRunInvokeCisSpecificDocuments",
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand",
      "ssm:GetCommandInvocation"
    ],
    "Resource": [
      "arn:aws:ssm:*:*:document/AmazonInspector2-InvokeInspectorSsmPluginCIS"
    ]
  },
  {
    "Sid": "AllowToRunCisCommandsToSpecificResources",
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:instance/*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceAccount": "${aws:PrincipalAccount}"
      }
    }
  },
  {
    "Sid": "AllowToPutCloudwatchMetricData",
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
```

```
     "StringEquals": {
      "cloudwatch:namespace": "AWS/Inspector2"
     }
    }
   },
   {
    "Sid": "AllowListAccessToECSAndEKS",
    "Effect": "Allow",
    "Action": [
        "ecs:ListClusters",
        "ecs:ListTasks",
        "eks:ListClusters"
    ],
    "Resource": [
        "*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
    },
    {
    "Sid": "AllowAccessToECSTasks",
    "Effect": "Allow",
    "Action": [
        "ecs:DescribeTasks"
    ],
    "Resource": "arn:aws:ecs:*:*:task/*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
     }
    }
   }
  ]
}
```

Amazon Inspector のサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS Management Console、、 AWS CLIまたは AWS API で Amazon Inspector をアクティブ化すると、Amazon Inspector によってサービスにリンクされたロールが作成されます。

Amazon Inspector のサービスリンクロールの編集

Amazon Inspector では、`AWSServiceRoleForAmazonInspector2` のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールが作成されると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

Amazon Inspector のサービスリンクロールの削除

Amazon Inspector の使用が不要になった場合は、`AWSServiceRoleForAmazonInspector2` サービスにリンクされたロールを削除することをお勧めします。ロールを削除する前に、アクティブ化 AWS リージョン されている各 で Amazon Inspector を非アクティブ化する必要があります。Amazon Inspector を非アクティブ化しても、ロールは削除されません。したがって、Amazon Inspector を再度アクティブ化すると、既存のロールを使用することができます。そうすることで、使用していないエンティティがアクティブにモニタリングまたはメンテナンスされるのを防ぐことができます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

サービスにリンクされたこのロールを削除したが、再作成する必要がある場合は、同じプロセスで、アカウントにロールを再作成することができます。Amazon Inspector をアクティブ化すると、Amazon Inspector がサービスリンクロールを再作成します。

> **Note**
>
> リソースを削除しようとしているときに Amazon Inspector サービスがロールを使用している場合は、削除が失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。

IAM コンソール、 AWS CLI、または AWS API を使用して、`AWSServiceRoleForAmazonInspector2`サービスにリンクされたロールを削除できます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

Amazon Inspector のエージェントレススキャンに対するサービスリンクロールのアクセス許可

Amazon Inspector のエージェントレススキャンでは、`AWSServiceRoleForAmazonInspector2Agentless` という名前のサービスリンクロールを使用します。このサービスリンクロールにより、Amazon Inspector はアカウントに Amazon EBS ボリュームのスナップショットを作成し、そのスナップショットからデータにアクセスできるようになります。このサービスリンクロールは、ロールを引き受ける上で `agentless.inspector2.amazonaws.com` サービスを信頼します。

> **Important**
>
> このサービスリンクロールのステートメントにより、Amazon Inspector は、`InspectorEc2Exclusion` タグを使用してスキャンから除外した EC2 インスタンスに対してエージェントレススキャンを実行できなくなります。さらに、このステートメントは、暗号化に使用される KMS キーに `InspectorEc2Exclusion` タグが付いている場合に、Amazon Inspector がボリュームの暗号化されたデータにアクセスできないようにします。詳細については、「Amazon Inspector スキャンからのインスタンスの除外」を参照してください。

`AmazonInspector2AgentlessServiceRolePolicy` という名前のロールのアクセス許可ポリシーにより、Amazon Inspector は次のようなタスクを実行することが許可されます。

- Amazon Elastic Compute Cloud (Amazon EC2) アクションを使用して、EC2 インスタンス、ボリューム、スナップショットに関する情報を取得します。
  - Amazon EC2 のタグ付けアクションを使用して、`InspectorScan` タグキーでスキャンのスナップショットにタグを付けます。
  - Amazon EC2 のスナップショットアクションを使用してスナップショットを作成し、`InspectorScan` タグキーでタグ付けしてから、`InspectorScan` タグキーでタグ付けされた Amazon EBS ボリュームのスナップショットを削除します。
- Amazon EBS アクションを使用して、`InspectorScan` タグキーでタグ付けされたスナップショットから情報を取得します。
- Select AWS KMS 復号アクションを使用して、 AWS KMS カスタマーマネージドキーで暗号化されたスナップショットを復号します。Amazon Inspector は、スナップショットの暗号化に使用された KMS キーに `InspectorEc2Exclusion` タグが付いている場合、スナップショットを復号しません。

ロールは、次のアクセス許可ポリシーを使用して設定されます。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "InstanceIdentification",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeInstances",
    "ec2:DescribeVolumes",
    "ec2:DescribeSnapshots"
   ],
   "Resource": "*"
  },
  {
   "Sid": "GetSnapshotData",
   "Effect": "Allow",
   "Action": [
    "ebs:ListSnapshotBlocks",
    "ebs:GetSnapshotBlock"
   ],
   "Resource": "arn:aws:ec2:*:*:snapshot/*",
   "Condition": {
    "StringLike": {
     "aws:ResourceTag/InspectorScan": "*"
    }
   }
  },
  {
   "Sid": "CreateSnapshotsAnyInstanceOrVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateSnapshots",
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*"
   ]
  },
  {
   "Sid": "DenyCreateSnapshotsOnExcludedInstances",
   "Effect": "Deny",
   "Action": "ec2:CreateSnapshots",
   "Resource": "arn:aws:ec2:*:*:instance/*",
```

```
    "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/InspectorEc2Exclusion": "true"
     }
    }
   },
   {
    "Sid": "CreateSnapshotsOnAnySnapshotOnlyWithTag",
    "Effect": "Allow",
    "Action": "ec2:CreateSnapshots",
    "Resource": "arn:aws:ec2:*:*:snapshot/*",
    "Condition": {
     "Null": {
      "aws:TagKeys": "false"
     },
     "ForAllValues:StringEquals": {
      "aws:TagKeys": "InspectorScan"
     }
    }
   },
   {
    "Sid": "CreateOnlyInspectorScanTagOnlyUsingCreateSnapshots",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:snapshot/*",
    "Condition": {
     "StringLike": {
      "ec2:CreateAction": "CreateSnapshots"
     },
     "Null": {
      "aws:TagKeys": "false"
     },
     "ForAllValues:StringEquals": {
      "aws:TagKeys": "InspectorScan"
     }
    }
   },
   {
    "Sid": "DeleteOnlySnapshotsTaggedForScanning",
    "Effect": "Allow",
    "Action": "ec2:DeleteSnapshot",
    "Resource": "arn:aws:ec2:*:*:snapshot/*",
    "Condition": {
     "StringLike": {
```

```
      "ec2:ResourceTag/InspectorScan": "*"
     }
    }
   },
   {
    "Sid": "DenyKmsDecryptForExcludedKeys",
    "Effect": "Deny",
    "Action": "kms:Decrypt",
    "Resource": "arn:aws:kms:*:*:key/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/InspectorEc2Exclusion": "true"
     }
    }
   },
   {
    "Sid": "DecryptSnapshotBlocksVolContext",
    "Effect": "Allow",
    "Action": "kms:Decrypt",
    "Resource": "arn:aws:kms:*:*:key/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceAccount": "${aws:PrincipalAccount}"
     },
     "StringLike": {
      "kms:ViaService": "ec2.*.amazonaws.com",
      "kms:EncryptionContext:aws:ebs:id": "vol-*"
     }
    }
   },
   {
    "Sid": "DecryptSnapshotBlocksSnapContext",
    "Effect": "Allow",
    "Action": "kms:Decrypt",
    "Resource": "arn:aws:kms:*:*:key/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceAccount": "${aws:PrincipalAccount}"
     },
     "StringLike": {
      "kms:ViaService": "ec2.*.amazonaws.com",
      "kms:EncryptionContext:aws:ebs:id": "snap-*"
     }
    }
```

```
    },
    {
     "Sid": "DescribeKeysForEbsOperations",
     "Effect": "Allow",
     "Action": "kms:DescribeKey",
     "Resource": "arn:aws:kms:*:*:key/*",
     "Condition": {
      "StringEquals": {
       "aws:ResourceAccount": "${aws:PrincipalAccount}"
      },
      "StringLike": {
       "kms:ViaService": "ec2.*.amazonaws.com"
      }
     }
    },
    {
     "Sid": "ListKeyResourceTags",
     "Effect": "Allow",
     "Action": "kms:ListResourceTags",
     "Resource": "arn:aws:kms:*:*:key/*"
    }
   ]
}
```

エージェントレススキャンのサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS Management Console、、 AWS CLIまたは AWS API で Amazon Inspector をアクティブ化すると、Amazon Inspector によってサービスにリンクされたロールが作成されます。

エージェントレススキャンのサービスリンクロールの編集

Amazon Inspector では、`AWSServiceRoleForAmazonInspector2Agentless` のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールが作成されると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

エージェントレススキャンのサービスリンクロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。これにより、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。

> **Important**
>
> `AWSServiceRoleForAmazonInspector2Agentless` ロールを削除するには、エージェントレススキャンが可能なすべてのリージョンで、スキャンモードをエージェントベースに設定する必要があります。

サービスリンクロールを IAM で手動削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonInspector2Agentless サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

Amazon Inspector アイデンティティとアクセスのトラブルシューティング

以下の情報を使用して、Amazon Inspector と IAM 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てます。

トピック

- Amazon Inspector でアクションを実行する認可がない
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに Amazon Inspector リソース AWS アカウント へのアクセスを許可したい

