aws

用户指南

Amazon Inspector

Amazon Inspector: 用户指南

Table of Contents

什么是 Amazon Inspector？

Amazon Inspector 是一项漏洞管理服务，可自动发现工作负载并持续对其进行扫描，以查找软件漏洞和意外的网络暴露。Amazon Inspector 可发现并扫描 Amazon EC2 实例、Amazon ECR 中的容器映像以及 Lambda 函数。当 Amazon Inspector 检测到软件漏洞或意外网络暴露时，它会创建一个调查发现，详细介绍问题所在。您可以使用 Amazon Inspector 控制台或 API 管理调查发现。

> **Note**
>
> 提交支持请求时，Amazon Inspector 可能会访问和处理相关调查结果的存储 AWS 区域 地点（但在相同的地理位置），以解决问题。

主题

Amazon Inspector 的特征

集中管理多个 Amazon Inspector 账户

如果您的 AWS 环境有多个帐户，则可以使用 Organizations 通过一个账户集中管理您的 AWS 环境。使用此方法时，您可以将某一账户指定为 Amazon Inspector 的委托管理员账户。

只需单击一下即可为整个组织激活 Amazon Inspector。此外，您还可以在将来有成员加入组织时自动为他们激活服务。Amazon Inspector 委托管理员账户可以管理组织成员的调查发现数据和某些设置。这包括查看所有成员账户的汇总结果详细信息、激活或停用对成员账户的扫描，以及查看 AWS 组织内扫描的资源。

持续扫描环境，查找漏洞和网络风险

有了 Amazon Inspector，便无需手动安排或配置评测扫描。Amazon Inspector 会自动发现并开始扫描符合条件的资源。Amazon Inspector 会自动重新扫描资源，以应对可能会引入新漏洞的变更，在资源的整个生命周期内持续评测您的环境，上述变更包括：在 EC2 实例中安装新程序包、安装补丁，以及发布影响资源的新常见漏洞和风险（CVE）。与传统的安全扫描软件不同，Amazon Inspector 对机群性能的影响微乎其微。

当发现漏洞或开放的网络路径时，Amazon Inspector 会生成调查发现供您调查。调查发现包括有关漏洞、受影响资源和补救建议的全面详细信息。如果您对调查发现进行了适当的补救，Amazon Inspector 会自动检测到补救措施并关闭该调查发现。

使用 Amazon Inspector 风险评分准确评测漏洞

当 Amazon Inspector 通过扫描收集有关环境的信息时，它会提供专门针对您的环境量身定制的严重性评分。Amazon Inspector 会检查构成漏洞的国家漏洞数据库（NVD）基本评分的安全指标，并根据您的计算环境进行调整。例如，如果漏洞可通过网络利用，但互联网上没有通往相应 Amazon EC2 实例的开放网络路径，则该服务可能会降低该实例的调查发现的 Amazon Inspector 评分。该评分采用 CVSS 格式，是对 NVD 提供的基本通用漏洞评分系统（CVSS）评分的修改。

使用 Amazon Inspector 控制面板识别具有高影响力的调查发现

Amazon Inspector 控制面板可提供整个环境中调查发现的总体视图。您可以在此控制面板中查看调查发现的详细信息。此控制面板包含有关环境中扫描覆盖范围、最重要的调查发现以及调查发现最多的资源的简化信息。Amazon Inspector 控制面板中基于风险的补救面板显示了影响实例和映像数量最多的调查发现。通过此面板，您可以更轻松地确定对环境影响最大的调查发现，查看调查发现的详细信息以及建议的解决方案。

使用自定义视图管理调查发现

除了控制面板外，Amazon Inspector 控制台还提供了调查发现视图。此页面列出了您的环境的所有调查发现，并提供了各个调查发现的详细信息。您可以查看按类别或漏洞类型分组的调查发现。在每个视图中，您都可以使用筛选条件进一步自定义结果。您还可以使用筛选条件创建抑制规则，在视图中隐藏不需要的调查发现。

您可以使用筛选条件和抑制规则生成调查发现报告，展示所有调查发现或自定义的调查发现。报告可以使用 CSV 或 JSON 格式生成。

使用其他服务和系统监控和处理调查发现

为了支持与其他服务和系统的集成，Amazon Inspector 将调查结果 EventBridge作为发现事件发布给亚马逊。 EventBridge 是一种无服务器事件总线服务，可以将调查结果数据路由到目标，例如 AWS Lambda 函数和亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题。借 EventBridge 助，您可以近乎实时地监控和处理调查结果，这是现有安全与合规工作流程的一部分。

如果你已激活 AWS Security Hub CSPM，那么 Amazon Inspector 还将向 Sec urity Hub CSPM 发布调查结果。Security Hub CSPM 是一项服务，可全面了解您在整个 AWS 环境中的安全状况，并帮助您根据安全行业标准和最佳实践检查您的环境。借助 Security Hub CSPM，您可以更轻松地监控和处理您的发现，这是对组织安全态势进行更广泛分析的一部分。 AWS

访问 Amazon Inspector

Amazon Inspector 在大多数版本中都可用 AWS 区域。有关当前可使用 Amazon Inspector 的区域的列表，请参阅 Amazon Web Services 一般参考中的 Amazon Inspector 端点和配额。要了解有关 AWS 区域的更多信息，请参阅 Amazon Web Services 一般参考中的管理 AWS 区域。在每个区域，您都可以通过以下方式使用 Amazon Inspector：

AWS 管理控制台

AWS 管理控制台 是一个基于浏览器的界面，可用于创建和管理 AWS 资源。作为该控制台的一部分，Amazon Inspector 控制台提供对 Amazon Inspector 账户和资源的访问。您可以通过 Amazon Inspector 控制台执行 Amazon Inspector 任务。

AWS 命令行工具

使用 AWS 命令行工具，您可以在系统的命令行中发出命令来执行 Amazon Inspector 任务。与控制台相比，使用命令行更快、更方便。如果要构建执行任务的脚本，命令行工具也会十分有用。

AWS 提供了两组命令行工具： AWS Command Line Interface (AWS CLI) 和 AWS Tools for PowerShell。有关安装和使用的信息 AWS CLI，请参阅《AWS 命令行界面用户指南》。有关安装和使用的 “工具” 的信息 PowerShell，请参阅《AWS Tools for PowerShell 用户指南》。

AWS SDKs

AWS SDKs 包含适用于各种编程语言和平台（包括 Java、Go、Python、C++ 和.NET）的库和示例代码。它们 SDKs 提供了对 Amazon Inspector 和其他内容的便捷编程访问 AWS 服务。它们可以执行多种任务，例如以加密方式对请求进行签名、管理错误以及自动重试请求等。有关安装和使用的信息 AWS SDKs，请参阅构建工具 AWS。

Amazon Inspector REST API

Amazon Inspector REST API 让您能够以编程方式全面访问 Amazon Inspector 账户和资源。借助此 API，您可以直接向 Amazon Inspector 发送 HTTPS 请求。但是，与 AWS 命令行工具和不同 SDKs，使用此 API 需要您的应用程序处理低级细节，例如生成哈希值来签署请求。

Amazon Inspector 入门

本节提供了在激活 Amazon Inspector 之前需要考虑的信息，并提供一个入门教程，介绍如何激活 Amazon Inspector 并使用 Amazon Inspector 控制台以及 Amazon Inspector API 查看调查发现。

主题

激活 Amazon Inspector 之前

在激活 Amazon Inspector 之前，请注意以下几点：

Amazon Inspector 是一项区域服务

您的数据存储在您激活 Amazon Inspector AWS 区域 的地方。在计划使用 Amazon Inspector 的所有 AWS 区域 地方，重复入门教程第一部分中的步骤。

Amazon Inspector 创建了与服务相关的角色 AWSService RoleForAmazonInspector 2 和 AWSService RoleForAmazonInspector 2Agentless

服务相关角色是 AWS Identity and Access Management (IAM) 中与 AWS 服务关联的角色。AWSServiceRoleForAmazonInspector2 和 AWSServiceRoleForAmazonInspector2Agentless 允许 Amazon Inspector 进行 AWS 服务 安全评估所需的访问权限。

具有管理员权限的 IAM 身份就可以启用 Amazon Inspector

通过 IAM 或 AWS IAM Identity Center 创建用户，以此来保护您的凭证。这将有助于您确保用户仅拥有管理 Amazon Inspector 所需的权限。有关更多信息，请参阅AWS 托管策略：AmazonInspectorFullAccess。

自动启用混合扫描

混合扫描包括基于代理的扫描和无代理扫描。默认情况下，Amazon Inspector 在所有符合条件的亚马逊 EC2 实例上使用这些扫描方法。有关更多信息，请参阅使用 Amazon Ins pector 扫描亚马逊 EC2 实例。

Amazon ECR 扫描和 Lambda 函数扫描不需要 SSM Agent

基于代理的扫描使用 SSM Agent 收集软件清单。无代理扫描使用 Amazon EBS 快照来收集软件清单。

> **Note**
>
> 默认情况下，SSM 代理已安装在基于亚马逊系统映像的亚马逊 EC2 实例中。但是，在某些情况下，您可能需要手动激活 SSM Agent。有关更多信息，请参阅《AWS Systems Manager 用户指南》中的使用 SSM Agent。

每月费用基于扫描的工作负载

有关更多信息，请参阅 Amazon Inspector 定价。

支持多账户 AWS Organizations

对于使用的组织 AWS Organizations，Amazon Inspector 支持委托管理员管理和基于组织策略的启用。组织策略提供集中化治理，并自动启用新帐户。有关这两种方法的详细说明，请参阅入门教程：激活 Amazon Inspector。

入门教程：激活 Amazon Inspector

本主题介绍了如何为独立账户环境（成员账户）以及多账户环境（委派管理员账户）激活 Amazon Inspector。激活 Amazon Inspector 后，它会自动开始发现工作负载并扫描其中是否存在软件漏洞和意外网络暴露。

Standalone account environment

以下步骤介绍了如何在控制台中为成员账户激活 Amazon Inspector。要以编程方式激活 Amazon Inspector，inspec tor2-。enablement-with-cli

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 选择开始使用。
3. 选择激活 Amazon Inspector。

为独立账户激活 Amazon Inspector 时，默认情况下会激活所有扫描类型。有关成员账户的信息，请参阅了解 Amazon Inspector 中的委派管理员账户和成员账户。

Multi-account (with AWS Organizations policy)

AWS Organizations 策略为在整个组织中启用 Amazon Inspector 提供了集中管理。当您使用组织策略时，系统会自动管理该政策涵盖的所有账户的 Amazon Inspector 启用，并且成员账户无法使用 Amazon Inspector API 修改策略管理的扫描。

先决条件

- 您的账户必须是 AWS Organizations 组织的一部分。
- 您必须拥有在中创建和管理组织策略的权限 AWS Organizations。
- 必须在中启用 Amazon Inspector 的可信访问权限 AWS Organizations。有关说明，请参阅AWS Organizations 用户指南中的为 Amazon Inspector 启用可信访问。
- Amazon Inspector 服务相关角色应存在于管理账户中。要创建它们，请在管理账户中启用 Amazon Inspector，或者从管理账户运行以下命令：
  - `aws iam create-service-linked-role --aws-service-name inspector2.amazonaws.com`
  - `aws iam create-service-linked-role --aws-service-name agentless.inspector2.amazonaws.com`
- 应指定 Amazon Inspector 的委托管理员。

> **Note**
>
> 如果没有管理账户和委托管理员等服务相关的 Amazon Inspector 角色，组织政策将强制启用 Amazon Inspector，但成员账户将无法与 Amazon Inspector 组织关联以进行集中调查和账户管理。

使用 AWS Organizations 策略启用 Amazon Inspector

1. 在创建组织策略之前，请为 Amazon Inspector 指定一名委托管理员，以确保成员账户与 Amazon Inspector 组织关联，以便集中查看调查结果。登录 AWS Organizations 管理账户，在 https://console.aws.amazon.com/inspector/v2/hom e 上打开 Amazon Inspector 控制台，然后按照中的步骤进行操作。为您的 AWS 组织指定委派管理员

> **Note**
>
> 我们强烈建议您的 AWS Organizations Amazon Inspector 委托管理员账户 ID 和 Amazon Inspector 指定的委托管理员账户 ID 保持不变。如果 AWS Organizations 委托管理员账户 ID 与 Amazon Inspector 委托管理员账户 ID 不同，则 Amazon Inspector 会优先使用检查员指定的账户 ID。当未设置 Amazon Inspector 委托管理员但已设置 AWS Organizations 委托管理员且管理账户具有亚马逊检查员服务相关角色时，Amazon Inspector 会自动将 AWS Organizations 委托管理员账户 ID 分配为亚马逊检查员委托的管理员。

2. 在 Amazon Inspector 控制台中，从管理账户导航到常规设置。在 “委托策略” 下，选择 “附加声明”。在 “附加策略声明” 对话框中，查看策略，选择 “我确认我已查看策略并了解其授予的权限”，然后选择 Attach stat ement。