Amazon Inspector でアクションを実行する認可がない

あるアクションを実行するアクセス許可がないというエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、`mateojackson` IAM ユーザーがコンソールを使用して、ある *`my-example-widget`* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な`inspector2:`*`GetWidget`* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 inspector2:GetWidget on resource: my-example-widget
```

この場合、`inspector2:GetWidget` アクションを使用して `my-example-widget`リソースへのアクセスを許可するように、`mateojackson` ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon Inspector にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下のエラーの例は、`marymajor` という名前の IAM ユーザーがコンソールを使用して Amazon Inspector でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon Inspector 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:

- Amazon Inspector 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、「Amazon Inspector と IAM の連携」を参照してください。
- 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 の IAM ユーザー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おけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

Amazon Inspector のモニタリング

モニタリングは、Amazon Inspector やその他の AWS ソリューションの可用性、信頼性、パフォーマンスを維持する上で重要な部分です。 AWS には、Amazon Inspector をモニタリングし、発生した問題を報告し、これらの問題を修正するためのアクションを実行するためのツールが用意されています。

- Amazon EventBridge は、イベントを使用してアプリケーションコンポーネントを接続し、スケーラブルなイベント駆動型アプリケーションを簡単に構築できるようにする AWS サービスです。EventBridge は、アプリケーション、Software-as-a-Service (SaaS) アプリケーション、 AWS サービスとルートからリアルタイムデータのストリームを配信するため、サービスで発生するイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。
- AWS CloudTrail は、 によって、または に代わって行われた API コールおよび関連イベントをキャプチャする AWS サービスです AWS アカウント。CloudTrail は、指定した Amazon S3 バケットにログファイルを配信するため、 を呼び出したユーザーとアカウント AWS、呼び出し元のソース IP アドレス、呼び出しの発生日時を特定できます。

AWS CloudTrailでの Amazon Inspector API コールのログ記録

Amazon Inspector は AWS CloudTrail、Amazon Inspector の IAM ユーザーまたはロール、または によって実行されたアクションを記録するサービス AWS のサービスである と統合されています。CloudTrail は、Amazon Inspector へのすべての 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、Amazon Inspector コンソールからの呼び出しと、Amazon Inspector API オペレーションへの呼び出しが含まれます。追跡を作成する場合は、Amazon Inspector 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。証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 によって収集されたデータを使用して、以下の情報を判断できます。

- Amazon Inspector に対して行われたリクエスト
- リクエストが行われた IP アドレス。
- 誰がリクエストを行ったか。
- リクエストが行われた時。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail での Amazon Inspector 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。Amazon Inspector 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] で AWS のサービス のその他のサービスのイベントと共に CloudTrail イベントにレコードされます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。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。

Amazon Inspector のイベントなど AWS アカウント、 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きに、証跡がすべての 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 AWS のサービス するように他の を設定できます。詳細については、以下の各トピック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」
- 複数のアカウントから CloudTrail ログファイルを受け取る
- CloudTrail ログ ファイルを複数のリージョンから受け取る

すべての Amazon Inspector アクションが Amazon CloudTrail により口グに記録されます。Amazon Inspector で実行できるすべてのアクションは、「Amazon Inspector API リファレンス」に記載されています。例えば、`CreateFindingsReport`、`ListCoverage`、`UpdateOrganizationConfiguration` 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、ルートユーザーまたは IAM ユーザーのどちらの認証情報を使用して送信されたかどうか。
- リクエストが、ロールとフェデレーティッドユーザーのどちらかの一時的なセキュリティ認証情報を使用して送信されたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

Amazon Inspector ログファイルエントリの理解

「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 ログファイルには、1 つ以上のログエントリがあります。イベント は、任意の送信元からの単一の要求を表します。イベントには、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、パブリック 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。

CloudTrail での Amazon Inspector スキャンの情報

Amazon Inspector スキャンは CloudTrail と統合されています。Amazon Inspector スキャン API オペレーションはすべて管理イベントとしてログ記録されます。Amazon Inspector が CloudTrail にログ記録する Amazon Inspector スキャン API オペレーションのリストについては、「Amazon Inspector API リファレンス」の「Amazon Inspector スキャン」を参照してください。

以下は、ScanSbom アクションを示す CloudTrail ログエントリの例です。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA123456789EXAMPLE:akua_mansa",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/akua_mansa",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA123456789EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-10-17T15:22:59Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-10-17T16:02:34Z",
    "eventSource": "gamma-inspector-scan.amazonaws.com",
    "eventName": "ScanSbom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-sdk-java/2.20.162 Mac_OS_X/13.5.2 OpenJDK_64-
Bit_Server_VM/17.0.8+7-LTS Java/17.0.8 vendor/Amazon.com_Inc. io/sync http/
UrlConnection cfg/retry-mode/legacy",
    "requestParameters": {
        "sbom": {
            "specVersion": "1.5",
            "metadata": {
                "component": {
                    "name": "debian",
                    "type": "operating-system",
                    "version": "9"
                }
            },
```

```
            "components": [
                {
                    "name": "packageOne",
                    "purl": "pkg:deb/debian/packageOne@1.0.0?arch=x86_64&distro=9",
                    "type": "application"
                }
            ],
            "bomFormat": "CycloneDX"
        }
    },
    "responseElements": null,
    "requestID": "f041a27f-f33e-4f70-b09b-5fbc5927282a",
    "eventID": "abc8d1e4-d214-4f07-bc56-8a31be6e36fe",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Amazon Inspector 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、AWS のサービス 「コンプライアンスプログラムによる対象範囲内」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 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス 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、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- 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 コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 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。このガイドは、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) を含む) のセキュリティコントロールを保護し、そのガイダンスに AWS のサービス マッピングするためのベストプラクティスをまとめたものです。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、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 – 不審な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 AWS のサービス 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知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 – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon Inspector の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、および高度に冗長なネットワークに接続されている、物理的に分離および隔離された複数のアベイラビリティーゾーン AWS リージョン を提供します。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イ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が高く、フォールトトレラントで、スケーラブルです。

Amazon Inspector 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 Amazon Inspector 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 がインフラストラクチャ AWS 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、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で Amazon Inspector にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Inspector でのインシデントへの対応

AWSでは、セキュリティが最優先事項です。「クラウドのセキュリティ」のAWS 「 責任共有モデル」で説明されているように、 AWS は AWS クラウド内のすべてのサービスを実行するインフラストラクチャを保護する責任があります。 AWS は、Amazon Inspector サービスに関連するインシデント対応についても責任を負います。

お客様は AWS 、 AWS クラウドでセキュリティを維持する責任を共有します。つまり、実装するセキュリティを制御します。これには、アクセスするすべての AWS ツールと機能が含まれます。また、責任共有モデルのユーザー側でインシデント対応を行う責任があることを意味します。

AWS クラウドで実行されているアプリケーションのすべての目的を満たすセキュリティベースラインを確立することで、対応できる逸脱を検出できます。インシデント対応は複雑なトピックであり、インシデント対応の影響と、選択した内容がお客様の企業目標にどのように影響するかをより良く理解するために、次のリソースをご確認ください:「AWS セキュリティインシデント対応ガイド」、「AWS セキュリティのベストプラクティス」、「AWS クラウド導入フレームワーク: セキュリティの観点」。

Amazon Inspector インターフェイスエンドポイント (AWS PrivateLink

を使用して AWS PrivateLink 、VPC と Amazon Inspector 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 AWS Direct Connect 接続を使用せずに、VPC 内にあるかのように Amazon Inspector にアクセスできます。VPC 内のインスタンスは、Amazon Inspector 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を利用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、Amazon Inspector 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「Access AWS のサービス through AWS PrivateLink」を参照してください。

Amazon Inspector に関する考慮事項

Amazon Inspector 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。

Amazon Inspector は、インターフェイスエンドポイントを介したすべての API アクションの呼び出しをサポートしています。

VPC エンドポイントポリシーは Amazon Inspector ではサポートされていません。デフォルトでは、インターフェイスエンドポイントを介して Amazon Inspector へのフルアクセスが許可されます。または、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、インターフェイスエンドポイントを介して Amazon Inspector へのトラフィックを制御することもできます。

Amazon Inspector のインターフェイスエンドポイントを作成する

Amazon Inspector 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 AWS Command Line Interface () を使用して作成できますAWS CLI。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を作成」を参照してください。

Amazon Inspector のインターフェイスエンドポイントを作成するときは、次のいずれかのサービス名を使用します。

```
com.amazonaws.region.inspector2
```

```
com.amazonaws.region.inspector-scan
```

region を該当する の AWS リージョン コードに置き換えます AWS リージョン。

インターフェイスエンドポイントのプライベート DNS を有効にすると、例えば、デフォルトのリージョン DNS 名を使用して、`service-name.us-east-1.amazonaws.com` または米国東部 (バージニア北部) `service-name.us-east-1.api.aws.com`に対して、Amazon Inspector への API リクエストを行うことができます。

Amazon Inspector との統合

Amazon Inspector は他の AWS サービスと統合されます。これらのサービスは Amazon Inspector からデータを取り込むことができるため、さまざまな方法で検出結果を表示できます。詳細については、次の統合オプションを参照してください。

Amazon Inspector と Amazon ECR の統合

Amazon Elastic Container Registry (Amazon ECR) は、プライベートレジストリをサポートする AWSマネージドコンテナイメージレジストリです。Amazon ECR プライベートレジストリは、可用性の高いスケーラブルなアーキテクチャでコンテナイメージをホストします。Amazon Inspector を使用して、Amazon ECR リポジトリにあるコンテナイメージをスキャンし、脆弱なオペレーティングシステムパッケージやプログラミング言語パッケージを確認することができます。詳細については、「Amazon Elastic Container Registry (Amazon ECR) と、Amazon Inspector の統合」を参照してください。

Amazon Inspector と の統合 AWS Security Hub

AWS Security Hub は、 のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスに照らして環境をチェックするのに役立ちます。Security Hub は、 AWS アカウント、サービス、およびサポートされている製品からセキュリティデータを収集します。Security Hub を使用して Amazon Inspector の検出結果データを取り込んで、統合されたすべての AWS サービスと AWS Partner Network 製品で検出結果の一元的な場所を作成できます。詳細については、「Amazon Inspector と の統合 AWS Security Hub」を参照してください。

Amazon Elastic Container Registry (Amazon ECR) と、Amazon Inspector の統合

Amazon Elastic Container Registry は、Docker および OCI イメージと AWS アーティファクトをサポートするフルマネージドコンテナレジストリです。Amazon ECR を使用する場合は、コンテナレジストリの拡張スキャンをアクティブ化できます。拡張スキャンをアクティブ化すると、Amazon Inspector はコンテナイメージを自動的に検出し、脆弱なオペレーティングシステムパッケージやプログラミング言語パッケージをスキャンします。この統合により、コンテナイメージに関する Amazon Inspector の検出結果を表示し、Amazon ECR コンソールでのスキャンの頻度と範囲を管理できるようになります。詳細については、「Amazon Inspector による Amazon ECR コンテナイメージのスキャン」を参照してください。

統合をアクティブ化する

統合をアクティブ化するには、Amazon Inspector コンソールまたは API を使用して Amazon Inspector のスキャンをアクティブ化するか、Amazon ECR コンソールまたは API を使用して Amazon Inspector による拡張スキャンを使用するようにリポジトリを設定します。

Amazon Inspector を使用して統合をアクティブ化する方法の詳細については、「Amazon Inspector の自動スキャンタイプ」を参照してください。

Amazon ECR での拡張スキャンのアクティブ化と設定については、「Amazon ECR ユーザーガイド」の「拡張スキャン」を参照してください。

マルチアカウント環境との統合を使用する

マルチアカウント環境のメンバーであれば、Amazon ECR から拡張スキャンをアクティブ化にできます。ただし、一度アクティブ化すると、Amazon Inspector の委任された管理者だけが非アクティブ化できます。非アクティブ化すると、「基本的なスキャン」に戻ります。詳細については、「Amazon Inspector の非アクティブ化」を参照してください。

Amazon Inspector と の統合 AWS Security Hub

Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。これにより、セキュリティ業界標準とベストプラクティスに照らして環境をチェックできます。Security Hub は、 AWS アカウント、サービス、およびサポートされている製品からセキュリティデータを収集します。この情報を使用して、セキュリティの傾向を分析し、セキュリティの問題を特定できます。Amazon Inspector と Security Hub の統合を有効にすると、Amazon Inspector は Security Hub に結果を送信し、Security Hub はセキュリティ体制の一部としてこれらの結果を分析できます。

Security Hub は、セキュリティの問題を調査結果として追跡します。一部の検出結果は、他の AWS サービスまたはサードパーティー製品で検出されたセキュリティ上の問題が原因である可能性があります。Security Hub は、一連のルールを使用してセキュリティ問題を検出し、検出結果を生成してツールを提供するため、検出結果を管理できます。Security Hub は、Amazon Inspector の検出結果が Amazon Inspector で終了された時点で、それらの検出結果をアーカイブします。検出結果の履歴と検出結果の詳細を表示したり、検出結果の調査ステータスを追跡したりすることもできます。

Security Hub は、AWS Security Finding 形式 (ASFF) で検出結果を処理します。この形式には、一意の識別子、重要度レベル、影響を受けるリソース、修復ガイダンス、ワークフローステータス、コンテキスト情報などの詳細が含まれます。

> **Note**
>
> Amazon Inspector Code Security によって生成されたセキュリティ検出結果は、この統合では使用できません。ただし、これらの特定の検出結果には、Amazon Inspector コンソールおよび Amazon Inspector API からアクセスできます。

トピック

- での Amazon Inspector の検出結果の表示 AWS Security Hub
- Amazon Inspector と Security Hub の統合のアクティブ化と設定
- 統合先からの検出結果フローの無効化
- Security Hub での Amazon Inspector のセキュリティコントロールの表示

での Amazon Inspector の検出結果の表示 AWS Security Hub

Amazon Inspector Classic と Amazon Inspector の検出結果は Security Hub で表示できます。

> **Note**
>
> Amazon Inspector の検出結果のみをフィルタリングするには、`"aws/inspector/ProductVersion": "2"` をフィルターバーに追加します。このフィルターは、Security Hub ダッシュボードから Amazon Inspector Classic の検出結果を除外します。

Amazon Inspector の検出結果例