> **Important**
>
> 管理账户必须具有以下权限才能附加委托政策声明：
>
> - 来自 AmazonInspector2 FullAccess _v2 托管策略的 Amazon Inspector 权限
> - AWS Organizations `organizations:PutResourcePolicy`来自AWSOrganizationsFullAccess托管策略的权限
>
> 如果缺少`organizations:PutResourcePolicy`权限，则操作将失败并显示错误：`Failed to attach statement to the delegation policy`。

3. 接下来，创建 Amazon Inspector AWS Organizations 政策。从导航窗格中选择 “管理”，然后选择 “配置”。
4. 配置漏洞管理策略。提供详细信息以及策略的名称和描述（可选）。
5. 在 “配置 Inspector” 页面的 “详细信息” 部分，输入策略的名称和描述。在 “能力选择” 中，执行以下任一操作：
   - 选择配置并启用所有功能（推荐）。这将启用所有 Inspector 功能，包括 EC2、ECR、Lambda 标准、Lambda 代码扫描和代码安全。
   - 选择选择权能子集。选择任何应开启的扫描类型功能。
6. 在 “账户选择” 部分，选择以下选项之一：

- 如果要将配置应用于所有组织单位和帐户，请选择 “所有组织单位和帐户”。
- 如果要将配置应用于特定的组织单位和帐户，请选择 “特定组织单位和帐户”。如果您选择此选项，请使用搜索栏或组织结构树来指定将应用该策略的组织单元和账户。
- 如果您不想将配置应用于任何组织单位或帐户，请选择 “无组织单位或帐户”。

7. 在 “区域” 部分，选择 “启用所有区域”、“禁用所有区域” 或 “指定区域”。

   - 如果您选择启用所有区域，则可以决定是否自动启用新区域。
   - 如果您选择禁用所有区域，则可以决定是否自动禁用新区域。
   - 如果选择指定区域，则必须选择要启用和禁用的区域。

   （可选）有关高级设置，请参阅中的指南 AWS Organizations。

   （可选）对于资源标签，将标签添加为键值对，以帮助您轻松识别配置。

8. 选择 “下一步”，查看您的更改，然后选择 “应用”。您的目标账户是根据策略进行配置的。策略的配置状态显示在 “策略” 页面的顶部。每项功能都提供有关其是否已配置或部署失败的状态。对于任何故障，请选择失败消息的链接以查看更多详细信息。要查看账户级别的有效策略，您可以查看配置页面上的组织选项卡，可以在其中选择一个账户。

当通过组织策略启用 Amazon Inspector 时，该政策所涵盖的账户无法通过 Amazon Inspector API 或控制台禁用策略管理的扫描类型。有关委托管理员和成员账户在组织政策下可以做什么和不能做什么的详细信息，请参阅使用 Amazon Inspector 管理多个账户 AWS Organizations。

Multi-account (without AWS Organizations policy)

> **Note**
>
> 您必须使用 AWS Organizations 管理账户才能完成此过程。只有 AWS Organizations 管理账号才能指定委派管理员。可能需要权限才能指定委派管理员。有关更多信息，请参阅 指定委托管理员所需的权限。

当您首次激活 Amazon Inspector 时，Amazon Inspector 会为账户创建服务相关角色 `AWSServiceRoleForAmazonInspector`。有关 Amazon Inspector 如何使用服务相关角色的信息，请参阅对 Amazon Inspector 使用服务相关角色。

指定 Amazon Inspector 委托管理员

1. 登录 AWS Organizations 管理账户，然后在 https://console.aws.amazon.com/inspector/v2/home 上打开 Amazon Inspector 控制台。
2. 选择开始。
3. 在 “委托管理员” 下，输入 AWS 账户 要指定为委派管理员的 12 位 ID。
4. 选择委派，然后再次选择委派。
5. （可选）如果您想为 AWS Organizations 管理账户激活 Amazon Inspector，请在 “服务权限” 下选择 “激活亚马逊检查器”。

当您指定委派管理员时，默认情况下会为该账户激活所有扫描类型。有关委派管理员账户的信息，请参阅了解 Amazon Inspector 中的委派管理员账户和成员账户。

Amazon Inspector 中的自动扫描类型

Amazon Inspector 使用专用扫描引擎，来监控您的资源中是否存在可操作的软件漏洞和意外网络暴露。当 Amazon Inspector 检测到软件漏洞或意外网络暴露时，它会创建一个调查发现。首次激活 Amazon Inspector 时，您的账户会自动注册所有扫描类型，其中包括 Amazon EC2 扫描、Amazon ECR 扫描、Lambda 标准扫描。

> **Note**
>
> Lambda 代码扫描是 Lambda 函数扫描的可选层，您可以随时激活该扫描。

主题

Amazon Inspector 扫描类型概述

Amazon Inspector 提供不同的扫描类型，这些类型侧重于您 AWS 环境中的特定资源类型。

Amazon EC2 扫描

当您激活 Amazon EC2 扫描时，Amazon Inspector 会扫描您的 EC2 实例中是否存在常见漏洞和漏洞 (CVEs)、网络泄露问题、网络可访问性问题、操作系统和编程语言包漏洞。Amazon Inspector 使用实例上安装的 SSM Agent 或通过实例的 Amazon EBS 快照执行扫描。有关更多信息，请参阅使用 Amazon Inspector 扫描 Amazon EC2 实例。默认情况下，在激活 Amazon EC2 扫描时，会自动启用混合扫描模式。有关更多信息，请参阅无代理扫描。

Amazon ECR 扫描

激活 Amazon ECR 扫描后，Amazon Inspector 会将您私有注册表中的所有存储库从基本扫描容器存储库转换为增强扫描存储库。您可以使用包含规则将此设置配置为仅在推送时扫描或扫描特定的存储库。Amazon Inspector 仅扫描 ECR 中处于活动状态（`imageStatus`字段

为ACTIVE）的 ECR 容器镜像。Amazon Inspector 会扫描过去 30 天内在 ECR 中推送或过渡到激活 (`lastActivatedAt`) 或在过去 90 天内提取的所有图像。默认情况下，Amazon Inspector 会继续监控映像 90 天。您可以随时更改此设置。有关更多信息，请参阅 使用 Amazon Inspector 扫描 Amazon Elastic Container Registry 容器映像。

Lambda 标准扫描

激活 Lambda 标准扫描后，Amazon Inspector 会发现您账户中的所有 Lambda 函数并立即扫描它们以查找漏洞。Amazon Inspector 会在部署新的 Lambda 函数和层后对其进行扫描。当它们更新或发布新 CVEs 内容时，Amazon Inspector 会重新扫描它们。有关扫描的更多信息，请参阅使用 Amazon Inspector 进行扫描 AWS Lambda。

Lambda 标准扫描 + Lambda 代码扫描

激活 Lambda 代码扫描后，Amazon Inspector 会发现您账户中的 Lambda 函数和层，并扫描它们以查找代码漏洞。这种扫描会评估 Lambda 函数中使用的应用程序包依赖关系。 CVEs激活此扫描类型后，还会激活 Lambda 标准扫描。有关更多信息，请参阅 使用 Amazon Inspector 进行扫描 AWS Lambda。

Amazon Inspector 代码安全性

此扫描类型会利用 Amazon Q 开发者版扫描引擎来扫描第一方应用程序代码、第三方应用程序依赖关系和基础设施即代码以查找漏洞。有关更多信息，请参阅 Amazon Inspector 代码安全性。

激活扫描类型

您可以随时激活某个扫描类型。激活某个扫描类型后，Amazon Inspector 将开始针对该扫描类型扫描符合条件的资源。

Amazon EC2 扫描

这种扫描类型会从 Amazon EC2 实例提取元数据，然后再将元数据与从安全公告中收集的规则进行比较。激活此扫描类型后，Amazon Inspector 会扫描您账户中所有符合条件的 Amazon EC2 实例，以查找其中是否存在程序包漏洞和网络可达性问题。激活此扫描类型后，您可以在实例选项卡中查看正在扫描的实例数量。

Amazon ECR 扫描

此扫描类型会扫描 Amazon ECR 中的容器映像和容器存储库。激活此扫描类型时，可以将私有注册表的扫描配置设置从基本扫描更改为增强扫描。激活 Amazon ECR 扫描后，您可以在容器映像和容器存储库选项卡中查看正在扫描的映像和存储库数量。

Lambda 标准扫描 + Lambda 代码扫描

Lambda 标准扫描是默认的 Lambda 扫描类型。激活 Lambda 标准扫描后，将对在过去 90 天内被调用或更新过的所有 Lambda 函数进行扫描，以查找其中是否存在软件漏洞。激活 Lambda 标准扫描后，您可以在 Lambda 函数选项卡中查看正在扫描的 Lambda 函数的数量。

Lambda 代码扫描会扫描 Lambda 函数中的自定义应用程序代码。激活 Lambda 代码扫描后，将对过去 90 天内被调用或更新过的所有 Lambda 函数进行扫描，以查找其中是否存在代码漏洞。激活 Lambda 标准扫描后，您可以在 Lambda 函数选项卡中查看正在扫描代码漏洞的 Lambda 函数的数量。

> **Note**
>
> 如果要激活 Lambda 代码扫描，则必须先激活 Lambda 代码扫描。

Amazon Inspector 代码安全性

此扫描类型会扫描第一方应用程序代码、第三方应用程序依赖关系以及基础设施即代码中是否存在漏洞。激活代码安全后，Amazon Inspector 会根据您的扫描配置开始扫描您的代码存储库中是否存在代码漏洞。激活 Amazon Inspector 代码安全后，您可以在代码存储库选项卡中查看正在扫描的代码存储库的数量。

激活扫描

以下步骤介绍了如何在 Amazon Inspector 中激活扫描类型。

> **Note**
>
> 如果您是 AWS 组织的委托管理员，则可以使用 shell 脚本为多个区域的多个账户启用 Amazon Inspector 扫描类型。有关更多信息，请参阅 insp ector2-enablement-with-cli on。 GitHub否则，请在以 Amazon Inspector 委派管理员的身份登录后完成以下步骤。

Console

激活扫描

1. 在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用页面右上角的选择 AWS 区域 器，选择要激活新扫描类型的区域。

3. 在导航窗格中，选择账户管理。
4. 在账户管理页面上，选择要为其激活扫描类型的账户。
5. 选择激活，然后选择要激活的扫描类型。
6. （推荐）在要激活该扫描类型的每个 AWS 区域 步骤中重复这些步骤。

API

运行启用 API 操作。在请求中，提供 IDs 您正在激活扫描的账户、等效令牌以及一个或多个`EC2`、`ECR`、`LAMBDA`、或，`LAMBDA_CODE`以激活该类型的扫描。`resourceTypes`

使用 Amazon Inspector 扫描 Amazon EC2 实例

Amazon Inspector Amazon EC2 扫描会从 EC2 实例提取元数据，然后再将这些元数据与从安全公告中收集的规则进行比较。Amazon Inspector 会扫描实例中是否存在程序包漏洞和网络可达性问题，然后生成调查发现。Amazon Inspector 每 12 小时执行一次网络可访问性扫描，并根据与 EC2 实例关联的扫描方法以可变的节奏进行软件包漏洞扫描。

程序包漏洞扫描可以使用基于代理或无代理的扫描方法执行。这两种扫描方法决定了 Amazon Inspector 如何以及何时从 EC2 实例收集软件清单来进行程序包漏洞扫描。基于代理的扫描使用 SSM Agent 收集软件清单，无代理扫描使用 Amazon EBS 快照收集软件清单。

Amazon Inspector 使用您为账户激活的扫描方法。首次激活 Amazon Inspector 时，您的账户会自动注册混合扫描类型，同时使用这两种扫描方法。不过，您可以随时更改此设置。有关如何激活扫描类型的信息，请参阅激活扫描类型。本节提供了有关 Amazon EC2 扫描的信息。

> **Note**
>
> Amazon EC2 扫描不会扫描与虚拟环境相关的文件系统目录，即使这些目录是通过深度检查预调配的也是如此。例如，由于路径 `/var/lib/docker/` 通常用于容器运行时，因此不会对其进行扫描。

基于代理的扫描

在所有符合条件的实例上，我们使用 SSM 代理持续执行基于代理的扫描。对于基于代理的扫描，Amazon Inspector 使用 SSM 关联以及通过这些关联安装的插件从实例收集软件清单。除了对操作系统程序包进行程序包漏洞扫描外，Amazon Inspector 基于代理的扫描还可以通过基于 Linux 的

Amazon EC2 实例的 Amazon Inspector 深度检查，检测基于 Linux 的实例中的应用程序编程语言包是否存在程序包漏洞。

以下过程说明了 Amazon Inspector 如何使用 SSM 收集清单和执行基于代理的扫描：

1. Amazon Inspector 会在您的账户中创建 SSM 关联，以便从实例中收集清单。对于某些实例类型（Windows 和 Linux），这些关联会在单个实例上安装插件以收集清单。
2. Amazon Inspector 使用 SSM 从实例中提取程序包清单。
3. Amazon Inspector 会评估提取的清单，并针对检测到的任何漏洞生成调查发现。