```
{
  "SchemaVersion": "2018-10-08",
  "Id": "arn:aws:inspector2:us-east-1:123456789012:finding/FINDING_ID",
  "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/inspector",
  "ProductName": "Inspector",
  "CompanyName": "Amazon",
  "Region": "us-east-1",
  "GeneratorId": "AWSInspector",
  "AwsAccountId": "123456789012",
  "Types": [
    "Software and Configuration Checks/Vulnerabilities/CVE"
  ],
```

```
  "FirstObservedAt": "2023-01-31T20:25:38Z",
  "LastObservedAt": "2023-05-04T18:18:43Z",
  "CreatedAt": "2023-01-31T20:25:38Z",
  "UpdatedAt": "2023-05-04T18:18:43Z",
  "Severity": {
    "Label": "HIGH",
    "Normalized": 70
  },
  "Title": "CVE-2022-34918 - kernel",
  "Description": "An issue was discovered in the Linux kernel through 5.18.9. A type
 confusion bug in nft_set_elem_init (leading to a buffer overflow) could be used by a
 local attacker to escalate privileges, a different vulnerability than CVE-2022-32250.
 (The attacker can obtain root access, but must start with an unprivileged user
 namespace to obtain CAP_NET_ADMIN access.) This can be fixed in nft_setelem_parse_data
 in net/netfilter/nf_tables_api.c.",
  "Remediation": {
    "Recommendation": {
      "Text": "Remediation is available. Please refer to the Fixed version in the
 vulnerability details section above. For detailed remediation guidance for each of the
 affected packages, refer to the vulnerabilities section of the detailed finding JSON."
    }
  },
  "ProductFields": {
    "aws/inspector/FindingStatus": "ACTIVE",
    "aws/inspector/inspectorScore": "7.8",
    "aws/inspector/resources/1/resourceDetails/awsEc2InstanceDetails/platform":
 "AMAZON_LINUX_2",
    "aws/inspector/ProductVersion": "2",
    "aws/inspector/instanceId": "i-0f1ed287081bdf0fb",
    "aws/securityhub/FindingId": "arn:aws:securityhub:us-east-1::product/aws/inspector/
arn:aws:inspector2:us-east-1:123456789012:finding/FINDING_ID",
    "aws/securityhub/ProductName": "Inspector",
    "aws/securityhub/CompanyName": "Amazon"
  },
  "Resources": [
    {
      "Type": "AwsEc2Instance",
      "Id": "arn:aws:ec2:us-east-1:123456789012:i-0f1ed287081bdf0fb",
      "Partition": "aws",
      "Region": "us-east-1",
      "Tags": {
        "Patch Group": "SSM",
        "Name": "High-SEv-Test"
      },
```

```
      "Details": {
        "AwsEc2Instance": {
          "Type": "t2.micro",
          "ImageId": "ami-0cff7528ff583bf9a",
          "IpV4Addresses": [
            "52.87.229.97",
            "172.31.57.162"
          ],
          "KeyName": "ACloudGuru",
          "IamInstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/
AmazonSSMRoleForInstancesQuickSetup",
          "VpcId": "vpc-a0c2d7c7",
          "SubnetId": "subnet-9c934cb1",
          "LaunchedAt": "2022-07-26T21:49:46Z"
        }
      }
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ACTIVE",
  "Vulnerabilities": [
    {
      "Id": "CVE-2022-34918",
      "VulnerablePackages": [
        {
          "Name": "kernel",
          "Version": "5.10.118",
          "Epoch": "0",
          "Release": "111.515.amzn2",
          "Architecture": "X86_64",
          "PackageManager": "OS",
          "FixedInVersion": "0:5.10.130-118.517.amzn2",
          "Remediation": "yum update kernel"
        }
      ],
      "Cvss": [
        {
          "Version": "2.0",
          "BaseScore": 7.2,
          "BaseVector": "AV:L/AC:L/Au:N/C:C/I:C/A:C",
          "Source": "NVD"
```