> **Note**
>
> 对于基于代理的扫描，Amazon EC2 实例必须由同一 AWS 账户中的 SSM 进行管理。

符合条件的实例

如果实例满足以下条件，Amazon Inspector 将使用基于代理的方法对其进行扫描：

- 该实例具有支持的操作系统。有关支持的操作系统列表，请参阅the section called “支持的操作系统：Amazon EC2 扫描”的基于代理的扫描支持列。
- 未使用 Amazon Inspector EC2 排除标签将该实例排除在扫描范围之外。
- 该实例由 SSM 托管。有关验证和配置该代理的说明，请参阅配置 SSM 代理。

基于代理的扫描行为

使用基于代理的扫描方法时，在以下情况下，Amazon Inspector 会对 EC2 实例启动新的漏洞扫描：

- 启动新 EC2 实例时。
- 在现有 EC2 实例（Linux 和 Mac）上安装新软件时。
- Amazon Inspector 在其数据库中添加新的常见漏洞和风险（CVE）项目，且该 CVE 与您的 EC2 实例（Linux 和 Mac）相关时。

初始扫描完成后，Amazon Inspector 会更新 EC2 实例的上次扫描时间字段。此后，当 Amazon Inspector 评估 SSM 清单时（默认为每 30 分钟一次），或者由于影响实例的新 CVE 被添加到 Amazon Inspector 数据库而需要重新扫描该实例时，上次扫描时间字段就会更新。

您可以通过账户管理页面的“实例”选项卡或通过使用 ListCoverage 命令，来查看上次扫描 EC2 实例是否存在漏洞的时间。

配置 SSM 代理

为了让 Amazon Inspector 检测到使用基于代理的扫描方法的 Amazon EC2 实例的软件漏洞，该实例必须是 Amazon EC2 Systems Manager（SSM）中的托管实例。SSM 托管实例已安装并运行了 SSM 代理，SSM 有权管理该实例。如果您已经在使用 SSM 来管理实例，那么无需执行其他步骤，即可开始基于代理的扫描。

SSM 代理默认安装在根据某些 Amazon 系统映像 (AMIs) 创建的 EC2 实例上。有关更多信息，请参阅 AWS Systems Manager 用户指南中的关于 SSM 代理。但是，即使已经安装了 SSM 代理，您也可能需要手动激活 SSM 代理，并授予 SSM 管理实例的权限。

以下步骤介绍了如何使用 IAM 实例配置文件将 Amazon EC2 实例配置为托管实例。这些步骤还提供了指向 AWS Systems Manager 用户指南中更多详细信息的链接。

附加实例配置文件时，建议使用 AmazonSSMManagedInstanceCore 策略。该策略拥有 Amazon Inspector EC2 扫描所需的所有权限。

> **Note**
>
> 您还可以使用 SSM 默认主机管理配置，自动通过 SSM 管理所有 EC2 实例，而无需使用 IAM 实例配置文件。有关更多信息，请参阅默认主机管理配置。

为 Amazon EC2 实例配置 SSM

1. 如果操作系统供应商未安装 SSM 代理，请先安装。有关更多信息，请参阅使用 SSM 代理。
2. AWS CLI 使用验证 SSM 代理是否正在运行。有关更多信息，请参阅检查 SSM 代理状态并启动代理。
3. 向 SSM 授予管理实例的权限。您可以通过创建 IAM 实例配置文件并将其附加到实例来授予相应权限。我们建议使用 AmazonSSMManagedInstanceCore 策略，因为该策略具有 Amazon Inspector 扫描所需的 SSM Distributor、SSM Inventory 和 SSM State Manager 权限。有关创建具有这些权限的实例配置文件并将其附加到实例的说明，请参阅为 Systems Manager 配置实例权限。
4. （可选）激活 SSM 代理的自动更新。有关更多信息，请参阅自动更新 SSM 代理。
5. （可选）将 Systems Manager 配置为使用 Amazon Virtual Private Cloud（Amazon VPC）端点。有关更多信息，请参阅创建 Amazon VPC 端点。

> **Important**
>
> Amazon Inspector 需要您的账户中具有 Systems Manager State Manager 关联才能收集软件应用程序清单。如果不存在相应关联，Amazon Inspector 会自动创建一个名为 `InspectorInventoryCollection-do-not-delete` 的关联。
> Amazon Inspector 还需要资源数据同步，如果不存在相应同步，则会自动创建一个名为 `InspectorResourceDataSync-do-not-delete` 的同步。有关更多信息，请参阅 AWS Systems Manager 用户指南中的配置清单的资源数据同步。每个账户可在每个区域拥有一定数量的资源数据同步。有关更多信息，请参阅 SSM 端点和配额中资源数据同步的最大数量（ AWS 账户 每个区域）。

为扫描创建的 SSM 资源

Amazon Inspector 需要您的账户中有大量 SSM 资源才能运行 Amazon EC2 扫描。以下资源是在您首次激活 Amazon Inspector EC2 扫描时创建的：

> **Note**
>
> 如果仍在为您的账户激活 Amazon Inspector Amazon EC2 扫描时删除了其中任何 SSM 资源，Amazon Inspector 将在下一个扫描间隔尝试重新创建这些资源。

`InspectorInventoryCollection-do-not-delete`

这是一个 Systems Manager State Manager (SSM) 关联，Amazon Inspector 使用它从您的 Amazon EC2 实例收集软件应用程序清单。如果您的账户已经有用于从 `InstanceIds*` 中收集清单的 SSM 关联，则 Amazon Inspector 将使用该关联而不是自己创建。

`InspectorResourceDataSync-do-not-delete`

这是一个资源数据同步，Amazon Inspector 使用它将收集的清单数据从 Amazon EC2 实例发送到 Amazon Inspector 拥有的 Amazon S3 存储桶。有关更多信息，请参阅 AWS Systems Manager 用户指南中的配置清单的资源数据同步。

`InspectorDistributor-do-not-delete`

这是 Amazon Inspector 用于扫描 Windows 实例的 SSM 关联。此关联会在 Windows 实例上安装 Amazon Inspector SSM 插件。如果插件文件被无意中删除，则此关联将在下一个关联间隔重新安装它。

`InvokeInspectorSsmPlugin-do-not-delete`

这是 Amazon Inspector 用于扫描 Windows 实例的 SSM 关联。此关联使 Amazon Inspector 可以使用该插件启动扫描，您也可以使用它来设置扫描 Windows 实例的自定义间隔。有关更多信息，请参阅 为 Windows 实例扫描设置自定义计划。

`InspectorLinuxDistributor-do-not-delete`

这是 Amazon Inspector 用于 Amazon EC2 Linux 深度检查的 SSM 关联。此关联会在 Linux 实例上安装 Amazon Inspector SSM 插件。

`InvokeInspectorLinuxSsmPlugin-do-not-delete`

这是 Amazon Inspector 用于 Amazon EC2 Linux 深度检查的 SSM 关联。此关联使 Amazon Inspector 可以使用该插件启动扫描。

> **Note**
>
> 停用 Amazon Inspector Amazon EC2 扫描或深度检查时，将不再调用 SSM 资源 `InvokeInspectorLinuxSsmPlugin-do-not-delete`。

无代理扫描

当您的账户处于混合扫描模式时，Amazon Inspector 会对符合条件的实例使用无代理扫描方法。混合扫描模式包括基于代理的扫描和无代理扫描，当您激活 Amazon EC2 扫描时会自动启用。

对于无代理扫描，Amazon Inspector 使用 EBS 快照从您的实例收集软件清单。无代理扫描会对实例进行扫描，看其是否存在操作系统程序包和应用程序编程语言程序包漏洞。

> **Note**
>
> 扫描 Linux 实例是否存在应用程序编程语言包漏洞时，无代理方法会扫描所有可用路径，而基于代理的扫描仅扫描默认路径和您在基于 Linux 的 Amazon EC2 实例的 Amazon Inspector 深度检查中指定的其他路径。这可能会导致同一个实例有不同的调查发现，具体取决于它是使用基于代理的方法还是使用无代理方法进行扫描。

以下过程说明了 Amazon Inspector 如何使用 EBS 快照收集清单和执行无代理扫描：

1. Amazon Inspector 会为附加到实例的所有卷创建一个 EBS 快照。当 Amazon Inspector 使用它时，快照存储在您的账户中，会使用 `InspectorScan` 标签键进行标记，并使用唯一的扫描 ID 作为标签值。
2. Amazon Inspector 使用 EBS Dire ct 从快照中检索数据， APIs并对快照进行漏洞评估。针对任何检测到的漏洞，都会生成调查发现。
3. Amazon Inspector 会删除它在您的账户中创建的 EBS 快照。

符合条件的实例

如果实例满足以下条件，Amazon Inspector 将使用无代理方法对其进行扫描：

- 该实例具有支持的操作系统。有关更多信息，请参阅the section called “支持的操作系统：Amazon EC2 扫描”的“基于代理的扫描支持”列。
- 该实例的状态为 `Unmanaged EC2 instance`、`Stale inventory` 或 `No inventory`。
- 该实例由 Amazon EBS 支持，具有以下文件系统格式之一：
  - `ext3`
  - `ext4`
  - `xfs`
- 未使用 Amazon EC2 排除标签将该实例排除在扫描范围之外。
- 连接到该实例的卷数量小于 8 个，其总大小小于或等于 1200 GB。

无代理扫描行为

当您的账户配置为混合扫描时，Amazon Inspector 会每 24 小时对符合条件的实例执行一次无代理扫描。Amazon Inspector 每小时都会检测和扫描新的符合条件的实例，其中包括没有 SSM 代理的新实例，或者状态已更改为 `SSM_UNMANAGED` 的现有实例。

每当 Amazon Inspector 在无代理扫描后扫描从实例提取的快照时，都会更新 Amazon EC2 实例的上次扫描时间字段。

您可以通过“账户管理”页面的“实例”选项卡或通过使用 ListCoverage 命令，来查看上次扫描 EC2 实例是否存在漏洞的时间。

管理扫描模式

EC2 扫描模式决定了 Amazon Inspector 在您的账户中执行 EC2 扫描时将使用哪些扫描方法。您可以在常规设置下的 EC2 扫描设置页面查看账户的扫描模式。独立账户或 Amazon Inspector 委派管理员可以更改扫描模式。当您将扫描模式设置为 Amazon Inspector 委派管理员时，系统会为您组织中的所有成员账户设置该扫描模式。Amazon Inspector 具有以下扫描模式：

基于代理的扫描 – 在此扫描模式下，Amazon Inspector 在扫描程序包漏洞时将仅使用基于代理的扫描方法。此扫描模式仅扫描您账户中的 SSM 托管实例，但其好处是可以提供持续扫描，以响应新的 CVE 或对实例的更改。基于代理的扫描还可以为符合条件的实例提供 Amazon Inspector 深度检查。这是新激活账户的默认扫描模式。

混合扫描 – 在此扫描模式下，Amazon Inspector 将结合使用基于代理和无代理的方法来扫描程序包漏洞。对于安装并配置了 SSM 代理的符合条件的 EC2 实例，Amazon Inspector 会使用基于代理的方法。对于不受 SSM 管理的符合条件的实例，Amazon Inspector 将对符合条件且由 EBS 支持的实例使用无代理方法。

更改扫描模式

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用页面右上角的选择 AWS 区域 器，选择要更改 EC2 扫描模式的区域。
3. 在侧面导航面板的常规设置下，选择 EC2 扫描设置。
4. 在扫描模式下，选择编辑。
5. 选择一个扫描模式，然后选择保存更改。

从 Amazon Inspector 扫描中排除实例

可以使用 `InspectorEc2Exclusion` 密钥为 Linux 和 Windows 实例添加标签，从而将其排除在 Amazon Inspector 扫描范围之外。标签密钥不区分大小写。包括标签值是可选的。有关添加标签的信息，请参阅标记 Amazon EC2 资源。

当为实例添加标签以将其排除在 Amazon Inspector 扫描范围之外时，Amazon Inspector 会将该实例标记为已排除，且不会为其创建调查发现。但是，Amazon Inspector SSM 插件将继续被调用。要防止调用该插件，必须允许访问实例元数据中的标签。

> **Note**
>
> 您无需为排除的实例付费。

此外，您可以通过使用标签标记用于加密该卷的 AWS KMS 密钥，将加密的 EBS 卷排除在无代理扫描之外。`InspectorEc2Exclusion`有关更多信息，请参阅标记密钥。

支持的操作系统

Amazon Inspector 会扫描支持的 Mac、Windows 和 Linux 实例中是否存在操作系统包中的漏洞。对于 Linux 实例，Amazon Inspector 可以使用 基于 Linux 的 Amazon EC2 实例的 Amazon Inspector 深度检查 生成应用程序编程语言包的调查发现。对于 Mac 和 Windows 实例，仅扫描操作系统程序包。

有关支持的操作系统的信息，包括无需使用 SSM 代理即可扫描哪些操作系统，请参见Amazon EC2 实例的状态值。

基于 Linux 的 Amazon EC2 实例的 Amazon Inspector 深度检查

Amazon Inspector 将 Amazon EC2 扫描覆盖率扩展至深度检查。借助深度检查，Amazon Inspector 可以检测基于 Linux 的 Amazon EC2 实例中的应用程序编程语言程序包是否存在程序包漏洞。Amazon Inspector 会扫描编程语言包库的默认路径。但是，除了默认情况下 Amazon Inspector 扫描的路径外，您还可以配置自定义路径。