```
        },
        {
          "Version": "3.1",
          "BaseScore": 7.8,
          "BaseVector": "CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H",
          "Source": "NVD"
        },
        {
          "Version": "3.1",
          "BaseScore": 7.8,
          "BaseVector": "CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H",
          "Source": "NVD",
          "Adjustments": []
        }
      ],
      "Vendor": {
        "Name": "NVD",
        "Url": "https://nvd.nist.gov/vuln/detail/CVE-2022-34918",
        "VendorSeverity": "HIGH",
        "VendorCreatedAt": "2022-07-04T21:15:00Z",
        "VendorUpdatedAt": "2022-10-26T17:05:00Z"
      },
      "ReferenceUrls": [
        "https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net.git/commit/?
id=7e6bc1f6cabcd30aba0b11219d8e01b952eacbb6",
        "https://lore.kernel.org/netfilter-devel/cd9428b6-7ffb-dd22-d949-
d86f4869f452@randorisec.fr/T/",
        "https://www.debian.org/security/2022/dsa-5191"
      ],
      "FixAvailable": "YES"
    }
  ],
  "FindingProviderFields": {
    "Severity": {
      "Label": "HIGH"
    },
    "Types": [
      "Software and Configuration Checks/Vulnerabilities/CVE"
    ]
  },
  "ProcessedAt": "2023-05-05T20:28:38.822Z"
}
```

Amazon Inspector と Security Hub の統合のアクティブ化と設定

Security Hub を有効にする AWS Security Hub と、 との Amazon Inspector 統合をアクティブ化できます。 https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-settingup.htmlSecurity Hub を有効にすると、Amazon Inspector と の統合が自動的にアクティブ化され、Amazon Inspector AWS Security Hub は Security AWS Finding 形式 (ASFF) を使用してすべての検出結果を Security Hub に送信し始めます。

統合先からの検出結果フローの無効化

Amazon Inspector が Security Hub に結果を送信しないようにするには、Security Hub コンソールまたは API と AWS CLI を使用できます。

Security Hub での Amazon Inspector のセキュリティコントロールの表示

Security Hub は、サポートされている AWS およびサードパーティー製品の検出結果を分析し、ルールに対して自動的かつ継続的なセキュリティチェックを実行して、独自の検出結果を生成します。ルールは、標準の要件が満たされているかどうかの判断に役立つセキュリティコントロールによって表されます。

Amazon Inspector はセキュリティコントロールを使用して、Amazon Inspector 機能が有効になっているかどうか、または有効にする必要があるかどうかを確認します。主な機能は以下のとおりです。

- Amazon EC2 スキャン
- Amazon ECR スキャン
- Lambda 標準スキャン
- Lambda コードスキャン

詳細については、「AWS Security Hub ユーザーガイド」の「Amazon Inspector のコントロール」を参照してください。

Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語

Amazon Inspector は、以下にインストールされているソフトウェアアプリケーションをスキャンできます。

- Amazon Elastic Compute Cloud (Amazon EC2) インスタンス

> **Note**
>
> Amazon EC2 インスタンスの場合、Amazon Inspector はエージェントベースのスキャンをサポートするオペレーティングシステムのパッケージの脆弱性をスキャンできます。Amazon Inspector は、ハイブリッドスキャンをサポートするオペレーティングシステムやプログラミング言語のパッケージの脆弱性をスキャンすることもできます。Amazon Inspector はツールチェーンの脆弱性をスキャンしません。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

- Amazon Elastic Container Registry (Amazon ECR) レポジトリに保存されたコンテナイメージ

> **Note**
>
> ECR コンテナイメージの場合、Amazon Inspector ではオペレーティングシステムとプログラミング言語パッケージの脆弱性をスキャンできます。Amazon Inspector は、 のツールチェーンの脆弱性をスキャンしませんRust。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

- AWS Lambda 関数

> **Note**
>
> Lambda 関数の場合、Amazon Inspector はプログラミング言語パッケージの脆弱性とコードの脆弱性をスキャンできます。Amazon Inspector はツールチェーンの脆弱性をスキャンしません。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

Amazon Inspector がリソースをスキャンする際、Amazon Inspector は 50 を超えるデータフィードをソースとして、共通脆弱性識別子 (CVE) の検出結果を生成します。これらのソースの例には、ベンダーセキュリティアドバイザーのデータフィードと脅威インテリジェンスフィード、および National Vulnerability Database (NVD) と MITRE が含まれます。Amazon Inspector は、ソースフィードからの脆弱性データを少なくとも 1 日 1 回更新します。

Amazon Inspector がリソースをスキャンするには、リソースがサポートされているオペレーティングシステムを実行しているか、サポートされているプログラミング言語を使用している必要があります。このセクションのトピックでは、Amazon Inspector がさまざまなリソースとスキャンタイプでサポートしているオペレーティングシステム、プログラミング言語、およびランタイムを示します。また、廃止されたオペレーティングシステムも一覧表示されます。

> **Note**
>
> ベンダーがオペレーティングシステムのサポートを終了した後は、Amazon Inspector では、そのオペレーティングシステムに対して限定的なサポートしか提供できません。

トピック

- サポートされるオペレーティングシステム
- 終了オペレーティングシステム
- サポートされているプログラミング言語
- ランタイムのサポート

サポートされるオペレーティングシステム

このセクションでは、Amazon Inspector がサポートしているオペレーティングシステムを示します。

サポートされているオペレーティングシステム: Amazon EC2スキャン

次の表に、Amazon Inspector が Amazon EC2 インスタンスのスキャンをサポートしているオペレーティングシステムを示します。各オペレーティングシステムのベンダーセキュリティアドバイザリと、エージェントベースのスキャンおよびエージェントレススキャンをサポートしているオペレーティングシステムが明記されています。

エージェントベースのスキャン方式を使用する場合、すべての対象インスタンスで継続的スキャンを実行するように SSM Agent を設定します。Amazon Inspector では、3.2.2086.0 以降のバージョンの SSM Agent を設定することをお勧めしています。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「SSM Agent の使用」を参照してください。

Linux オペレーティングシステムの検出は、デフォルトのパッケージマネージャーリポジトリ (rpm および dpkg) でのみサポートされており、サードパーティーアプリケーション、拡張サポートリポジトリ (RHEL EUS、E4S、AUS、TUS)、およびオプションのリポジトリ (アプリケーションストリーム) は含まれません。Amazon Inspector は実行中のカーネルの脆弱性をスキャンします。などの一部のオペレーティングシステムではUbuntu、アップグレードをアクティブな検出結果に表示するために再起動が必要です。

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ	エージェントレススキャンのサポート	エージェントベースのスキャンのサポート
AlmaLinux	8	ALSA	はい	はい
AlmaLinux	9	ALSA	はい	はい
Amazon Linux (AL2)	AL2	ALAS	はい	はい
Amazon Linux 2023 (AL2023)	AL2023	ALAS	はい	はい
Bottlerocket	1.7.0 以降	GHSA, CVE	いいえ	はい
Debian サーバー (Bullseye)	11	DSA	はい	はい
Debian サーバー (Bookworm)	12	DSA	はい	あり
Fedora	41	CVE	あり	あり
Fedora	42	CVE	あり	はい
OpenSUSE Leap	15.6	CVE	はい	はい

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ	エージェントレススキャンのサポート	エージェントベースのスキャンのサポート
Oracle Linux (Oracle)	8	ELSA	はい	はい
Oracle Linux (Oracle)	9	ELSA	はい	はい
Red Hat Enterprise Linux (RHEL)	8	RHSA	はい	はい
Red Hat Enterprise Linux (RHEL)	9	RHSA	はい	はい
Rocky Linux	8	RLSA	はい	はい
Rocky Linux	9	RLSA	はい	はい
SUSE Linux Enterprise Server (SLES)	15.6	SUSE CVE	はい	はい
Ubuntu (Xenial)	16.04	USN、Ubuntu Pro (esm-infra および esm-apps)	はい	はい
Ubuntu (Bionic)	18.04	USN、Ubuntu Pro (esm-infra および esm-apps)	はい	はい
Ubuntu (Focal)	20.04	USN、Ubuntu Pro (esm-infra および esm-apps)	はい	はい

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ	エージェントレススキャンのサポート	エージェントベースのスキャンのサポート
Ubuntu (Jammy)	22.04	USN、Ubuntu Pro (esm-infra および esm-apps)	はい	あり
Ubuntu (Noble Numbat)	24.04	USN, Ubuntu Pro (esm-infra & esm-apps)	あり	あり
Ubuntu (Oracular Oriole)	24.10	USN	あり	あり
Ubuntu (Plucky Puffin)	25.04	USN	あり	はい
Windows Server	2016	MSKB	いいえ	はい
Windows Server	2019	MSKB	いいえ	はい
Windows Server	2022	MSKB	いいえ	はい
Windows Server	2025	MSKB	いいえ	はい
macOS (Mojave)	10.14	APPLE-SA	いいえ	はい
macOS (Catalina)	10.15	APPLE-SA	いいえ	はい
macOS (Big Sur)	11	APPLE-SA	いいえ	はい
macOS (Monterey)	12	APPLE-SA	いいえ	はい
macOS (Ventura)	13	APPLE-SA	いいえ	はい

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ	エージェントレススキャンのサポート	エージェントベースのスキャンのサポート
macOS (Sonoma)	14	APPLE-SA	いいえ	はい

サポートされているオペレーティングシステム: Amazon Inspector による Amazon ECR スキャン

次の表に、Amazon Inspector が Amazon ECR リポジトリ内のコンテナイメージのスキャンをサポートするオペレーティングシステムを示します。また、各オペレーティングシステムのベンダーセキュリティアドバイザリも示します。

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ
Alpine Linux (Alpine)	3.19	Alpine SecDB
Alpine Linux (Alpine)	3.20	Alpine SecDB
Alpine Linux (Alpine)	3.21	Alpine SecDB
Alpine Linux (Alpine)	3.22	Alpine SecDB
AlmaLinux	8	ALSA
AlmaLinux	9	ALSA
Amazon Linux (AL2)	AL2	ALAS
Amazon Linux 2023 (AL2023)	AL2023	ALAS
BusyBox	—	—
Chainguard	–	CVE
Debian Server (Bullseye)	11	DSA

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ
Debian Server (Bookworm)	12	DSA
Fedora	41	CVE
Fedora	42	CVE
OpenSUSE Leap	15.6	CVE
Oracle Linux (Oracle)	8	ELSA
Oracle Linux (Oracle)	9	ELSA
Photon OS	4	PHSA
Photon OS	5	PHSA
Red Hat Enterprise Linux (RHEL)	8	RHSA
Red Hat Enterprise Linux (RHEL)	9	RHSA
Rocky Linux	8	RLSA
Rocky Linux	9	RLSA
SUSE Linux Enterprise Server (SLES)	15.6	SUSE CVE
Ubuntu (Xenial)	16.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Ubuntu (Bionic)	18.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Ubuntu (Focal)	20.04	USN, Ubuntu Pro (esm-infra & esm-apps)

オペレーティングシステム	バージョン	ベンダーのセキュリティアドバイザリ
Ubuntu (Jammy)	22.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Ubuntu (Noble Numbat)	24.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Ubuntu (Oracular Oriole)	24.10	USN
Ubuntu (Plucky Puffin)	25.04	USN
Wolfi	–	CVE

サポートされているオペレーティングシステム: CIS スキャン

次の表に、Amazon Inspector が CIS スキャンでサポートするオペレーティングシステムを示します。また、各オペレーティングシステムの CIS ベンチマークバージョンも示します。

> **Note**
>
> CIS 標準は x86_64 オペレーティングシステムを対象としています。一部のチェックは評価されないか、ARM ベースのリソースで無効な修復手順が返される場合があります。

オペレーティングシステム	バージョン	CIS ベンチマークバージョン
Amazon Linux 2	AL2	3.0.0
Amazon Linux 2023	AL2023	1.0.0
Red Hat Enterprise Linux (RHEL)	8	3.0.0
Red Hat Enterprise Linux (RHEL)	9	2.0.0

オペレーティングシステム	バージョン	CIS ベンチマークバージョン
Rocky Linux	8	2.0.0
Rocky Linux	9	1.0.0
SUSE Linux Enterprise Server	15	2.0.1
Ubuntu (Bonic)	18.04	2.1.0
Ubuntu (Focal)	20.04	2.0.1
Ubuntu (Jammy)	22.04	1.0.0
Ubuntu (Noble Numbat)	24.04	1.0.0
Windows Server	2016	3.0.0
Windows Server	2019	2.0.0
Windows Server	2022	2.0.0

終了オペレーティングシステム

次の表は、廃止されたオペレーティングシステムと廃止されたオペレーティングシステムの一覧です。

Amazon Inspector は、以下の廃止されたオペレーティングシステムを完全にはサポートしていませんが、引き続き Amazon EC2 インスタンスとそれらを実行している Amazon ECR コンテナイメージをスキャンします。セキュリティのベストプラクティスとして、廃止されたオペレーティングシステムのサポートされているバージョンに移行することをお勧めします。Amazon Inspector が終了したオペレーティングシステムについて生成した検出結果については、情報提供のみを目的としてご利用ください。