> **Note**
>
> 可以结合“默认主机管理配置”设置来使用深度检查。但是，您必须创建或使用配置了 `ssm:PutInventory` 和 `ssm:GetParameter` 权限的角色。

为了对您基于 Linux 的 Amazon EC2 实例执行深度检查扫描，Amazon Inspector 会使用通过 Amazon Inspector SSM 插件收集的数据。为了管理 Amazon Inspector SSM 插件并对 Linux 执行深度检查，Amazon Inspector 会自动在您的账户中创建 SSM 关联 `InvokeInspectorLinuxSsmPlugin-do-not-delete`。Amazon Inspector 每 6 小时从基于 Linux 的 Amazon EC2 实例收集更新的应用程序清单。

> **Note**
>
> Windows 或 Mac 实例不支持深度检查。

本节介绍了如何管理 Amazon EC2 实例的 Amazon Inspector 深度检查，包括如何为 Amazon Inspector 设置自定义扫描路径。

主题

访问或停用深度检查

> **Note**
>
> 对于在 2023 年 4 月 17 日之后激活 Amazon Inspector 的账户，深度检查将作为 Amazon EC2 扫描的一部分自动激活。

管理深度检查

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v 2/home 中打开 Amazon Inspector 控制台
2. 在导航窗格中，选择常规设置，然后选择 Amazon EC2 扫描设置。
3. 在 Amazon EC2 实例的深度检查下，您可以为您的组织或您自己的账户设置自定义路径。

您可以使用 GetEc2 DeepInspectionConfiguration API 以编程方式检查单个账户的激活状态。您可以使用 BatchGetMemberEc2DeepInspectionStatus API 以编程方式检查多个账户的激活状态。

如果您在 2023 年 4 月 17 日之前激活了 Amazon Inspector，则可以通过控制台横幅或 UpdateEc2DeepInspectionConfiguration API 激活深度检查。如果您是 Amazon Inspector 中某组织的委派管理员，则可以使用 BatchUpdateMemberEc2DeepInspectionStatus API 为自己和成员账户激活深度检查。

您可以通过 UpdateEc2DeepInspectionConfiguration API 停用深度检查。组织中的成员账户无法停用深度检查。因此，成员账户必须由其委派管理员使用 BatchUpdateMemberEc2DeepInspectionStatus API 停用。

Amazon Inspector 深度检查的自定义路径

您可以设置自定义路径，以便 Amazon Inspector 在对 Linux Amazon EC2 实例执行深度检查时进行扫描。当您设置自定义路径时，Amazon Inspector 会扫描该目录及其中的所有子目录中的程序包。

所有账户都可以定义最多 5 个自定义路径。组织的委派管理员可以定义 10 个自定义路径。

Amazon Inspector 除了扫描所有账户的以下默认路径外，还会扫描所有自定义路径：

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

> **Note**
>
> 自定义路径必须是本地路径。Amazon Inspector 不会扫描映射的网络路径，例如网络文件系统挂载或 Amazon S3 文件系统挂载等。

自定义路径的格式设置

自定义路径不能超过 256 个字符。以下是自定义路径外观的示例：

路径示例

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

> **Note**
>
> 每个实例的程序包限制是 5000 个。程序包清单收集时间上限为 15 分钟。Amazon Inspector 建议您选择自定义路径，以规避这些限制。

使用 Amazon Inspector 控制台和 Amazon Inspector API 设置自定义路径

以下步骤描述了如何使用 Amazon Inspector 控制台以及使用 Amazon Inspector API 为 Amazon Inspector 深度检查设置自定义路径。设置自定义路径后，Amazon Inspector 会在下一次深度检查中包含该路径。

Console

1. 以授权管理员身份登录，AWS 管理控制台 然后在 https://console.aws.amazon.com/inspector/v2/home 上打开 Amazon Inspector 控制台
2. 使用 AWS 区域 选择器选择要激活 Lambda 标准扫描的区域。
3. 在导航窗格中，依次选择常规设置和 EC2 扫描设置。
4. 在自己账户的自定义路径下，选择编辑。
5. 在路径文本框中输入自定义路径。
6. 选择保存。

API

运行 UpdateEc2DeepInspectionConfiguration 命令。对于 `packagePaths`，指定要扫描的路径数组。

Amazon Inspector 深度检查的自定义计划

默认情况下，Amazon Inspector 每 6 小时从基于 Amazon EC2 实例收集一次应用程序清单。不过，可以运行以下命令来控制 Amazon Inspector 执行此操作的频率。

命令示例 1：列出关联以查看关联 ID 和当前间隔

以下命令显示关联 `InvokeInspectorLinuxSsmPlugin-do-not-delete` 的关联 ID。

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

命令示例 2：更新关联以包括新的间隔

以下命令使用关联 `InvokeInspectorLinuxSsmPlugin-do-not-delete` 的关联 ID。您可以将 `schedule-expression` 速率从 6 小时设置为新的间隔，例如 12 小时。

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

```
--region your-Region
```

> **Note**
>
> 根据您的使用案例，如果您将 schedule-expression 速率从 6 小时设置为 30 分钟这样的间隔，则可能会超出每日 SSM 清单限制。这会导致结果延迟，并且您可能会遇到 Amazon EC2 实例部分处于错误状态的情况。

支持的编程语言

对于 Linux 实例，Amazon Inspector 深度检查还可以生成应用程序编程语言程序包及操作系统程序包的调查发现。

对于 Mac 和 Windows 实例，Amazon Inspector 深度检查只能生成操作系统程序包的调查发现。

有关支持的编程语言的更多信息，请参阅支持的编程语言：Amazon EC2 深度检查。

使用 Amazon Inspector 扫描 Windows EC2 实例

Amazon Inspector 会自动发现所有支持的 Windows 实例，并将其纳入连续扫描，无需任何额外操作。有关支持哪些实例的信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。Amazon Inspector 会定期运行 Windows 扫描。Windows 实例在发现时进行扫描，然后每 6 小时扫描一次。但是，可以在首次扫描后调整默认扫描间隔。

激活 Amazon EC2 扫描后，Amazon Inspector 会为 Windows 资源创建以下 SSM 关联：InspectorDistributor-do-not-delete、InspectorInventoryCollection-do-not-delete 和 InvokeInspectorSsmPlugin-do-not-delete。若要在 Windows 实例上安装 Amazon Inspector SSM 插件，InspectorDistributor-do-not-delete SSM 关联会使用 AWS-ConfigureAWSPackage SSM 文档和 AmazonInspector2-InspectorSsmPlugin SSM Distributor 程序包。有关更多信息，请参阅适用于 Windows 的 Amazon Inspector SSM 插件。若要收集实例数据并生成 Amazon Inspector 调查发现，InvokeInspectorSsmPlugin-do-not-delete SSM 关联会每 6 小时运行一次 Amazon Inspector SSM 插件。但是，您可以使用 cron 或 rate 表达式自定义此设置。

> **Note**
>
> Amazon Inspector 将更新的开放漏洞和评测语言（OVAL）定义文件暂存到 S3 存储桶 inspector2-oval-prod-*your-AWS-Region*。Amazon S3 存储桶包含扫描中使用的

OVAL 定义。不应修改这些 OVAL 定义。否则，Amazon Inspector 在新版本发布 CVEs 时将不会对其进行扫描。

Windows 实例的 Amazon Inspector 扫描要求

要扫描 Windows 实例，Amazon Inspector 要求实例满足以下条件：

- 该实例是 SSM 托管实例。有关设置扫描实例的说明，请参阅配置 SSM 代理。
- 实例操作系统是支持的 Windows 操作系统之一。有关支持的操作系统类型的完整列表，请参阅Amazon EC2 实例的状态值。
- 该实例安装了 Amazon Inspector SSM 插件。Amazon Inspector 会在发现托管实例时自动为其安装 Amazon Inspector SSM 插件。有关该插件的详细信息，请参阅下一个主题。

> **Note**
>
> 如果主机在 Amazon VPC 中运行，但没有出站互联网访问权限，则 Windows 扫描要求主机能够访问区域 Amazon S3 端点。要了解如何配置 Amazon S3 Amazon VPC 端点，请参阅 Amazon Virtual Private Cloud 用户指南中的创建网关端点。如果您的 Amazon VPC 终端节点策略限制对外部 S3 存储桶的访问，则必须明确允许访问由 Amazon Inspector 维护的存储桶 AWS 区域 ，该存储桶存储用于评估您的实例的 OVAL 定义。此存储桶使用以下格式：`inspector2-oval-prod-REGION`。

为 Windows 实例扫描设置自定义计划

您可以使用 SSM 为 `InvokeInspectorSsmPlugin-do-not-delete` 关联设置 cron 表达式或 rate 表达式，从而自定义 Windows Amazon EC2 实例扫描之间的间隔时间。有关更多信息，请参阅 AWS Systems Manager 用户指南中的参考：适用于 Systems Manager 的 cron 和 rate 表达式，或使用以下说明。

从以下代码示例中选择一个，使用 rate 表达式或 cron 表达式将 Windows 实例的扫描节奏从默认的 6 小时更改为 12 小时。

以下示例要求您使用名AssociationId为的关联`InvokeInspectorSsmPlugin-do-not-delete`。你可以AssociationId通过运行以下 AWS CLI 命令来检索你的：

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

> **Note**
>
> AssociationId是区域性的，因此您需要先为每个区域检索一个唯一的 ID AWS 区域。然后，您可以运行上述命令，在要为 Windows 实例设置自定义扫描计划的每个区域更改扫描节奏。