ベンダーポリシーに従って、次のオペレーティングシステムはパッチ更新を受け取らなくなります。廃止されたオペレーティングシステムに対しては、新しいセキュリティアドバイザリがリリースされない場合があります。ベンダーは、標準サポートが終了したオペレーティングシステムについては、既存のセキュリティアドバイザリと検出情報をフィードから削除する場合があります。その結果、Amazon Inspector は既知の CVE に対する検出結果の生成を停止できます。

終了したオペレーティングシステム: Amazon EC2 スキャン

オペレーティングシステム	バージョン	終了
Amazon Linux (AL1)	2012	2021 年 12 月 31 日
CentOS Linux (CentOS)	7	2024 年 6 月 30 日
CentOS Linux (CentOS)	8	2021 年 12 月 31 日
Debian サーバー (Jessie)	8	2020 年 6 月 30 日
Debian Server (Stretch)	9	2022 年 6 月 30 日
Debian サーバー (Buster)	10	2024 年 6 月 30 日
Fedora	33	2021 年 11 月 30 日
Fedora	34	2022 年 6 月 7 日
Fedora	35	2022 年 12 月 13 日
Fedora	36	2023 年 5 月 16 日
Fedora	37	2023 年 12 月 15 日
Fedora	38	2024 年 5 月 21 日
Fedora	39	2024 年 11 月 26 日
Fedora	40	2025 年 5 月 13 日
OpenSUSE Leap	15.2	2021 年 12 月 1 日
OpenSUSE Leap	15.3	2022 年 12 月 1 日
OpenSUSE Leap	15.4	2023 年 12 月 7 日
OpenSUSE Leap	15.5	December 31, 2024
Oracle Linux (Oracle)	6	2021 年 3 月 1 日

オペレーティングシステム	バージョン	終了
Oracle Linux (Oracle)	7	2024 年 12 月 31 日
Red Hat Enterprise Linux (RHEL)	6	2020 年 11 月 30 日
Red Hat Enterprise Linux (RHEL)	7	2024 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12	2016 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12.1	2017 年 5 月 31 日
SUSE Linux Enterprise Server (SLES)	12.2	2018 年 3 月 31 日
SUSE Linux Enterprise Server (SLES)	12.3	2019 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12.4	2020 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12.5	2024 年 10 月 31 日
SUSE Linux Enterprise Server (SLES)	15	2019 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.1	2021 年 1 月 31 日
SUSE Linux Enterprise Server (SLES)	15.2	2021 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.3	2022 年 12 月 31 日

オペレーティングシステム	バージョン	終了
SUSE Linux Enterprise Server (SLES)	15.4	2023 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.5	2024 年 12 月 31 日
Ubuntu (Trusty)	12.04	2017 年 4 月 28 日
Ubuntu (Trusty)	14.04	2024 年 4 月 1 日
Ubuntu (Groovy)	20.10	2021 年 7 月 22 日
Ubuntu (Hirsute)	21.04	2022 年 1 月 20 日
Ubuntu (Impish)	21.10	2022 年 7 月 31 日
Ubuntu (Kinetic)	22.10	July 20, 2023
Ubuntu (Lunar Lobster)	23.04	January 25, 2024
Ubuntu (Mantic Minotaur)	23.10	2024 年 7 月 11 日
Windows Server	2012	2023 年 10 月 10 日
Windows Server	2012 R2	2023 年 10 月 10 日

終了したオペレーティングシステム: Amazon ECR スキャン

オペレーティングシステム	バージョン	終了
Alpine Linux (Alpine)	3.2	2017 年 5 月 1 日
Alpine Linux (Alpine)	3.3	2017 年 11 月 1 日
Alpine Linux (Alpine)	3.4	2018 年 5 月 1 日
Alpine Linux (Alpine)	3.5	2018 年 11 月 1 日

オペレーティングシステム	バージョン	終了
Alpine Linux (Alpine)	3.6	2019 年 5 月 1 日
Alpine Linux (Alpine)	37	2019 年 11 月 1 日
Alpine Linux (Alpine)	3.8	2020 年 5 月 1 日
Alpine Linux (Alpine)	3.9	2020 年 11 月 1 日
Alpine Linux (Alpine)	3.10	2021 年 5 月 1 日
Alpine Linux (Alpine)	3.11	2021 年 11 月 1 日
Alpine Linux (Alpine)	3.12	2022 年 5 月 1 日
Alpine Linux (Alpine)	3.13	2022 年 11 月 1 日
Alpine Linux (Alpine)	3.14	May 1, 2023
Alpine Linux (Alpine)	3.15	November 1, 2023
Alpine Linux (Alpine)	3.16	May 23, 2024
Alpine Linux (Alpine)	3.17	November 22, 2024
Alpine Linux (Alpine)	3.18	May 09 2025
Amazon Linux (AL1)	2012	2021 年 12 月 31 日
CentOS Linux (CentOS)	7	2024 年 6 月 30 日
CentOS Linux (CentOS)	8	2021 年 12 月 31 日
Debian サーバー (Jessie)	8	2020 年 6 月 30 日
Debian Server (Stretch)	9	2022 年 6 月 30 日
Debian サーバー (Buster)	10	2024 年 6 月 30 日
Fedora	33	2021 年 11 月 30 日

オペレーティングシステム	バージョン	終了
Fedora	34	2022 年 6 月 7 日
Fedora	35	2022 年 12 月 13 日
Fedora	36	2023 年 5 月 16 日
Fedora	37	2023 年 12 月 15 日
Fedora	38	2024 年 5 月 21 日
Fedora	39	2024 年 11 月 26 日
Fedora	40	2025 年 5 月 13 日
OpenSUSE Leap	15.2	2021 年 12 月 1 日
OpenSUSE Leap	15.3	2022 年 12 月 1 日
OpenSUSE Leap	15.4	December 7, 2023
OpenSUSE Leap	15.5	December 31, 2024
Oracle Linux (Oracle)	6	2021 年 3 月 1 日
Oracle Linux (Oracle)	7	2024 年 12 月 31 日
Photon OS	2	2021 年 12 月 2 日
Photon OS	3	2024 年 3 月 1 日
Red Hat Enterprise Linux (RHEL)	6	2020 年 6 月 30 日
Red Hat Enterprise Linux (RHEL)	7	2024 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12	2016 年 6 月 30 日

オペレーティングシステム	バージョン	終了
SUSE Linux Enterprise Server (SLES)	12.1	2017 年 5 月 31 日
SUSE Linux Enterprise Server (SLES)	12.2	2018 年 3 月 31 日
SUSE Linux Enterprise Server (SLES)	12.3	2019 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12.4	2020 年 6 月 30 日
SUSE Linux Enterprise Server (SLES)	12.5	2024 年 10 月 31 日
SUSE Linux Enterprise Server (SLES)	15	2019 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.1	2021 年 1 月 31 日
SUSE Linux Enterprise Server (SLES)	15.2	2021 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.3	2022 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.4	2023 年 12 月 31 日
SUSE Linux Enterprise Server (SLES)	15.5	2024 年 12 月 31 日
Ubuntu (Trusty)	12.04	2017 年 4 月 28 日
Ubuntu (Trusty)	14.04	2024 年 4 月 1 日
Ubuntu (Groovy)	20.10	2021 年 7 月 22 日

オペレーティングシステム	バージョン	終了
Ubuntu (Hirsute)	21.04	2022 年 1 月 20 日
Ubuntu (Impish)	21.10	2022 年 7 月 31 日
Ubuntu (Kinetic)	22.10	July 20, 2023
Ubuntu (Lunar Lobster)	23.04	January 25, 2024
Ubuntu (Mantic Minotaur)	23.10	2024 年 7 月 11 日

サポートされているプログラミング言語

このセクションでは、Amazon Inspector がサポートするプログラミング言語を一覧表示します。

サポートされているプログラミング言語: Amazon EC2 エージェントレススキャン

Amazon Inspector は現在、対象となる Amazon EC2 インスタンスでエージェントレススキャンを実行するときに、次のプログラミング言語をサポートしています。詳細については、「エージェントレススキャン」を参照してください。

> **Note**
>
> Amazon Inspector は、 Goおよび のツールチェーンの脆弱性をスキャンしませんRust。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

- C#
- Go
- Java
- JavaScript
- PHP
- Python

- Ruby
- Rust

サポートされているプログラミング言語: Amazon EC2 詳細検査

現在、Amazon Inspector は、Amazon EC2 Linux インスタンスでの詳細検査スキャンの実行時に、次のプログラミング言語をサポートしています。詳細については、「Linux ベースの Amazon EC2 インスタンスの Amazon Inspector の詳細な検査Amazon EC2」を参照してください。

- Java (.ear、.jar、.par、.war アーカイブ形式)
- JavaScript
- Python

Amazon Inspector は Systems Manager Distributor を使用して、Amazon EC2 インスタンスの詳細検査用のプラグインをデプロイします。

> **Note**
>
> Deep inspection は Bottlerocket オペレーティングシステムではサポートされていません。

詳細検査スキャンを実行するには、Systems Manager Distributor と Amazon Inspectorが、Amazon EC2 インスタンスオペレーティングシステムをサポートしている必要があります。Systems Manager Distributor でサポートされるオペレーティングシステムの詳細については、「Systems Manager ユーザーガイド」の「サポートされているパッケージのプラットフォームとアーキテクチャ」を参照してください。

サポートされているプログラミング言語: Amazon ECR スキャン

現在、Amazon Inspector は、Amazon ECR リポジトリでのコンテナイメージのスキャン時に、次のプログラミング言語をサポートしています。

> **Note**
>
> Amazon Inspector は、 のツールチェーンの脆弱性をスキャンしませんRust。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

- C#
- Go
- Go ツールチェーン
- Java
- Java JDK
- JavaScript
- PHP
- Python
- Ruby
- Rust

ランタイムのサポート

このセクションでは、Amazon Inspector がサポートしているランタイムを示します。

サポートされているランタイム: Amazon Inspector Lambda 標準スキャン

現在、Amazon Inspector Lambda 標準スキャンは、サードパーティー製ソフトウェアパッケージの脆弱性についての Lambda 関数のスキャン時に、次のプログラミング言語のランタイムをサポートしています。

> **Note**
>
> Amazon Inspector は、 のツールチェーンの脆弱性をスキャンしませんRust。アプリケーションの構築に使用されるプログラミング言語コンパイラのバージョンは、これらの脆弱性をもたらします。

- Go
  - go1.x
- Java
  - java8
  - java8.al2
  - java11

  - java17
  - java21
- .NET
  - .NET 6
  - .NET 8
- Node.js
  - nodejs12.x
  - nodejs14.x
  - nodejs16.x
  - nodejs18.x
  - nodejs20.x
  - nodejs22.x
- Python
  - python3.7
  - python3.8
  - python3.9
  - python3.10
  - python3.11
  - python3.12
  - python3.13
- Ruby
  - ruby2.7
  - ruby3.2
  - ruby3.3
- Custom runtimes
  - AL2
  - AL2023

サポートされているランタイム: Amazon Inspector Lambda コードスキャン

現在、Amazon Inspector Lambda コードスキャンは、Lambda 関数のスキャンによるコード脆弱性の検出時に、次のプログラミング言語のランタイムをサポートしています。

- Java
  - java8
  - java8.al2
  - java11
  - java17
- .NET
  - .NET 6
  - .NET 8
- Node.js
  - nodejs12.x
  - nodejs14.x
  - nodejs16.x
  - nodejs18.x
  - nodejs20.x
- Python
  - python3.7
  - python3.8
  - python3.9
  - python3.10
  - python3.11
  - python3.12
- Ruby
  - ruby2.7
  - ruby3.2
  - ruby3.3

Amazon Inspector の非アクティブ化

Amazon Inspector は、Amazon Inspector コンソールで、または Amazon Inspector API を使用して非アクティブ化できます。アカウントのすべてのスキャンタイプを非アクティブ化すると、そのアカウントに対して Amazon Inspector は自動的に非アクティブ化されます。

アカウントの Amazon Inspector を非アクティブ化すると、そのアカウントに対してすべてのスキャンタイプが非アクティブ化されます。さらに、そのアカウントの Amazon Inspector スキャン設定は、フィルター、抑制ルール、および検出結果を含めて、すべて削除されます。

Amazon Inspector Amazon EC2 スキャンを非アクティブ化すると、Amazon Inspector は以下の SSM 関連付けを削除します:

- `InspectorDistributor-do-not-delete`
- `InspectorInventoryCollection-do-not-delete`
- `InvokeInspectorSsmPlugin-do-not-delete`。さらに、この関連付けを通じてインストールされた Amazon Inspector SSM プラグインは、すべての Windows ホストから削除されます。詳細については、「Windows EC2 インスタンスのスキャン」を参照してください。

> **Note**
>
> Amazon Inspector を非アクティブ化すると、サービス料金は発生しなくなります。ただし、Amazon Inspector はいつでも再アクティブ化できます。

さまざまなリソースのスキャンタイプを非アクティブ化する方法については、「スキャンタイプの非アクティブ化」を参照してください。

前提条件

アカウントタイプに応じて、以下の点を考慮してください。

- アカウントがスタンドアロンの Amazon Inspector アカウントの場合は、いつでも Amazon Inspector を非アクティブ化できます。
- アカウントがマルチアカウント環境のメンバーアカウントの場合、Amazon Inspector を非アクティブ化することはできません。Amazon Inspector を非アクティブ化するには、組織の委任管理者に連絡する必要があります。

- 組織の委任管理者である場合は、Amazon Inspector を非アクティブ化する前に、すべてのメンバーアカウントの関連付けを解除する必要があります。

> **Note**
>
> 委任管理者として Amazon Inspector を非アクティブ化すると、組織の自動アクティブ化機能は非アクティブになります。

Amazon Inspector を非アクティブ化する

> **Note**
>
> Amazon Inspector を非アクティブ化にする前に、検出結果のエクスポートを検討してください。

Console

Amazon Inspector を非アクティブ化するには

1. 認証情報を使用してサインインし、Amazon Inspector コンソール (https://console.aws.amazon.com/inspector/v2/home) を開きます。
2. ページの右上隅にある AWS リージョン セレクターを使用して、Amazon Inspector を非アクティブ化するリージョンを選択します。
3. ナビゲーションペインで [全般設定] を選択します。
4. [Inspector を非アクティブ化] を選択します。
5. 確認を求められたら、テキストボックスに「deactivate」と入力し、[Inspector を非アクティブ化] を選択します。
6. (推奨) Amazon Inspector を非アクティブ化するリージョンごとに、これらの手順を繰り返します。

API

無効化 APL オペレーションを実行します。リクエストには、非アクティブ化するアカウント ID を指定し、すべてのスキャンを非アクティブ化する `resourceTypes` の `EC2, ECR, LAMBDA` を指定すると、アカウントが非アクティブ化されます。

Amazon Inspector のクォータ

このセクションでは、 AWS リージョンあたりの Amazon Inspector クォータ数を示します。

リソース	デフォルト	コメント
メンバーアカウント	10,000	Amazon Inspector の委任管理者アカウントに関連付けられたメンバーアカウントの最大数。制限はクォ AWS Organizationsータに基づいています。
抑制ルール	500	リージョンごとの AWS アカウントあたりの保存済み抑制ルールの最大数。クォータの引き上げはリクエストできません。
Amazon EC2 ネットワークの検出結果	10,000	AWS アカウントあたりの Amazon EC2 ネットワーク検出結果の最大数。クォータの引き上げはリクエストできません。
CIS スキャン設定	500	CIS スキャン設定の最大数。クォータの引き上げはリクエストできません。

Amazon Inspector Classic に関連するクォータのリストについては、「AWS 全般のリファレンス」の「Amazon Inspector Classic サービスクォータ」を参照してください。に関連付けられているクォータのリストについては AWS Organizations、「」のAWS Organizations 「サービスクォータ」を参照してくださいAWS 全般のリファレンス。

リージョンとエンドポイント

このトピックには、Amazon Inspector および Amazon Inspector スキャンのエンドポイントを示すテーブルが含まれています。また、Amazon Inspector 機能 AWS リージョン をサポートする を示すテーブルも含まれています。 AWS リージョン Amazon Inspector が利用可能な を表示するには、のAmazon Inspector エンドポイントとクォータ」を参照してくださいAmazon Web Services 全般のリファレンス。

Amazon Inspector のサービスエンドポイント

次の表は、Amazon Inspector のサービスエンドポイントを示しています。Amazon Inspector エンドポイントの命名規則は ですinspector2.*Region*.amazonaws.com。

リージョン名	リージョン	エンドポイント	プロトコル
カナダ西部 (カルガリー)	ca-west-1	inspector2.ca-west-1.amazonaws.com inspector2.ca-west-1.api.aws.com	HTTPS
米国東部 (バージニア北部)	us-east-1	inspector2.us-east-1.amazonaws.com inspector2.us-east-1.api.aws.com inspector2-fips.us-east-1.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	inspector2.us-east-2.amazonaws.com inspector2.us-east-2.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
		inspector2-fips.us-east-2.amazonaws.com	
米国西部 (北カリフォルニア)	us-west-1	inspector2.us-west-1.amazonaws.com inspector2.us-west-1.api.aws.com inspector2-fips.us-west-1.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	inspector2.us-west-2.amazonaws.com inspector2.us-west-2.api.aws.com inspector2-fips.us-west-2.amazonaws.com	HTTPS
アフリカ (ケープタウン)	af-south-1	inspector2.af-south-1.amazonaws.com inspector2.af-south-1.api.aws.com	HTTPS
アジアパシフィック (香港)	ap-east-1	inspector2.ap-east-1.amazonaws.com inspector2.ap-east-1.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (シドニー)	ap-southeast-2	inspector2.ap-southeast-2.amazonaws.com inspector2.ap-southeast-2.api.aws.com	HTTPS
アジアパシフィック (ジャカルタ)	ap-southeast-3	inspector2.ap-southeast-3.amazonaws.com inspector2.ap-southeast-3.api.aws.com	HTTPS
アジアパシフィック (メルボルン)	ap-southeast-4	inspector2.ap-southeast-4.amazonaws.com inspector2.ap-southeast-4.api.aws.com	HTTPS
アジアパシフィック (マレーシア)	ap-southeast-5	inspector2.ap-southeast-5.amazonaws.com inspector2.ap-southeast-5.api.aws.com	HTTPS
アジアパシフィック (タイ)	ap-southeast-7	inspector2.ap-southeast-7.amazonaws.com inspector2.ap-southeast-7.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (ムンバイ)	ap-south-1	inspector2.ap-south-1.amazonaws.com inspector2.ap-south-1.api.aws.com	HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	inspector2.ap-south-2.amazonaws.com inspector2.ap-south-2.api.aws.com	HTTPS
アジアパシフィック (大阪)	ap-northeast-3	inspector2.ap-northeast-3.amazonaws.com inspector2.ap-northeast-3.api.aws.com	HTTPS
アジアパシフィック (ソウル)	ap-northeast-2	inspector2.ap-northeast-2.amazonaws.com inspector2.ap-northeast-2.api.aws.com	HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	inspector2.ap-southeast-1.amazonaws.com inspector2.ap-southeast-1.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (シドニー)	ap-southeast-2	inspector2.ap-southeast-2.amazonaws.com inspector2.ap-southeast-2.api.aws.com	HTTPS
アジアパシフィック (東京)	ap-northeast-1	inspector2.ap-northeast-1.amazonaws.com inspector2.ap-northeast-1.api.aws.com	HTTPS
カナダ (中部)	ca-central-1	inspector2.ca-central-1.amazonaws.com inspector2.ca-central-1.api.aws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	inspector2.eu-central-1.amazonaws.com inspector2.eu-central-1.api.aws.com	HTTPS
欧州 (アイルランド)	eu-west-1	inspector2.eu-west-1.amazonaws.com inspector2.eu-west-1.api.aws.com	HTTPS
欧州 (ロンドン)	eu-west-2	inspector2.eu-west-2.amazonaws.com inspector2.eu-west-2.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
ヨーロッパ (ミラノ)	eu-south-1	inspector2.eu-south-1.amazonaws.com inspector2.eu-south-1.api.aws.com	HTTPS
欧州 (スペイン)	region-eu-south-2	inspector2.region-eu-south-2.amazonaws.com inspector2.region-eu-south-2.api.aws.com	HTTPS
欧州 (パリ)	eu-west-3	inspector2.eu-west-3.amazonaws.com inspector2.eu-west-3.api.aws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	inspector2.eu-north-1.amazonaws.com inspector2.eu-north-1.api.aws.com	HTTPS
欧州 (チューリッヒ)	eu-central-2	inspector2.eu-central-2.amazonaws.com inspector2.eu-central-2.api.aws.com	HTTPS
イスラエル (テルアビブ)	il-central-1	inspector2.il-central-1.amazonaws.com inspector2.il-central-1.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
中東 (アラブ首長国連邦)	me-central-1	inspector2.me-central-1.amazonaws.com inspector2.me-central-1.api.aws.com	HTTPS
中東 (バーレーン)	me-south-1	inspector2.me-south-1.amazonaws.com inspector2.me-south-1.api.aws.com	HTTPS
メキシコ (中部)	mx-central-1	inspector2.mx-central-1.amazonaws.com inspector2.mx-central-1.api.aws.com	HTTPS
南米 (サンパウロ)	sa-east-1	inspector2.sa-east-1.amazonaws.com inspector2.sa-east-1.api.aws.com	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	inspector2.us-gov-east-1.amazonaws.com inspector2.us-gov-east-1.api.aws.com inspector2-fips.us-gov-east-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
AWS GovCloud (米国西部)	us-gov-west-1	inspector2.us-gov-west-1.amazonaws.com inspector2.us-gov-west-1.api.aws.com inspector2-fips.us-gov-west-1.amazonaws.com	HTTPS

Amazon Inspector スキャン API のエンドポイント

次の表は、Amazon Inspector スキャン API を呼び出すときに使用できるリージョンエンドポイントを示しています。API を使用する場合は、エンドポイントと、現在認証されているリージョンに対応する AWS リージョンを指定する必要があります。

Amazon Inspector スキャンのエンドポイントの命名規則は `inspector-scan.`*`region`*`.amazonaws.com` です。例えば、`us-west-2` で認証されている場合は、エンドポイント `inspector-scan.us-west-2.amazonaws.com` を使用して `inspector-scan` API を呼び出します。

リージョン名	リージョン	エンドポイント	プロトコル
カナダ西部 (カルガリー)	ca-west-1	inspector-scan.ca-west-1.amazonaws.com inspector-scan.ca-west-1.api.aws.com	HTTPS
米国東部 (オハイオ)	us-east-2	inspector-scan.us-east-2.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
		inspector-scan.us-east-2.api.aws.com inspector-scan-fips.us-east-2.amazonaws.com	
米国東部 (バージニア北部)	us-east-1	inspector-scan.us-east-1.amazonaws.com inspector-scan.us-east-1.api.aws.com inspector-scan-fips.us-east-1.amazonaws.com	HTTPS
米国西部 (北カリフォルニア)	us-west-1	inspector-scan.us-west-1.amazonaws.com inspector-scan.us-west-1.api.aws.com inspector-scan-fips.us-west-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
米国西部 (オレゴン)	us-west-2	inspector-scan.us-west-2.amazonaws.com inspector-scan.us-west-2.api.aws.com inspector-scan-fips.us-west-2.amazonaws.com	HTTPS
アフリカ (ケープタウン)	af-south-1	inspector-scan.af-south-1.amazonaws.com inspector-scan.af-south-1.api.aws.com	HTTPS
アジアパシフィック (香港)	ap-east-1	inspector-scan.ap-east-1.amazonaws.com inspector-scan.ap-east-1.api.aws.com	HTTPS
アジアパシフィック (ジャカルタ)	ap-southeast-3	inspector-scan.ap-southeast-3.amazonaws.com inspector-scan.ap-southeast-3.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (ムンバイ)	ap-south-1	inspector-scan.ap-south-1.amazonaws.com inspector-scan.ap-south-1.api.aws.com	HTTPS
アジアパシフィック (大阪)	ap-northeast-3	inspector-scan.ap-northeast-3.amazonaws.com inspector-scan.ap-northeast-3.api.aws.com	HTTPS
アジアパシフィック (ソウル)	ap-northeast-2	inspector-scan.ap-northeast-2.amazonaws.com inspector-scan.ap-northeast-2.api.aws.com	HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	inspector-scan.ap-south-2.amazonaws.com inspector-scan.ap-south-2.api.aws.com	HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	inspector-scan.ap-southeast-1.amazonaws.com inspector-scan.ap-southeast-1.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (シドニー)	ap-southeast-2	inspector-scan.ap-southeast-2.amazonaws.com inspector-scan.ap-southeast-2.api.aws.com	HTTPS
アジアパシフィック (メルボルン)	ap-southeast-4	inspector-scan.ap-southeast-4.amazonaws.com inspector-scan.ap-southeast-4.api.aws.com	HTTPS