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 * * ? *)"
```

使用 Amazon Inspector 扫描 Amazon Elastic Container Registry 容器映像

Amazon Inspector 会扫描存储在 Amazon Elastic Container Registry 中的容器映像是否存在软件漏洞，以生成程序包漏洞调查发现。激活 Amazon ECR 扫描后，会将 Amazon Inspector 设置为私有注册表的首选扫描服务。

> **Note**
>
> Amazon ECR 使用注册策略向 AWS 委托人授予权限。该委托人拥有致电 Amazon Inspect APIs or 进行扫描所需的权限。设置注册表策略的范围时，不得在 deny 中添加 `ecr:*` 操作或 `PutRegistryScanningConfiguration`。在启用和禁用 Amazon ECR 扫描时，这样会导致在注册表级别出现错误。

通过基本扫描，可以将存储库配置为在推送时扫描，也可以执行手动扫描。使用增强扫描，可以在注册表级别扫描操作系统和编程语言程序包漏洞。要 side-by-side比较基本扫描和增强扫描之间的区别，请参阅 Amazon Inspector 常见问题解答。

> **Note**
>
> 基本扫描服务通过 Amazon ECR 提供并进行计费。有关更多信息，请参阅 Amazon Elastic Container Registry 定价。增强型扫描通过 Amazon Inspector 提供并进行计费。有关更多信息，请参阅 Amazon Inspector 定价。

有关如何激活 Amazon ECR 扫描的信息，请参阅激活扫描类型。有关如何查看调查发现的信息，请参阅查看 Amazon Inspector 调查发现。有关如何在映像级别查看 Amazon ECR 内的调查发现的信息，请参阅《Amazon Elastic Container Registry 用户指南》中的映像扫描。您可以使用 “ AWS 服务 不可用于基本扫描” 来管理发现，例如AWS Security Hub CSPM 和 Amazon EventBridge。

您可以通过报道页面和，在 Amazon Inspector 中查看每个存储库的扫描配置 APIs。但是，只能在 Amazon ECR 中修改基本扫描与连续扫描的配置设置。Amazon Inspector 提供对这些设置的可见性，但不提供直接修改功能。有关更多信息，请参阅《Amazon ECR 用户指南》中的在 Amazon ECR 中扫描映像是否存在软件漏洞。

本节提供了有关 Amazon ECR 扫描的信息，并介绍了如何为 Amazon ECR 存储库配置增强扫描。

Amazon ECR 扫描的扫描行为

首次激活 Amazon ECR 扫描时，Amazon Inspector 会检测过去 14 天内推送的映像。然后，Amazon Inspector 会扫描这些映像，并将扫描状态设置为 `ACTIVE`。Amazon Inspector 将仅扫描 ECR 中处于活动状态的图像（`imageStatus`字段为`ACTIVE`）。Amazon Inspector 不会扫描状态为 ECR（`imageStatus`字段为`ARCHIVED`）的图片。

如果启用连续扫描，Amazon Inspector 将监控图像在 14 天内（默认）、 last-in-use日期在 14 天内（默认）或在配置的重新扫描持续时间内扫描图像。对于 2025 年 5 月 16 日之前创建的 Amazon Inspector 账户，默认配置是重新扫描以监控映像是否在过去 90 天内被推送或拉取。有关更多信息，请参阅配置 Amazon ECR 重新扫描持续时间。

对于连续扫描，Amazon Inspector 会对以下情况中的容器映像启动新的漏洞扫描：

- 每当推送新的容器映像时。
- 每当 Amazon Inspector 在其数据库中添加新的常见漏洞和风险（CVE）项目，并且 CVE 与该容器映像相关时（仅限持续扫描）。

- 每当容器镜像在 ECR 中从存档转换为活动镜像时。

如果您将存储库配置为推送扫描，则只有在推送映像时才会对其进行扫描。

您可以通过账户管理页面的容器映像选项卡或使用 ListCoverage API 来查看上次检查容器映像是否存在漏洞的时间。发生以下事件时，Amazon Inspector 会更新 Amazon ECR 映像的上次扫描时间字段：

- Amazon Inspector 完成对容器映像的初始扫描时。
- 由于 Amazon Inspector 数据库中添加了影响该容器映像的新的常见漏洞和风险（CVE）项目，因此 Amazon Inspector 重新扫描容器映像时。

已存档的 ECR 容器镜像

Amazon Inspector 不扫描存档在 ECR (`imageStatusisARCHIVED`) 中的容器图像。当 ECR 中的活动图像转换为存档图像时，Amazon Inspector 会自动关闭调查结果，然后在 3 天后删除调查结果。如果存档的容器镜像在 ECR 中变为活动镜像，Amazon Inspector 会触发新的扫描。

将容器映像映射到正在运行的容器

Amazon Inspector 通过将容器映像映射到 Amazon Elastic Container Service（Amazon ECS）和 Amazon Elastic Kubernetes Service（Amazon EKS）中正在运行的容器，来提供全面的容器安全管理。这些映射可让您深入了解正在运行的容器上映像的漏洞。

> **Note**
>
> 仅使用托管式策略 `AWSReadOnlyAccess` 无法提供查看 Amazon ECR 映像与正在运行的容器之间映射的足够权限。要查看容器映像映射信息，您需要同时使用 `AWSReadOnlyAccess` 和 `AWSInspector2ReadOnlyAccess` 托管式策略。

您可以根据运营风险确定修复工作的优先级，并在整个容器生态系统中保持安全覆盖范围。您可以查看当前正在使用的容器映像数量，以及过去 24 小时内在 Amazon ECS 或 Amazon EKS 集群上最后使用的具体容器映像。您还可以查看部署了多少 Amazon ECS 任务和 Amazon EKS 容器组（pod）。可以在 Amazon Inspector 控制台的容器映像调查发现的详细信息屏幕上并对 `FilterCriteria` 数据类型使用 `ecrImageInUseCount` 和 `ecrImageLastInUseAt` 筛选条件来找到此信息。对于新的容器映像或账户，数据可能需要长达 36 小时才能使用。之后，此数据每 24 小时更新一次。有关更多信息，请参阅查看 Amazon Inspector 调查发现和查看 Amazon Inspector 调查发现的详细信息。

> **Note**
>
> 激活 Amazon ECR 扫描并将存储库配置为连续扫描时，会将这些数据自动发送到 Amazon ECR 调查发现。必须在 Amazon ECR 存储库级别配置连续扫描。有关更多信息，请参阅《Amazon Elastic Container Registry 用户指南》中的增强扫描。

您也可以根据集群中的容器镜像的 last-in-use日期重新扫描这些镜像。

搭载 Amazon ECS 和 Amazon EKS 的 Fargate 也支持此功能。

支持的操作系统和媒体类型

有关支持的操作系统的信息，请参阅支持的操作系统：使用 Amazon Inspector 执行 Amazon ECR 扫描。

Amazon Inspector 对 Amazon ECR 存储库的扫描涵盖以下支持的媒体类型：

映像清单

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

映像配置

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

映像层

- "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 不支持使用 `"application/vnd.docker.distribution.manifest.list.v2+json"` 媒体类型来扫描 Amazon ECR 存储库。

配置 Amazon ECR 重新扫描持续时间

Amazon ECR 重新扫描持续时间设置决定了 Amazon Inspector 持续监控存储库中的容器映像的时间。您可以为图像 last-in-use日期、上次提取日期和推送日期配置重新扫描持续时间。最佳做法是，根据您的环境配置最合适的重新扫描持续时间。

如果您经常构建映像，请选择较短的扫描持续时间。对于长时间使用的映像，请选择更长的扫描持续时间。新帐户（包括添加到组织中的新帐户）的默认扫描持续时间为 14 天。

只要映像在集群上的上次使用或推送时间在 14 天内（默认情况下），Amazon Inspector 就会继续监控和重新扫描映像。如果映像在配置的推送日期和上次使用日期内未被推送或未在运行的容器上使用，则 Amazon Inspector 将停止对其进行监控。如果需要，可以选择将设置更改为按上次拉取日期监控映像，而不是按上次使用日期监控映像。当 Amazon Inspector 停止监控映像时，它会将映像扫描状态代码设置为非活动，并将原因代码设置为已过期。然后，Amazon Inspector 会计划关闭所有相关的映像调查发现。

例如，如果延长拉取日期持续时间，Amazon Inspector 会将更改应用于配置为持续扫描的存储库中所有正在主动扫描的映像。但是，即使您在新的持续时间内推送了非活动映像，它们仍处于非活动状态。

通过委派管理员账户配置重新扫描持续时间时，Amazon Inspector 会将该设置应用于组织中的所有成员账户。如果委派管理员账户未启用 Amazon ECR 扫描，则无法查看 API 映像的集群。

对于多架构映像，不支持 last-in-use日期跟踪。使用多架构映像时，我们建议您根据图像拉取或推送事件而不是 last-in-use日期来配置扫描，以确保正确的重新扫描行为。

> **Note**
>
> 2025 年 5 月 16 日之前配置的所有重新扫描持续时间设置将保持不变。您可以继续使用先前配置的任何默认设置。

映像重新扫描持续时间

映像重新扫描持续时间决定了 Amazon Inspector 监控映像的时间长度。图像重新扫描持续时间包括两种模式：上次使用日期（默认）或上次提取日期。如果您想使用 Amazon E ECS/Amazon KS 集群活动中的上次使用日期，请选择上次使用日期（默认）。如果您想使用 Amazon ECR 映像的上次拉取日期来重新扫描映像，请选择上次拉取日期。可选择以下重新扫描持续时间选项：

- 14 天（默认）
- 30 天
- 60 天
- 90 天
- 180 天

映像推送日期持续时间

映像推送日期持续时间决定了 Amazon Inspector 在将映像推送到存储库之后会持续监控映像的时间长度。可选择以下重新扫描持续时间选项：

- 14 天（默认）
- 30 天
- 60 天
- 90 天
- 180 天
- 生命周期

配置 Amazon ECR 重新扫描持续时间

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 选择您要配置 Amazon ECR 重新扫描持续时间 AWS 区域 的位置。
3. 在导航窗格中，依次选择常规设置和 ECR 扫描设置。
4. 在 ECR 重新扫描持续时间下，选择映像重新扫描模式，然后选择相应的持续时间。
5. 在映像推送日期下，选择映像推送日期。
6. 选择保存。

了解 ECR 容器镜像状态

Inspector 仅扫描 ECR 容器镜像中的`ACTIVE`图像。不扫描处于`ARCHIVED`状态的 ECR 容器映像。要了解有关扫描行为的更多信息，请参阅Amazon ECR 扫描的扫描行为。

当 ECR 容器镜像在 ECR 中的图像状态转换为，`ACTIVE`Inspector 会使用该`lastActivatedAt`字段来监控重新扫描的持续时间。

使用 Amazon Inspector 进行扫描 AWS Lambda

Amazon Inspector 对 AWS Lambda 功能和层的支持可提供持续的自动安全漏洞评估。Amazon Inspector 提供两种类型的 Lambda 函数扫描。

Amazon Inspector Lambda 标准扫描

此扫描类型是默认的 Lambda 扫描类型。它会扫描 Lambda 函数和层中的应用程序依赖关系，以查找程序包漏洞。

Amazon Inspector Lambda 代码扫描

这种扫描类型会扫描 Lambda 函数及层中的自定义应用程序代码，以查找代码漏洞。您可以激活 Lambda 标准扫描，也可以同时激活 Lambda 标准扫描和 Lambda 代码扫描。

如果要激活 Lambda 代码扫描，则必须先激活 Lambda 代码扫描。有关更多信息，请参阅激活扫描类型。

激活 Lambda 函数扫描后，Amazon Inspector 会在您的账户中创建以下服务相关通道：`cloudtrail:CreateServiceLinkedChannel` 和 `cloudtrail:DeleteServiceLinkedChannel`。Amazon Inspector 管理这些渠道，并使用它们来监控扫描 CloudTrail 事件。这些频道允许您查看账户中的 CloudTrail 事件，就像有跟踪一样 CloudTrail。我们建议您在中创建自己的跟踪 CloudTrail 以管理您账户中的事件。有关如何查看这些通道的信息，请参阅《AWS CloudTrail 用户指南》中的查看服务相关通道。

> **Note**
>
> Amazon Inspector 不支持扫描使用客户托管密钥加密的 Lambda 函数。这一点适用于 Lambda 标准扫描和 Lambda 代码扫描。

Lambda 函数扫描的扫描行为

激活后，Amazon Inspector 会扫描您账户中过去 90 天内调用或更新的所有 Lambda 函数。在以下情况下，Amazon Inspector 会对 Lambda 函数启动漏洞扫描：

- Amazon Inspector 发现现有 Lambda 函数时。
- 将新的 Lambda 函数部署到 Lambda 服务时。
- 部署现有 Lambda 函数或其层的应用程序代码或依赖项更新时。
- Amazon Inspector 在其数据库中添加新的常见漏洞和风险（CVE）项目，且该 CVE 与您的函数相关时。

Amazon Inspector 会在每个 Lambda 函数的整个生命周期内对其进行监控，直到该函数被删除或被排除在扫描范围之外。

您可以通过账户管理页面的 Lambda 函数选项卡或使用 ListCoverage API 来查看上次检查 Lambda 函数是否存在漏洞的时间。发生以下事件时，Amazon Inspector 会更新 Lambda 函数的上次扫描时间字段：

- Amazon Inspector 完成对 Lambda 函数的初始扫描时。
- 更新 Lambda 函数时。
- 由于影响 Lambda 函数的新 CVE 项目添加到 Amazon Inspector 数据库中，Amazon Inspector 重新扫描该函数时。

支持的运行时系统和符合条件的函数

对于 Lambda 标准扫描和 Lambda 代码扫描，Amazon Inspector 支持不同的运行时系统。有关每种扫描类型支持的运行时系统的列表，请参阅支持的运行时系统：Amazon Inspector Lambda 标准扫描和支持的运行时系统：Amazon Inspector Lambda 代码扫描。

除了具有受支持的运行时系统之外，Lambda 函数还需要满足以下条件才有资格进行 Amazon Inspector 扫描：

- 过去 90 天内调用或更新过该函数。
- 该函数被标记为 $LATEST。
- 该函数未按标签从扫描中排除。

> **Note**
>
> 过去 90 天内未调用或修改的 Lambda 函数将自动排除在扫描范围之外。如果 Lambda 函数再次被调用或对函数代码进行了更改，则 Amazon Inspector 将恢复对自动排除的函数的扫描。

Amazon Inspector Lambda 标准扫描

Amazon Inspector Lambda 标准扫描可识别您添加到 Lambda 函数代码和层的应用程序包依赖项中的软件漏洞。例如，如果 Lambda 函数使用的 `python-jwt` 程序包版本存在已知脆弱性，则 Lambda 标准扫描将生成针对该函数的调查发现。

如果 Amazon Inspector 在 Lambda 函数应用程序包依赖项中检测到脆弱性，则 Amazon Inspector 会生成详细的程序包脆弱性类型的调查发现。

有关激活扫描类型的说明，请参阅激活扫描类型。

> **Note**
>
> Lambda 标准扫描不会扫描 Lambda 运行时环境中默认安装的 AWS SDK 依赖项。Amazon Inspector 仅扫描使用函数代码上传的依赖项或从层继承的依赖项。

> **Note**
>
> 停用 Amazon Inspector Lambda 标准扫描也将同时停用 Amazon Inspector Lambda 代码扫描。

从 Lambda 标准扫描中排除函数

您可以向 Lambda 函数添加标签，从而将其排除在 Amazon Inspector Lambda 标准扫描之外。从扫描中排除函数能够防止出现无法操作的警报。向要排除的函数添加标签时，该标签必须具有以下键值对。

- 键：`InspectorExclusion`
- 值：`LambdaStandardScanning`

本主题介绍如何向要从扫描中排除的函数添加标签。有关在 Lambda 中添加标签的更多信息，请参阅在 Lambda 函数上使用标签。

从扫描中排除函数

1. 使用您的凭证登录，然后打开 Lambda 控制台，网址为：https://console.aws.amazon.com/lambda/。
2. 从导航窗格中选择函数。
3. 选择要排除在 Amazon Inspector Lambda 标准扫描之外的函数的名称。
4. 选择 Configuration（配置），然后选择 Tags（标签）。
5. 选择管理标签，然后选择添加新标签。
   a. 对于键，输入 `InspectorExclusion`。
   b. 对于 Value（值），输入 `LambdaStandardScanning`
6. 选择保存。

Amazon Inspector Lambda 代码扫描

> **Important**
>
> 该特征可捕获 Lambda 函数的代码片段，以突出显示检测到的漏洞。这些片段可能显示硬编码的凭证或其他敏感材料。

借助此特征，Amazon Inspector 可根据 AWS 安全最佳实践扫描 Lambda 函数中的应用程序代码是否存在代码漏洞，检测数据泄露、注入缺陷、缺少加密和弱加密等。Amazon Inspector 会使用自动推理和机器学习来评估 Lambda 函数应用程序代码。它还会使用与 Amazon Q 合作开发的内部检测器来识别违反政策的行为和漏洞。

Amazon Inspector 在 Lambda 函数应用程序代码中检测到漏洞时，会生成代码漏洞。此调查发现类型包括展示问题的代码片段，以及问题在代码中的具体位置。此外，它还建议如何修复问题。建议包括即插即用代码块，用于替换易受攻击的代码行。除了针对该调查发现类型的一般代码补救指南外，还提供了这些代码修复。

代码修复建议由自动推理提供支持。某些代码补救建议可能无法按预期起作用。您应对自己采纳的代码补救建议负责。在采纳代码补救建议之前，请务必仔细审视这些建议。您可能需要对其进行编辑，以确保代码符合您的预期。有关更多信息，请参阅负责任的人工智能政策。

如果要激活 Lambda 代码扫描，则必须先激活 Lambda 代码扫描。有关更多信息，请参阅激活扫描类型。有关哪些 AWS 区域支持该特征的信息，请参阅特定于区域的特征可用性。

在代码漏洞调查发现中加密代码

Amazon Q 会存储使用 Lambda 代码扫描检测到的与代码漏洞调查发现相关的代码片段。默认情况下，Amazon Q 控制用于加密代码的 AWS 拥有的密钥。但是，您可以通过 Amazon Inspector API 使用自己的客户自主管理型密钥进行加密。有关更多信息，请参阅对调查发现中的代码进行静态加密。

从 Lambda 代码扫描中排除函数

您可以向 Lambda 函数添加标签，从而将其排除在 Amazon Inspector Lambda 代码扫描之外。从扫描中排除函数能够防止出现无法操作的警报。向要排除的函数添加标签时，该标签必须具有以下键值对。

- 密钥 – `InspectorCodeExclusion`
- 值 – `LambdaCodeScanning`

本主题介绍如何向要从代码扫描中排除的函数添加标签。有关在 Lambda 中添加标签的更多信息，请参阅在 Lambda 函数上使用标签。

从代码扫描中排除函数

1. 使用您的凭证登录，然后打开 Lambda 控制台，网址为：https://console.aws.amazon.com/lambda/。
2. 从导航窗格中选择函数。
3. 选择要排除在 Amazon Inspector Lambda 代码扫描之外的函数的名称。
4. 选择 Configuration（配置），然后选择 Tags（标签）。
5. 选择管理标签，然后选择添加新标签。
   a. 对于键，输入 `InspectorCodeExclusion`。
   b. 对于 Value（值），输入 `LambdaCodeScanning`
6. 选择保存。

在 Amazon Inspector 中停用扫描类型

停用某扫描类型后，您将无法访问该扫描类型生成的任何调查发现。如果您重新激活扫描类型，Amazon Inspector 会扫描所有符合条件的资源来生成新的调查发现。如果要记录调查发现，可以

通过调查发现报告的形式，将调查发现导出到 Amazon Simple Storage Service（Amazon S3）存储桶。有关更多信息，请参阅 导出 Amazon Inspector 调查发现报告。停用扫描类型时，停用扫描类型的 AWS 帐户可能会遇到以下变化：

Amazon EC2 扫描

为账户停用 Amazon Inspector Amazon EC2 扫描时，以下 SSM 关联将被删除：

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

此外，已从所有 Windows 主机中移除 Amazon Inspector SSM 插件。有关更多信息，请参阅 扫描 Windows EC2 实例。

Amazon ECR 扫描

为账户停用 Amazon ECR 扫描后，该账户的 Amazon ECR 扫描类型将从使用 Amazon Inspector 进行增强扫描更改为使用 Amazon ECR 进行基本扫描。

Lambda 标准扫描

为账户停用 Lambda 标准扫描后，如果扫描类型已激活，则会停用 Lambda 代码扫描。您还可以删除 Amazon Inspector 在激活 Lambda 标准扫描时创建的 CloudTrail 服务相关渠道。

Amazon Inspector 代码安全性

当您为账户停用代码安全时，会删除与之相关联的所有集成、项目和扫描配置。如果您的账户是组织的委派管理员，则您只会停用自己账户的代码安全，而成员账户将变为独立账户。

停用扫描

停用某个账户的所有扫描类型会在 AWS 区域停用该账户的 Amazon Inspector。有关更多信息，请参阅 停用 Amazon Inspector。

要在多账户环境中完成此步骤，请在以 Amazon Inspector 委托管理员身份登录后完成以下步骤。

Console

停用扫描

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用页面右上角的 AWS 区域 选择器，选择要停用扫描的区域。
3. 在导航窗格中，选择账户管理。
4. 选择账户选项卡以显示账户的扫描状态。
5. 选中要停用扫描的每个账户对应的复选框。
6. 选择操作，然后从停用选项中选择要停用的扫描类型。
7. （推荐）在要停 AWS 区域 用该扫描类型的每个扫描类型中重复这些步骤。

API

运行禁用 API 操作。在请求中，提供 IDs 您要停用扫描的帐户，并`resourceTypes`提供一个或多个`EC2`、`ECR`、`LAMBDA`、或`LAMBDA_CODE`以停用扫描。

Amazon EC2 实例操作系统的 Center for Internet Security（CIS）扫描

Amazon Inspector CIS 扫描（CIS 扫描）可对您的 Amazon EC2 实例操作系统进行基准测试，确保您根据 Center for Internet Security 确定的最佳实践建议对其进行配置。CIS 安全基准提供了用于安全配置系统的行业标准配置基准和最佳实践。在为账户启用 Amazon Inspector EC2 扫描后，您可以执行或计划 CIS 扫描。有关如何激活 Amazon EC2 扫描的信息，请参阅激活扫描类型。

> **Note**
>
> CIS 标准适用于 x86_64 操作系统。在基于 ARM 的资源上，某些检查可能无法进行评估或返回无效修复说明。

Amazon Inspector 根据实例标签以及您定义的扫描计划对目标 Amazon EC2 实例执行 CIS 扫描。Amazon Inspector 会对每个目标实例执行一系列实例检查。每项检查都会评估您的系统配置是否符合特定的 CIS 基准建议。每项检查都有一个 CIS 检查 ID 和标题，与该平台的 CIS 基准建议对应。CIS 扫描完成后，您可以查看结果来了解该系统的哪些实例检查通过、跳过或失败。

> **Note**
>
> 要执行或计划 CIS 扫描，必须有安全的互联网连接。但是，如果要对私有实例运行 CIS 扫描，则必须使用 VPC 端点。

主题

Amazon Inspector CIS 扫描的 Amazon EC2 实例要求

要在您的 Amazon EC2 实例上运行 CIS 扫描，Amazon EC2 实例必须满足以下条件：

- 实例操作系统是 CIS 扫描支持的操作系统之一。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。
- 实例是 Amazon EC2 Systems Manager 实例。有关更多信息，请参阅《AWS Systems Manager 用户指南》中的使用 SSM Agent。
- 该实例已安装 Amazon Inspector SSM 插件。Amazon Inspector 会自动在托管实例上安装此插件。
- 该实例具有实例配置文件，该配置文件授予 SSM 管理实例的权限，并授予 Amazon Inspector 对实例运行 CIS 扫描的权限。要授予这些权限，请将 Amazon SSMManaged InstanceCore 和 AmazonInspector2 个ManagedCisPolicy策略附加到一个 IAM 角色。然后将 IAM 角色作为实例配置文件附加到该实例。有关创建和附加实例配置文件的说明，请参阅《Amazon EC2 用户指南》中的使用 IAM 角色。

> **Note**
>
> 在 Amazon EC2 实例上运行 CIS 扫描之前，您无需启用 Amazon Inspector 深度检查。如果您禁用 Amazon Inspector 深度检查，Amazon Inspector 会自动安装 SSM Agent，但不会再调用 SSM Agent 来运行深度检查。不过，因此，您的账户中存在 `InspectorLinuxDistributor-do-not-delete` 关联。

在私有 Amazon EC2 实例上运行 CIS 扫描的 Amazon Virtual Private Cloud 端点要求

您可以通过 Amazon 网络对 Amazon EC2 实例运行 CIS 扫描。但是，如果您想在私有 Amazon EC2 实例上运行 CIS 扫描，则必须创建 Amazon VPC 端点。为 Systems Manager 创建 Amazon VPC 端点时，需要以下端点：

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

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的为 Systems Manager 创建 VPC 端点。

> **Note**
>
> 目前，有些 AWS 区域 不支持该amazonaws.com.*region*.inspector2端点。

运行 CIS 扫描

您可以按需运行一次性 CIS 扫描，也可以按计划重复扫描。要运行扫描，请先创建扫描配置。

创建扫描配置时，您可以指定用于查找实例的标签键值对。如果您是组织的 Amazon Inspector 委派管理员，则可以在扫描配置中指定多个账户，Amazon Inspector 将在每个账户中查找带有指定标签的实例。您可以为扫描选择 CIS 基准等级。对于每个基准测试，CIS 都支持等级 1 和等级 2 配置文件，旨在为不同环境可能需要的不同安全等级提供基准。

- 等级 1 - 建议可在任何系统上配置的基本安全设置。实施这些设置会让服务很少中断或根本不会中断。这些建议的目标是减少系统入口点的数量，从而降低整体网络安全风险。
- 等级 2 - 为高安全性环境建议更高级的安全设置。实施这些设置需要进行规划和协调，以最大限度地降低业务影响的风险。这些建议的目标是协助您符合监管合规性要求。

等级 2 是等级 1 的延伸。如果选择等级 2，Amazon Inspector 会检查针对等级 1 和等级 2 建议的所有配置。

定义扫描参数后，您可以选择是将其作为一次性扫描来运行（在完成配置后便运行），还是作为重复扫描来运行。重复扫描可以每天、每周或每月运行，时间由您选择。

> **Tip**
>
> 我们建议选择扫描运行期间不太可能影响系统的日期和时间。

使用管理 Amazon Inspector CIS 扫描的注意事项 AWS Organizations

当您在组织中运行 CIS 扫描时，Amazon Inspector 委派管理员和成员账户会以不同的方式与 CIS 扫描配置和扫描结果进行交互。

Amazon Inspector 委派管理员如何与 CIS 扫描配置和扫描结果进行交互

当委派管理员为所有账户或特定成员账户创建扫描配置时，该配置归组织所有。组织拥有的扫描配置有一个 ARN，将组织 ID 指定为所有者：

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

委派管理员可以管理组织拥有的扫描配置，即使这些配置是由其他账户创建。

委派管理员可以查看其组织中任何账户的扫描结果。

如果委派管理员创建了扫描配置并指定 SELF 为目标账户，则即使该委派管理员离职，他们依然拥有扫描配置。但是，委派管理员无法以 SELF 为目标来更改扫描配置的目标。

> **Note**
>
> 委派管理员无法向组织拥有的 CIS 扫描配置添加标签。

Amazon Inspector 成员账户如何与 CIS 扫描配置和扫描结果进行交互

当成员账户创建 CIS 扫描配置时，该配置归其所有。但是，委派管理员可以查看该配置。如果成员账户离职，则委派管理员将无法查看该配置。

> **Note**
>
> 委派管理员无法编辑成员账户创建的扫描配置。

成员账户、以 SELF 为目标的委派管理员，以及独立账户都拥有自己创建的扫描配置。这些扫描配置有一个 ARN，将账户 ID 显示为所有者：

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

成员账户可以在其账户中查看扫描结果，包括委派管理员计划的 CIS 扫描的扫描结果。

Amazon Inspector 拥有用于 Amazon Inspector CIS 扫描的 Amazon S3 存储桶

开放式漏洞和评估语言（OVAL，Open Vulnerability and Assessment Language）是一项信息安全工作，旨在使评测和报告计算机系统的机器状态实现标准化。下表列出了 Amazon Inspector 拥有的所有用于 CIS 扫描的 Amazon S3 存储桶及 OVAL 定义。Amazon Inspector 会暂存 CIS 扫描所需的 OVAL 定义文件。 VPCs 如有必要，应将亚马逊 Inspector 拥有的 Amazon S3 存储桶列入许可名单。

> **Note**
>
> 以下 Amazon Inspector 拥有的每个 Amazon S3 存储桶的详细信息都不会发生变化。但是，该表可能会不定期更新，以反映新支持的 AWS 区域。您不能将 Amazon Inspector 拥有的 Amazon S3 存储桶用于其他 Amazon S3 操作或在您自己的 Amazon S3 存储桶中使用。

CIS 存储桶	AWS 区域
`cis-datasets-prod-arn-5908f6f`	欧洲地区（斯德哥尔摩）
`cis-datasets-prod-bah-8f88801`	中东（巴林）
`cis-datasets-prod-bjs-0f40506`	中国（北京）
`cis-datasets-prod-bom-435a167`	亚太地区（孟买）
`cis-datasets-prod-cdg-f3a9c58`	欧洲地区（巴黎）
`cis-datasets-prod-cgk-09eb12f`	亚太地区（雅加达）
`cis-datasets-prod-cmh-63030b9`	美国东部（俄亥俄州）
`cis-datasets-prod-cpt-02c5c6f`	非洲（开普敦）
`cis-datasets-prod-dub-984936f`	欧洲地区（爱尔兰）
`cis-datasets-prod-fra-6eb96eb`	欧洲地区（法兰克福）
`cis-datasets-prod-gru-de69f99`	南美洲（圣保罗）

CIS 存储桶	AWS 区域
cis-datasets-prod-hkg-8e30800	亚太地区（香港）
cis-datasets-prod-iad-8438411	美国东部（弗吉尼亚州北部）
cis-datasets-prod-icn-f4eff1c	亚太地区（首尔）
cis-datasets-prod-kix-5743b21	亚太地区（大阪）
cis-datasets-prod-lhr-8b1fbd0	欧洲地区（伦敦）
cis-datasets-prod-mxp-7b1bbce	欧洲地区（米兰）
cis-datasets-prod-nrt-464f684	亚太地区（东京）
cis-datasets-prod-osu-5bead6f	AWS GovCloud （美国东部）
cis-datasets-prod-pdt-adadf9c	AWS GovCloud （美国西部）
cis-datasets-prod-pdx-acfb052	美国西部（俄勒冈州）
cis-datasets-prod-sfo-1515ba8	美国西部（北加利福尼亚）
cis-datasets-prod-sin-309725b	亚太地区（新加坡）
cis-datasets-prod-syd-f349107	亚太地区（悉尼）
cis-datasets-prod-yul-5e0c95e	加拿大（中部）
cis-datasets-prod-zhy-5a8eacb	中国（宁夏）
cis-datasets-prod-zrh-67e0e3d	欧洲（苏黎世）

创建 CIS 扫描配置

本主题介绍如何创建 CIS 扫描配置。

运行 CIS 扫描

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用 AWS 区域 下拉列表选择要运行 CIS 扫描的 AWS 区域。
3. 在导航窗格中，选择按需扫描，然后选择 CIS 扫描。
4. 选择创建新扫描。
5. 在扫描配置名称中，输入扫描配置名称。
6. 在目标资源标签中，输入要扫描的实例的键和相应的值。您最多可以为每个键指定五个不同的值，总共可以指定 25 个要包含在扫描中的标签。
7. 对于 CIS 基准等级，您可以为基本安全配置选择等级 1，为高级安全配置选择等级 2。
8. 对于目标账户，请指定要包含在 CIS 扫描中的账户。有关更多信息，请参阅 使用管理 Amazon Inspector CIS 扫描的注意事项 AWS Organizations。

   如果您的账户是委派管理员账户，则可以选择所有账户或指定账户。所有账户选项将查找组织中的所有账户。指定账户则仅查找组织中的个别账户。如果选择此选项，则可以通过用逗号分隔账户来指定多个账户。您也可以输入 SELF 而不是账户 ID 来为您的账户创建扫描配置

   如果您的账户是组织中的独立账户或者成员账户，您可以选择自主来为您的账户创建扫描配置。
9. 对于计划，选择一次性扫描，这样将在您完成扫描配置创建后立即运行扫描，或者选择重复扫描，这样将在您指定的时间运行扫描。
10. 确认选择后，选择创建。

查看 CIS 扫描结果

Amazon Inspector 会为运行的每个扫描配置创建一个扫描作业，并使用唯一的扫描 ID 收集扫描结果。CIS 扫描结果在 90 天内可用。您可以通过检查或扫描的资源查看 CIS 扫描结果：

- 按检查汇总的扫描结果 - 按扫描期间执行的各项检查对扫描结果进行分组。对于每项检查，您都会收到一份报告，说明有多少资源失败、跳过或通过。
- 按扫描的资源汇总的扫描结果 - 按扫描期间扫描查找的每个扫描资源对扫描结果进行分组。对于每种资源，您都会收到一份报告，说明资源有多少次检查失败、跳过或通过。

本主题介绍如何查看 CIS 扫描结果。

查看扫描结果

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用 AWS 区域 下拉列表选择创建 CIS 扫描配置的 AWS 区域。
3. 在导航窗格中，选择按需扫描，然后选择 CIS 扫描。
4. 选择扫描结果选项卡。
5. 在计划者列下，选择要查看的扫描计划 ID。或者选择要查看的扫描计划 ID 所在的行，然后选择查看详细信息。
6. 选择检查可查看执行的每项检查，或者选择扫描的资源可查看扫描期间查找的每个扫描的资源。

您还可以查看计划的 CIS 扫描的详细信息。

查看计划的 CIS 扫描的详细信息

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用 AWS 区域 下拉列表选择创建 CIS 扫描配置的 AWS 区域。
3. 在导航窗格中，选择按需扫描，然后选择 CIS 扫描。
4. 选择已计划选项卡。
5. 在扫描配置名称列下，选择要查看的扫描配置的名称。或者选择要查看的扫描配置所在的行，然后选择查看详细信息。

编辑 CIS 扫描配置

本主题介绍如何编辑 CIS 扫描配置。

编辑 CIS 扫描配置

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用 AWS 区域 下拉列表选择创建 CIS 扫描配置的 AWS 区域。
3. 在导航窗格中，选择按需扫描，然后选择 CIS 扫描。
4. 选择已计划选项卡。
5. 选择要编辑的配置所在的行，然后选择编辑。

下载 CIS 扫描结果

可使用 Amazon Inspector 控制台或 API 下载 PDF 或 CSV 格式的 CIS 扫描。

> **Note**
>
> 对于在 2024 年 5 月 3 日之后收集的 CIS 扫描，您只能下载 CSV 格式的 CIS 扫描结果。

本主题介绍如何使用 Amazon Inspector 控制台下载 CIS 扫描。

通过控制台下载 CIS 扫描结果

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用 AWS 区域 下拉列表选择创建 CIS 扫描配置的 AWS 区域。
3. 在导航窗格中，选择按需扫描，然后选择 CIS 扫描。
4. 选择扫描结果选项卡。
5. 在计划者列下，选择要查看的扫描计划 ID。或者选择要查看的扫描计划 ID 所在的行，然后选择查看详细信息。
6. 选择下载，然后选择 PDF 或 CSV。如果您的账户是委派管理员账户，则可以选择选择账户来下载特定成员账户的结果。

Amazon Inspector Code Security

Amazon Inspector 是一项漏洞管理服务，可自动发现工作负载并持续对其进行扫描，以查找软件漏洞和意外的网络暴露。借助 Code Security，Amazon Inspector 会扫描第一方应用程序源代码、第三方应用程序依赖关系以及“基础设施即代码”中的漏洞。您可以在 Amazon Inspector 控制台中或通过 Amazon Inspector API 激活 Code Security。激活 Code Security 后，可以创建扫描配置并将其应用于您的代码存储库，以确定其扫描的频率和时间。您可以随时查看、编辑和删除扫描配置。有关提供 Code Security 的 AWS 区域的信息，请参阅区域和端点。有关定价的信息，请参阅 Amazon Inspector 定价。

代码安全的先决条件

在开始使用代码安全之前，您必须先激活代码安全并决定如何加密您的数据。此信息可以是集成凭证、代码等信息，也可以是与集成、代码存储库和项目相关的任何其他信息。默认情况下，您的数据使用 AWS 拥有的密钥进行加密。这意味着密钥由相应服务创建、拥有和管理。如果要拥有和管理用于加密数据的密钥，可以创建客户托管的 KMS 密钥。

激活 Code Security

激活 Code Security 的方式与激活所有自动扫描类型的方式相同。有关更多信息，请参阅激活扫描类型。

创建客户管理的密钥进行访问 AWS KMS

默认情况下，您的数据使用 AWS 拥有的密钥进行加密。这意味着密钥由相应服务创建、拥有和管理。如果要拥有和管理用于加密数据的密钥，可以创建客户托管的 KMS 密钥。Amazon Inspector 不会与您的数据进行交互。Amazon Inspector 仅从您的源代码提供商的存储库中摄取元数据。有关如何创建客户托管 KMS 密钥的信息，请参阅《AWS Key Management Service 用户指南》中的创建 KMS 密钥。

策略示例

创建客户托管密钥时，使用以下策略示例。

> **Note**
>
> 以下策略中的 FAS 权限特定于 Amazon Inspector，因为它们仅允许 Amazon Inspector 执行那些 API 调用。

JSON