アジアパシフィック (マレーシア)	ap-southeast-5	inspector-scan.ap-southeast-5.amazonaws.com inspector-scan.ap-southeast-5.api.aws.com	HTTPS
アジアパシフィック (タイ)	ap-southeast-7	inspector-scan.ap-southeast-7.amazonaws.com inspector-scan.ap-southeast-7.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (東京)	ap-northeast-1	inspector-scan.ap-northeast-1.amazonaws.com inspector-scan.ap-northeast-1.api.aws.com	HTTPS
カナダ (中部)	ca-central-1	inspector-scan.ca-central-1.amazonaws.com inspector-scan.ca-central-1.api.aws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	inspector-scan.eu-central-1.amazonaws.com inspector-scan.eu-central-1.api.aws.com	HTTPS
欧州 (アイルランド)	eu-west-1	inspector-scan.eu-west-1.amazonaws.com inspector-scan.eu-west-1.api.aws.com	HTTPS
欧州 (ロンドン)	eu-west-2	inspector-scan.eu-west-2.amazonaws.com inspector-scan.eu-west-2.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
ヨーロッパ (ミラノ)	eu-south-1	inspector-scan.eu-south-1.amazonaws.com inspector-scan.eu-south-1.api.aws.com	HTTPS
欧州 (スペイン)	eu-south-2	inspector-scan.eu-south-2.amazonaws.com inspector-scan.eu-south-2.api.aws.com	HTTPS
欧州 (パリ)	eu-west-3	inspector-scan.eu-west-3.amazonaws.com inspector-scan.eu-west-3.api.aws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	inspector-scan.eu-north-1.amazonaws.com inspector-scan.eu-north-1.api.aws.com	HTTPS
欧州 (チューリッヒ)	eu-central-2	inspector-scan.eu-central-2.amazonaws.com inspector-scan.eu-central-2.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
中東 (アラブ首長国連邦)	me-central-1	inspector-scan.me-central-1.amazonaws.com inspector-scan.me-central-1.api.aws.com	HTTPS
中東 (バーレーン)	me-south-1	inspector-scan.me-south-1.amazonaws.com inspector-scan.me-south-1.api.aws.com	HTTPS
メキシコ (中部)	mx-central-1	inspector-scan.mx-central-1.amazonaws.com inspector-scan.mx-central-1.api.aws.com	HTTPS
イスラエル (テルアビブ)	il-central-1	inspector-scan.il-central-1.amazonaws.com inspector-scan.il-cetral-1.api.aws.com	HTTPS
南米 (サンパウロ)	sa-east-1	inspector-scan.sa-east-1.amazonaws.com inspector-scan.sa-east-1.api.aws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
AWS GovCloud (米国東部)	us-gov-east-1	inspector-scan.us-gov-east-1.amazonaws.com inspector-scan.us-gov-east-1.api.aws.com inspector-scan-fips.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	inspector-scan.us-gov-west-1.amazonaws.com inspector-scan.us-gov-west-1.api.aws.com inspector-scan-fips.us-gov-west-1.amazonaws.com	HTTPS

リージョン固有機能の可用性

このセクションでは、 AWS リージョンで利用可能な Amazon Inspector の機能について説明します。

Amazon EC2 リージョンのエージェントレス EC2 スキャン

次の表は、Amazon EC2 のエージェントレススキャン AWS リージョン が現在利用可能な を示しています。

リージョン名	リージョンコード
米国東部 (バージニア北部)	us-east-1
米国東部 (オハイオ)	us-east-2
米国西部 (北カリフォルニア)	us-west-1
米国西部 (オレゴン)	us-west-2
アフリカ (ケープタウン)	af-south-1
アジアパシフィック (香港)	ap-east-1
アジアパシフィック (東京)	ap-northeast-1
アジアパシフィック (ソウル)	ap-northeast-2
アジアパシフィック (大阪)	ap-northeast-3
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 (ハイデラバード)	ap-south-2
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 (ジャカルタ)	ap-southeast-3
アジアパシフィック (メルボルン)	ap-southeast-4
アジアパシフィック (マレーシア)	ap-southeast-5
アジアパシフィック (タイ)	ap-southeast-7
カナダ (中部)	ca-central-1
カナダ西部 (カルガリー)	ca-west-1
欧州 (ストックホルム)	eu-north-1

リージョン名	リージョンコード
欧州 (フランクフルト)	eu-central-1
欧州 (チューリッヒ)	eu-central-2
欧州 (アイルランド)	eu-west-1
欧州 (ロンドン)	eu-west-2
欧州 (パリ)	eu-west-3
欧州 (ミラノ)	eu-south-1
欧州 (スペイン)	eu-south-2
イスラエル (テルアビブ)	il-central-1
中東 (アラブ首長国連邦)	me-central-1
中東 (バーレーン)	me-south-1
メキシコ (中部)	mx-central-1
南米 (サンパウロ)	sa-east-1
AWS GovCloud (米国東部)	us-gov-east-1
AWS GovCloud (米国西部)	us-gov-west-1

Lambda コードスキャンリージョン

次の表は、Lambda コードスキャン AWS リージョン が現在利用可能な を示しています。

リージョン名	リージョンコード
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
米国東部 (オハイオ)	us-east-2

リージョン名	リージョンコード
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
欧州 (ロンドン)	eu-west-2
欧州 (ストックホルム)	eu-north-1
アジアパシフィック (シンガポール)	ap-southeast-1

> **Important**
>
> Lambda コードスキャンが利用できない で Amazon Inspector Enable API を使用して AWS リージョン Lambda コードスキャンを有効にしようとすると、次のアクセス拒否エラーが表示されます。
>
> ```
> An error occurred (AccessDeniedException) when calling the Enable operation:
> Lambda code scanning is not supported in unsupported-AWS #####
> ```

Amazon Inspector コードセキュリティリージョン

次の表は、Amazon Inspector Code Security AWS リージョン が現在利用可能な を示しています。

リージョン名	リージョンコード
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
米国東部 (オハイオ)	us-east-2

リージョン名	リージョンコード
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
欧州 (ロンドン)	eu-west-2
欧州 (ストックホルム)	eu-north-1
アジアパシフィック (シンガポール)	ap-southeast-1

AWS GovCloud (US) リージョン

最新情報については、「AWS GovCloud (US) ユーザーガイド」の「Amazon Inspector」を参照してください。

ドキュメント履歴

次の表に、2021 年 11 月以降の「Amazon Inspector ユーザーガイド」の各リリースにおける重要な変更点を示します。ドキュメントの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

変更	説明	日付
新しいポリシー	Amazon Inspector は、Amazon Inspector へのフルアクセスと他の関連サービスへのアクセスを提供する新しい管理ポリシーを追加します。詳細については、AWS Amazon Inspector の マネージドポリシー」を参照してください。	2025 年 7 月 3 日
更新された機能	Amazon Inspector が新しい で利用可能になりました AWS リージョン。詳細については、「 リージョンとエンドポイント」を参照してください。	2025 年 7 月 1 日
更新された機能	Amazon Inspector は、クローズされた検出結果の保持期間を更新します。Amazon Inspector は、関連付けられたリソースが削除された場合、終了した場合、またはスキャンの対象ではなくなった場合、3 日後に検出結果を削除します。詳細については、Amazon Inspector の検出	2025 年 6 月 25 日

	結果について」を参照してください。	
更新された機能	Amazon Inspector は、Amazon EC2 スキャンと Amazon ECR スキャン用にサポートされているオペレーティングシステムを更新します。Amazon EC2 スキャンでFedoraバージョン 42 とUbuntuバージョン 25.04 がサポートされるようになりました。Amazon ECR スキャンで、Alpineバージョン 3.22、Fedoraバージョン 42、およびUbuntuバージョン 25.04 がサポートされるようになりました。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2025 年 6 月 18 日
新機能	Amazon Inspector は、ファーストパーティアプリケーションのソースコード、サードパーティーアプリケーションの依存関係、Infrastructure as Code の脆弱性をスキャンするようになりました。詳細については、Amazon Inspector Code Security」を参照してください。	2025 年 6 月 17 日

プラグインの更新	Amazon Inspector は、Amazon Inspector SSM プラグインが CVE-2025-0913 および の脆弱性検出結果を生成するシナリオを認識しているCVE-2025-4673 。Amazon Inspector SSM プラグインがこれらの CVEs の影響を受けていないことが確認され、この検出を解決するための更新がデプロイされました。	2025 年 6 月 13 日
新機能	Amazon Inspector は、アクティブに使用されているコンテナイメージと、クラスターでコンテナイメージが最後に使用された日時を表示できるようになりました。詳細については、「実行中のコンテナへのコンテナイメージのマッピング」を参照してください。	2025 年 5 月 16 日
新機能	Amazon Inspector は、アクティブに使用されているコンテナイメージと、クラスターでコンテナイメージが最後に使用された日時を表示できるようになりました。詳細については、「実行中のコンテナへのコンテナイメージのマッピング」を参照してください。	2025 年 5 月 16 日

サポートされているオペレーティングシステムの更新	Amazon Inspector のサポートの追加 BusyBox 詳細については、Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2025 年 5 月 13 日
更新されたポリシー	Amazon Inspector は、 という名前のサービスにリンクされたロールに新しいアクセス許可を追加しますAmazonInspector2ServiceRolePolicy 。このアクセス許可により、IP アドレスとインターネットゲートウェイを記述できます。詳細については、AWS Amazon Inspector の マネージドポリシー」を参照してください。	2025 年 4 月 29 日
プラグインの更新	Amazon Inspector は、Amazon Inspector SSM プラグインが の脆弱性検出結果を生成するシナリオを認識していますCVE-2025-22871 。Amazon Inspector SSM プラグインがこれらの CVEs の影響を受けていないことが確認され、この検出を解決するための更新がデプロイされました。	2025 年 4 月 21 日

プラグインの更新	Amazon Inspector は、Amazon Inspector SSM プラグインが CVE-2020-8911 、CVE-2020-8912 、および の脆弱性検出結果を生成するシナリオを認識していますCVE-2024-45337 。Amazon Inspector がこれらの CVEs、この検出を解決するために更新がデプロイされました。	2025 年 4 月 18 日
Amazon Inspector SBOM Generator の章の更新	Amazon Inspector は Amazon Inspector SBOM Generator のバージョンを更新します。詳細については、Amazon Inspector SBOM Generator の以前のバージョン」を参照してください。	2025 年 4 月 16 日
Amazon Inspector SBOM Generator の章の更新	Amazon Inspector はAmazon Inspector SBOM Generator の章に新しいトピックを追加します。このトピックでは、がソフトウェア部品表のライセンス情報Sbomgenを追跡する方法について説明します。詳細については、Amazon Inspector SBOM Generator ライセンスコレクション」を参照してください。	2025 年 4 月 16 日

マネージドポリシーの更新	Amazon Inspector は、Amazon ECS および Amazon EKS アクションへの読み取り専用アクセスを許可するアクセス許可を追加します。詳細については、Amazon Inspector のサービスにリンクされたロールのアクセス許可」を参照してください。	2025 年 3 月 25 日
サポートされているオペレーティングシステムの更新	Amazon Inspector は、Amazon EC2 と Amazon ECR のスキャンSUSE Linux Enterprise Server 12.5の一部として をサポートしなくなりました。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2025 年 3 月 21 日
サポートされているオペレーティングシステムの更新	Amazon Inspector は、Amazon ECR スキャンWolfiに Chainguardと のサポートを追加します。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2025 年 3 月 21 日
目次の更新	Amazon Inspector にAmazon Inspector リソースのタグ付けに関する章が追加されました。詳細については、Amazon Inspector リソースのタグ付け」を参照してください。	2025 年 2 月 25 日

目次の更新	Amazon Inspector はAmazon Inspector SBOM Generator の章に新しいトピックを追加します。詳細については、Amazon Inspector SBOM Generator comprehensive operating system collection」を参照してください。	2025 年 1月 28 日
更新された機能	Amazon Inspector は、Lambda 標準スキャンでサポートされているランタイムのリストpython3.13にnodejs202.xと を追加します。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2025 年 1 月 24 日
更新された機能	Amazon Inspector はOracle Linux (Oracle)、Amazon EC2 と Amazon ECR でサポートされているオペレーティングシステムのリストから Linux Enterprise Server (SLES) 7 と SUSE 15.5 を削除します。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2024 年 12 月 31 日