```
{
  "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:us-east-1:111122223333: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.us-east-1.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.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

创建 KMS 密钥后，您可以使用以下 Amazon Inspector APIs。

- UpdateEncryptionKey — 与 f CODE_REPOSITORY or `resourceType` 和一起使用CODE作为扫描类型，配置客户托管 KMS 密钥的使用。
- GetEncryptionKey — 与 f CODE_REPOSITORY or `resourceType` 和一起使用CODE作为扫描类型来配置 KMS 密钥配置的检索。
- ResetEncryptionKey — 与 for `resourceType` 和 CODE_REPOSITORY for 一起使用CODE可重置您的 KMS 密钥配置并使用 AWS 拥有的 KMS 密钥。

创建 Amazon Inspector 与您的代码存储库之间的集成

本节包括介绍如何在 Amazon Inspector 与您的代码存储库之间创建集成的主题。创建集成时，所有代码存储库都会在 Amazon Inspector 控制台的 Code Security 页面上作为项目列出。本节中的其他主题介绍了如何访问您的集成和项目。

Code Security 最多只能导入 10 万个项目，并且仅监控每个存储库的默认分支。一个项目最多可以与三个默认扫描配置关联。

Code Security 每个账户仅支持最多 100 个集成。Code Security 集成没有委派管理员账户/成员账户关系的概念。

为避免遇到限制，建议不要多次使用同一台主机进行集成。

与 GitHub SaaS、GitHub Enterprise Cloud 和 GitHub Enterprise Server 的集成需要公共互联网访问权限。

Important

出于安全考量等任何原因，第三方集成可能会在未事先通知的情况下被临时或永久禁用。

在 Amazon Inspector 与 GitHub 之间创建集成

本主题介绍如何在 Amazon Inspector 与 GitHub 之间创建集成。

Note

如果这是您首次创建集成，系统会提示您在第 2 步中创建默认扫描配置。在创建扫描配置时，可以选择扫描频率、扫描分析和要扫描的存储库。创建默认扫描配置与创建常规扫描配置相

同。但是，默认扫描配置会自动与导入 Amazon Inspector 的任何新项目和现有项目相关联。如果要创建默认扫描配置，请选择继续此配置。您只能创建一次默认扫描配置。如果您创建默认扫描配置，系统不会提示您再次创建默认扫描配置。每个账户和每个组织只能创建一次默认扫描配置。如果您不想创建默认扫描配置，请选择跳过配置。但是，下次创建集成时，系统会提示您创建默认扫描配置。在创建默认扫描配置或跳过创建默认扫描配置后，您将被定向至集成工作流的第 3 步，您需要在该步骤中输入集成详细信息。

与 GitHub SaaS、GitHub Enterprise Cloud 和 GitHub Enterprise Server 的集成需要公共互联网访问权限。

> **Note**
>
> Amazon Inspector 仅扫描和监控您的默认分支。如果您创建新的默认分支，Amazon Inspector 会扫描并更新该新的默认分支。

> **Important**
>
> 在完成集成创建之前，系统会引导您授权在 Amazon Inspector 与 GitHub 之间建立连接。您必须完成此步骤，才能完成该过程。如果您关闭弹出窗口，将无法继续执行操作。

在 Amazon Inspector 与 GitHub 之间创建集成

1. 使用您的凭证登录。打开 Amazon Inspector 控制台：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。选择连接到，然后选择 GitHub。
3. 在集成详细信息下，输入您的集成名称，然后选择连接到 GitHub。
4. 在弹出窗口中选择授权，以在 Amazon Inspector 和 GitHub 之间创建连接。
5. 在成功横幅中，选择转到 GitHub 连接创建页面。
6. 输入 GitHub 应用程序的安装 ID。如果您已安装 GitHub 应用程序，则可以在 GitHub 应用程序页面的 GitHub 中找到安装 ID，或者在 GitHub 应用程序 URL 的末尾找到安装 ID。如果您尚未安装 GitHub 应用程序，请选择安装新应用程序。这样会将您引导至 GitHub，您可以在其中选择 GitHub 组织并指定存储库范围。
7. 选择连接到 GitHub。

创建集成后，可能会遇到 Amazon Inspector 无法刷新访问令牌的情况。如果集成主机不可用或 Amazon Inspector 遇到其他通信问题，可能会发生这种情况。要修复此问题，可以在 Code Security 页面的集成选项卡中对连接重新进行身份验证。在状态列下，集成显示为非活动状态，并且 Amazon Inspector 提供了重新进行身份验证的选项。选择重新进行身份验证。您将被重定向到集成工作流，您可以在其中完成连接设置。