更新された機能	Amazon Inspector Ubuntu は、Amazon EC2 と Amazon ECR でサポートされているオペレーティングシステムのリストに 24.10 を追加します。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2024 年 12 月 12 日
目次の更新	Amazon Inspector は、Amazon Inspector SBOM Generator の章に新しいトピックを追加します。詳細については、Amazon Inspector SBOM Generator」を参照してください。	2024 年 12 月 9 日
更新された機能	Amazon Inspector は`amazon:inspector:sbom_generator` テーブルを更新して名前空間を追加および削除します。詳細については、Amazon Inspector での CycloneDX 名前空間の使用」を参照してください。	2024 年 12 月 9 日
更新された機能	Amazon Inspector は、CodePipeline でのスキャンアクションをサポートするように CI/CD 統合機能を更新します。詳細については、CodePipeline での Amazon Inspector スキャンアクションの使用」を参照してください。	2024 年 11 月 26 日

目次の更新	Amazon Inspector は目次を再編成して、Amazon Inspector SBOM Generator の章を含めます。詳細については、Amazon Inspector SBOM Generator」を参照してください。	2024 年 11 月 22 日
更新された機能	Amazon Inspector は、Amazon EC2 Fedora と Amazon ECR でサポートされているオペレーティングシステムのリストから 39 を削除します。 Amazon EC2 詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2024 年 11 月 22 日
更新された機能	Amazon Inspector は、Amazon ECR でサポートされているオペレーティングシステムのリストから Alpine 3.17 を削除します。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2024 年 11 月 22 日
更新された機能	Amazon Inspector は、Amazon Inspector SBOM Generator の以前のバージョンSbomgenにバージョンを追加します。 Amazon Inspector	2024 年 11 月 19 日

更新された機能	Amazon Inspector が AL2 をサポート対象ランタイムとして追加しました。詳細については、「Amazon Inspector でサポートされているオペレーティングシステムとプログラミング言語」を参照してください。	2024 年 8 月 26 日
更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントは、Amazon Inspector が AWS Lambdaで関数タグを返すことを許可します。	2024 年 7 月 31 日
更新された機能	Amazon Inspector が新しいセキュリティコントロールをリリースします。詳細については、「AWS Security Hub ユーザーガイド」の「Amazon Inspector のコントロール」を参照してください。	2024 年 7 月 11 日
更新された機能	Amazon Inspector SBOM Generator は、Dockerfiles と Docker コンテナイメージをスキャンして、セキュリティの脆弱性を引き起こす可能性のある設定ミスがないことを確認するようになりました。詳細については、「Amazon Inspector Dockerfile チェック」を参照してください。	2024 年 6 月 10 日

更新された機能	Amazon Inspector が CodeCatalyst アクションをサポートするように CI/CD 統合機能を更新し、Amazon Inspector 脆弱性スキャンを CodeCatalyst ワークフローに追加できるようになりました。詳細については、「CodeCatalyst アクションの使用」を参照してください。	2024 年 6 月 7 日
更新された機能	Amazon Inspector に、CIS スキャン結果の CSV ファイルをダウンロードするオプションが含まれました。詳細については、「Amazon EC2 インスタンスの Center for Internet Security (CIS) スキャン」の「CIS スキャン結果の表示とダウンロード」を参照してください。	2024 年 5 月 3 日
更新された機能	Amazon Inspector が CI/CD 統合機能を更新して GitHub Actions をサポートし、Amazon Inspector の脆弱性スキャンを GitHub ワークフローに追加できるようになりました。詳細については、「GitHub Actions での Amazon Inspector の使用」を参照してください。	2024 年 4 月 29 日

更新された機能	Amazon Inspector がマネージドポリシー AmazonInspector2FullAccess を更新し、サービスにリンクされたロール AWSServiceRoleForAmazonInspector2Agentless が作成されるようになりました。これにより、ユーザーは Amazon Inspector を有効にするときに、エージェントベーススキャンとエージェントレススキャンを実行できます。	2024 年 4 月 24 日
更新された機能	Amazon Inspector は、終了した検出結果の保持期間を 30 日から 7 日に更新します。詳細については、「Amazon Inspector の検出結果について」を参照してください。	2024 年 2 月 12 日
更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントにより、Amazon Inspector はインスタンスの CIS スキャンを開始できます。	2024 年 1 月 23 日
新しいポリシー	Amazon Inspector が、インスタンスプロファイルの一部として使用してインスタンスの CIS スキャンを許可できる、新しい AmazonInspector2ManagedCisPolicy ポリシーを追加しました。	2024 年 1 月 23 日

新機能	Amazon Inspector は、コンテナイメージをプルするときに ECR 再スキャン期間を更新するようになりました。プッシュ日またはプル日に基づいて再スキャン期間を変更する方法については、「ECR 再スキャン期間の設定」を参照してください。	2024 年 1 月 23 日
新機能	Amazon Inspector は EC2 インスタンスで Center for Internet Security (CIS) スキャンを実行できるようになりました。詳細については、「Amazon Inspector の CIS スキャン」を参照してください。	2024 年 1 月 23 日
新機能	Amazon Inspector で CI/CD パイプライン内のコンテナイメージをスキャンできるようになりました。詳細については、「Amazon Inspector と CI/CD の統合」を参照してください。	2023 年 11 月 30 日

新しいポリシー	Amazon Inspector には、Amazon Inspector が エージェントレススキャンのために EC2 インスタンスから Amazon EBS スナップショットをスキャンすることを許可する新しいポリシーが追加されました。ポリシーの詳細については、「エージェントレススキャン」を参照してください。	2023 年 11 月 27 日
新機能	Amazon Inspector では、エージェントレススキャンによって、サポート対象 Linux Amazon EC2 インスタンスの SSM エージェントなしでのスキャンがサポートされるようになりました。詳細については、「エージェントレススキャン」を参照してください。	2023 年 11 月 27 日
新しくサポートされたリソース	Amazon Inspector は MacOS Amazon EC2 インスタンスのスキャンをサポートするようになりました。サポートされている MacOS バージョンについては、「Supported operating systems: Amazon EC2 scanning」を参照してください。	2023 年 10 月 5 日

新しいリージョン	Amazon Inspector は現在、アジアパシフィック (ジャカルタ)、アフリカ (ケープタウン)、アジアパシフィック (大阪)、および欧州 (チューリッヒ) で利用可能です。	2023 年 9 月 29 日
新機能	除外タグを使用して Amazon Inspector のスキャンから EC2 インスタンスを除外できるようになりました。	2023 年 9 月 14 日
新機能	Amazon Inspector には、Elastic Load Balancing ターゲットグループの一部である Amazon EC2 インスタンスのネットワーク設定をスキャンできるようにする新しいアクセス許可が追加されました。	2023 年 8 月 31 日
新機能	Amazon Inspector では、パッケージ脆弱性の検出結果に関する脆弱性インテリジェンスの詳細を提供できるようになりました。	2023 年 7 月 31 日
更新された機能	Amazon Inspector には、読み取り専用ユーザーがリソースのソフトウェア部品表 (SBOM) をエクスポートできる新しいアクセス許可が追加されました。	2023 年 6 月 29 日

新機能	Amazon Inspector によってスキャンされているリソースの SBOM をエクスポートできるようになりました。	2023 年 6 月 13 日
新機能	Lambda コードスキャンの一般提供が開始されました。Lambda コードスキャンの検出結果で特定されたコードを暗号化できる新機能が追加されました。さらに、Lambda コードスキャンにより、コードの修復と書き換えが提案されるようになりました。	2023 年 6 月 13 日
更新された機能	Amazon Inspector では AmazonInspector2ReadOnlyAccess ポリシーに新しいステートメントが追加されました。新しいステートメントにより、読み取り専用ユーザーは、自分のアカウントの Lambda コードスキャンのステータスと検出結果の詳細を取得できます。	2023 年 5 月 2 日
新機能	Amazon Inspector に脆弱性データベース検索が追加されました。これにより、Amazon Inspector が特定の CVE を対象としているかどうかを確認できます。	2023 年 5 月 1 日

更新された機能	Amazon Inspector は、Lambda スキャンをアクティブ化するときに Amazon Inspector がアカウントに AWS CloudTrail サービスにリンクされたチャネルを作成できるようにする新しいアクセス許可をAmazonInspector2ServiceRolePolicyポリシーに追加しました。これにより、Amazon Inspector はアカウントの CloudTrail イベントをモニタリングできます。	2023 年 4 月 30 日
更新された機能	Amazon Inspector では AmazonInspector2FullAccess ポリシーに新しいステートメントが追加されました。新しいステートメントにより、ユーザーは Lambda コードスキャンのコード脆弱性検出結果を取得できます。	2023 年 4 月 17 日
更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントにより、Amazon EC2 詳細検査向けに定義したカスタムパスに関する情報を Amazon Inspector が Amazon EC2 Systems Manager に送信できるようになります。	2023 年 4 月 17 日

新機能	Amazon Inspector は、Amazon Inspector 詳細検査という形で Linux EC2 インスタンスのサポートを追加しました。これにより、アプリケーションプログラミング言語パッケージのパッケージの脆弱性についてインスタンスがスキャンされます。	2023 年 4 月 17 日
更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントにより、Amazon Inspector は AWS Lambda 関数内の開発者コードのスキャンをリクエストし、Amazon CodeGuru Security からスキャンデータを受信できます。さらに、Amazon Inspector には IAM ポリシーを確認するためのアクセス許可が追加されました。Amazon Inspector はこの情報を使用して Lambda 関数のコード脆弱性をスキャンします。	2023 年 2 月 28 日
新機能	Amazon Inspector では、Lambda コードスキャンという形で Lambda 関数のサポートが追加され、Lambda 関数の開発者コードをスキャンしてセキュリティの脆弱性を調べます。	2023 年 2 月 28 日

更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントにより、Amazon Inspector は AWS Lambda 関数が最後に呼び出された日に関する情報を CloudWatch から取得できます。この情報を使用して、過去 90 日間にアクティブだった環境内の Lambda 関数に絞ってスキャンを行います。	2023 年 2 月 20 日
更新された機能	Amazon Inspector では AmazonInspector2ServiceRolePolicy ポリシーに新しいステートメントが追加されました。新しいステートメントにより、Amazon Inspector では AWS Lambda 関数に関する情報を取得できます。Amazon Inspector はこの情報を使用して Lambda 関数をスキャンし、セキュリティの脆弱性がないか調べます。	2022 年 11 月 28 日
新機能	Amazon Inspector は、スキャン AWS Lambda 関数のサポートを追加します。	2022 年 11 月 28 日

更新された内容	Amazon Inspector から Amazon Simple Storage Service (Amazon S3) バケットに調査結果レポートをエクスポートする手順、ポリシー例、およびヒントを追加しました。	2022 年 10 月 14 日
新しいコンテンツ	Amazon Inspector コンソールを使用した AWS 環境の Amazon Inspector カバレッジの評価に関する情報を追加しました。 Amazon Inspector この情報には、環境内の個々のリソースのステータス値についての説明が含まれます。	2022 年 10 月 7 日
新機能	Amazon Inspector では、パッケージの脆弱性を修復する方法について、さらなる詳細を提供するようになりました。検出結果の詳細に新しいフィールドが追加されました。新しいフィールドには、パッケージの更新によって修正が可能かどうかについてのコンテキストが表示されます。修正が入手可能な場合は、検出結果の [推奨される修復] セクションに、修正を行うために実行できるコマンドが表示されます。	2022 年 9 月 2 日

更新された機能	Amazon Inspector には AmazonInspector2ServiceRolePolicy ポリシーに新しいアクションが追加されました。新しいアクションにより、Amazon Inspector は SSM 関連付けの実行を記述できます。Amazon Inspector では、`AmazonInspector2` 所有の SSM ドキュメントとの SSM 関連付けを作成、更新、削除、開始できるように、リソーススコープも追加されました。	2022 年 8 月 31 日
新機能	Amazon Inspector は Windows インスタンスのスキャンをサポートするようになりました。Amazon Inspector では、サポートされている Windows オペレーティングシステムを実行している SSM マネージドインスタンスをスキャンできるようになりました。Windows ホストのスキャンは Amazon Inspector SSM プラグインによって実行されます。このプラグインは、Amazon Inspector によって自動的に作成された新しい SSM 関連付けを通じてインストールされ、呼び出されます。	2022 年 8 月 31 日

更新された機能	Amazon Inspector は、Amazon Inspector が他の AWS パーティションでソフトウェアインベントリを収集できるように、AmazonInspector2ServiceRolePolicyポリシーのリソーススコープを更新しました。	2022 年 8 月 12 日
更新された機能	AmazonInspector2ServiceRolePolicy ポリシーでは、Amazon Inspector がアクションのリソーススコープを再構築し、Amazon Inspector が SSM 関連付けを作成、削除、および更新できるようにしました。	2022 年 8 月 10 日

新機能	Amazon Inspector では ECR 自動再スキャン期間設定の変更がサポートされるようになりました。Amazon ECR 自動再スキャン期間設定で、Amazon Inspector がリポジトリにプッシュされたイメージを継続的にモニタリングする期間を決定します。イメージがスキャン期間より古い場合、Amazon Inspector はイメージをスキャンせず、そのイメージに関する既存の検出結果をすべて終了します。すべての新規アカウントでは、ECR 自動再スキャン期間が自動的にライフタイム期間に設定されます。以前に作成されたアカウントの ECR 自動再スキャン期間は 30 日でしたが、スキャンの期間を 30 日、180 日、またはライフタイム、から選択できるようになりました。	2022 年 6 月 25 日
新しい機能	Amazon Inspector は、Amazon Inspector 機能への読み取り専用アクセスを許可する新しい AWS 管理ポリシーである AmazonInspector2ReadOnlyAccessポリシーを追加しました。	2022 年 1 月 21 日
一般提供	これは、「Amazon Inspector ユーザーガイド」の初回一般リリースです。	2021 年 11 月 29 日

AWS 用語集

最新の AWS 用語については、 AWS の用語集 リファレンスのAWS 用語集を参照してください。

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。