如果您删除集成的系统设置，可能会无限期丢失连接。如果发生这种情况，您必须删除集成并创建一个新的集成。删除集成后，您将丢失所有项目并扫描与该集成关联的配置。

在 Amazon Inspector 与 GitLab Self Managed 之间创建集成

本主题介绍如何在 Amazon Inspector 与 GitLab Self Managed 中的代码存储库之间创建集成。

所需信息

创建连接时，需要提供以下信息：

- 集成名称 – 这是添加到您的集成正文中的名称。
- 端点 URL – 这是用于访问您的 GitLab Self Managed 实例的 URL。
- 个人访问令牌 – 个人访问令牌使用管理员账户在 GitLab Self Managed 中创建，必须包括以下作用域：`api`、`read_api`、`read_repository` 和 `write_repository`。

> **Note**
>
> Amazon Inspector 仅扫描和监控您的默认分支。如果您创建新的默认分支，Amazon Inspector 会扫描并更新该新的默认分支。

在 Amazon Inspector 与 GitLab Self Managed 之间创建集成

以下步骤介绍了如何在 Amazon Inspector 与 GitLab Self Managed 中的代码存储库之间创建连接。

> **Note**
>
> 如果这是您首次创建集成，系统会提示您在第 2 步中创建默认扫描配置。在创建扫描配置时，可以选择扫描频率、扫描分析和要扫描的存储库。创建默认扫描配置与创建常规扫描配置相同。但是，默认扫描配置会自动与导入 Amazon Inspector 的任何新项目和现有项目相关联。如果要创建默认扫描配置，请选择继续此配置。您只能创建一次默认扫描配置。如果您创建默

认扫描配置，系统不会提示您再次创建默认扫描配置。每个账户和每个组织只能创建一次默认扫描配置。如果您不想创建默认扫描配置，请选择跳过配置。但是，下次创建集成时，系统会提示您创建默认扫描配置。在创建默认扫描配置或跳过创建默认扫描配置后，您将被定向至集成工作流的第 3 步，您需要在该步骤中输入集成详细信息。

Important

在完成集成创建之前，系统会提示您授权在 Amazon Inspector 与 GitLab 自行管理之间创建连接。您必须完成此步骤，才能完成该过程。如果您关闭弹出窗口，将无法继续执行操作。

创建与 GitLab 自行管理的连接

1. 使用您的凭证登录。在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 从导航窗格中，选择 Code Security。选择 Connect t o 并选择GitLab 自我管理。
3. 在集成详细信息下，输入以下内容：
   a. 在集成名称中，输入添加到集成正文中的名称。
   b. 在端点 URL 中，输入用于访问您的 GitLab 自行管理实例的 URL。
   c. 对于个人访问令牌，请输入具有所需作用域的个人访问令牌。
4. 选择“连接到 GitLab”。
5. 在弹出窗口中选择授权，以完成在 Amazon Inspector 和 GitLab 之间创建连接的过程。

创建集成后，可能会遇到 Amazon Inspector 无法刷新访问令牌的情况。如果集成主机不可用或 Amazon Inspector 遇到其他通信问题，可能会发生这种情况。要修复此问题，可以在 Code Security 页面的集成选项卡中对连接重新进行身份验证。在状态列下，集成显示为非活动状态，并且 Amazon Inspector 提供了重新进行身份验证的选项。选择重新进行身份验证。您将被重定向到集成工作流，您可以在其中完成连接设置。

如果您删除集成的系统设置，可能会无限期丢失连接。如果发生这种情况，您必须删除集成并创建一个新的集成。删除集成后，您将丢失所有项目并扫描与该集成关联的配置。

查看与代码存储库的集成

本主题介绍如何在 Amazon Inspector 控制台中查看集成。

在 Amazon Inspector 控制台中查看集成

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择集成。在此选项卡中，您可以查看所有已配置的集成，并查看有关所有集成的基本信息。此信息包括集成的名称、集成的状态以及源代码提供商名称。

向提供商进行身份验证

创建集成后，可能会遇到 Amazon Inspector 无法刷新访问令牌的情况。如果集成主机不可用或 Amazon Inspector 遇到其他通信问题，可能会发生这种情况。要修复此问题，可以在 Code Security 页面的集成选项卡中对连接重新进行身份验证。在状态列下，集成显示为非活动状态，并且 Amazon Inspector 提供了重新进行身份验证的选项。选择重新进行身份验证。您将被重定向到集成工作流，您可以在其中完成连接设置。

如果您删除集成的系统设置，可能会无限期丢失连接。如果发生这种情况，您必须删除集成并创建一个新的集成。删除集成后，您将丢失所有项目并扫描与该集成关联的配置。

查看代码存储库

本主题介绍如何在 Amazon Inspector 控制台中查看代码存储库。

在 Amazon Inspector 控制台中查看代码存储库

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择代码存储库。在此选项卡中，您可以查看列为项目的所有代码存储库，并查看有关它们的基本信息。此信息包括每个项目的名称和扫描状态。您还可以查看与项目相关的配置以及上次扫描项目的时间。您甚至可以在搜索栏中筛选项目。

查看项目详细信息

本主题介绍如何在 Amazon Inspector 控制台中查看项目的详细信息。如果您的账户是组织的委派管理员，您可以查看属于成员账户的项目的详细信息。

在 Amazon Inspector 控制台中查看代码项目

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择代码存储库。在此选项卡中，您可以查看列为项目的所有代码存储库，并查看有关它们的基本信息。此信息包括每个项目的名称和扫描状态。您还可以查看与项目相关的配置以及上次扫描项目的时间。您甚至可以在搜索栏中筛选项目。
4. 选择一个项目。或者选择一个项目，然后选择查看详细信息。在项目详细信息屏幕上，您可以查看有关该项目的基本信息。此信息包括项目的名称和 ID 以及集成 ARN。它包括有关项目扫描时间和提供类型的信息。您甚至可以查看与项目相关的调查发现，也可以导出调查发现并为调查发现创建抑制规则。

删除集成

以下步骤介绍了如何在 Amazon Inspector 控制台中删除集成。删除集成后，您将丢失所有项目并扫描与该集成关联的配置。

在 Amazon Inspector 控制台中删除集成。

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择集成。在此选项卡中，您可以查看所有已配置的集成，并查看有关所有集成的基本信息。这些信息包括集成的名称、集成的状态和集成提供商的类型。
4. 选择一个集成，然后选择删除。

创建扫描配置

在创建扫描配置之前，您必须创建与 Amazon Inspector 的集成。首次创建集成时，系统会提示您创建默认扫描配置。本主题介绍如何创建常规扫描配置。默认扫描配置和常规扫描配置的区别在于，默认扫描配置会自动附加到新项目。您可以跳过创建默认扫描配置。

代码安全最多仅支持 500 个常规扫描配置。代码安全仅支持每个账户和每个组织设置 1 个默认扫描配置。一个扫描配置最多只能与 10 万个项目关联。

一个项目最多可以与总计 4 个扫描配置关联。如果已创建默认扫描配置，则这包括默认扫描配置。无法标记组织的扫描配置。

如果组织的委派管理员创建扫描配置，则扫描配置将在组织级别创建，并应用于组织中的所有成员账户。如果委派管理员创建默认扫描配置，也会发生同样的情况。

在创建扫描配置时，可以选择扫描频率、扫描分析和要扫描的存储库。扫描频率可以是基于变化的定期扫描，也可以自定义。如果选择基于变化的定期扫描，则可以选择启用定期扫描。如果启用定期扫描，则可以将扫描频率设置为扫描发生在周几或月中的某日。自定义扫描允许您选择在更改代码时启用扫描以及进行定期扫描。如果您在更改代码时启用扫描，则可以指定要包含在合并和拉取请求中的扫描触发器。

如果提交 ID 在设定时间内没有变化，可以跳过扫描。对于定期扫描，如果提交 ID 在 1 周内扫描之间没有变化，则会跳过扫描。对于按需扫描，如果提交 ID 在 24 小时内扫描之间没有变化，则会跳过扫描。

> **Note**
>
> 如果扫描配置仅具有针对合并请求和拉取请求的触发器，则仅呈现前 25 个关键或重要扫描结果，并且仅在源代码管理平台中显示。在 Amazon Inspector 中不会显示任何内容。

创建常规扫描配置

1. 使用您的凭证登录。在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 从导航窗格中，选择代码安全。
3. 选择配置，然后选择创建扫描配置。
4. 在扫描详细信息下，执行以下操作：
   - 对于配置名称，输入扫描配置的名称。
5. 在扫描频率下，通过选择基于变化的定期扫描或自定义扫描类型和触发器，来指定代码扫描的频率。
   a. （选项 1）如果选择基于变化的定期扫描，请选择启用定期扫描或禁用定期扫描。
      - 如果选择启用定期扫描，请通过选择要扫描代码的具体星期和日期来设置扫描频率。
   b. （选项 2）如果选择自定义扫描，请决定是否在更改代码时启用扫描以及进行定期扫描。

i. 选择更改代码时启用扫描或更改代码时禁用扫描。如果选择更改代码时启用扫描，请从下拉菜单中指定扫描触发的时间。

ii. 选择启用定期扫描或禁用定期扫描。如果选择启用定期扫描，请通过选择要扫描代码的具体星期和日期来设置扫描频率。也可以根据基于事件的触发器进行扫描。这些事件包括最初针对默认分支打开新的拉取请求时，以及提交被合并或推送到默认分支时。对现有拉取请求的后续更新或修订不会触发扫描。要触发新的扫描，请关闭并重新打开拉取请求。

6. 在扫描分析下，决定是配置完成扫描分析还是配置自定义扫描分析：

   a. （选项 1）如果选择完成扫描分析，将应用以下所有扫描分析：

      - 静态应用程序安全测试 – 分析源代码中是否存在漏洞。
      - IaC 扫描 – 分析配置和预调配基础设施的脚本和代码。
      - 静态软件组成分析 – 检查应用程序中的开源包。

   b. （选项 2）如果选择自定义的扫描分析，则必须从下拉菜单中选择至少一种前面提到的扫描分析类型：

7. （可选）对于标签，创建要应用于项目的键值对。最多可以创建 50 个标签。
8. 选择下一步。
9. 在存储库选择下，选择所有存储库或特定存储库。

   a. （选项 1）如果选择所有存储库，则会对任何现有存储库启用扫描。

   b. （选项 2）如果选择特定存储库，则仅对您指定的存储库启用扫描。

10. 选择下一步。
11. 查看您的选择，然后选择创建扫描配置。

> **Note**
>
> 常规扫描配置仅适用于所有现有的代码存储库。它们不会应用于新的代码存储库。

查看扫描配置

以下步骤介绍了如何在 Amazon Inspector 控制台中查看扫描配置。

> **Note**
>
> 当您在组织级别查看扫描配置时，Code Security 屏幕中的某些详细信息会有所不同，以反映您的 AWS 账户。

查看扫描配置的详细信息

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择配置以查看您的扫描配置列表。如果您是委派管理员，则列表中包含您组织的扫描配置。您可以看到每个扫描配置的名称以及每个扫描配置的创建者（AWS 账户 ID 或组织 ID）。您还可以查看哪些扫描类型和扫描分析类型应用于配置。您甚至可以按搜索栏中的不同字段筛选扫描配置。

查看扫描配置的详细信息

以下步骤介绍了如何在 Amazon Inspector 控制台中查看扫描配置的详细信息。

查看扫描配置的详细信息

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择配置。
4. 选择要查看其详细信息的配置。扫描配置详细信息屏幕提供了扫描配置的概述。在此屏幕上，您可以查看扫描配置 ARN、启用了哪些扫描频率类型以及启用了哪些扫描分析类型。您也可以从此屏幕中删除扫描配置。如果您正在查看属于您的组织的扫描配置，也可以在此屏幕上进行编辑。

编辑扫描配置

您可以随时编辑扫描配置。在编辑扫描配置时，您可以更改扫描频率、扫描分析、标签以及要扫描的存储库。例如，您可以编辑扫描配置，以暂停对特定存储库的扫描。以下过程介绍如何编辑扫描配置。

编辑扫描配置

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择配置。
4. 选择要编辑的配置，然后选择编辑。您也可以选择要编辑的配置，然后选择编辑。

删除扫描配置

您可以随时删除扫描配置。本主题介绍如何删除扫描配置。

删除扫描配置

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择配置。
4. 选择要删除的配置，然后选择删除。或者选择要删除的配置，然后选择删除。

执行按需扫描

可以为您的项目执行按需操作。执行按需扫描时，所有已配置的扫描配置的并集将应用于选定项目。如果您的账户是组织的委派管理员账户，您可以对属于成员账户的项目执行按需扫描。以下步骤介绍了如何在 Amazon Inspector 控制台中执行按需扫描。

执行按需扫描

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格中，选择 Code Security。
3. 选择代码存储库。
4. 选择要扫描的项目，然后选择按需扫描。

Amazon Inspector Code Security 支持的语言

本主题包含 Amazon Inspector Code Security 支持的语言。

SAST 支持的语言

- C#（建议使用 .Net 6.0 及更高版本以外的所有版本）
- C（C 11 或更早版本）
- C++（C++ 17 或更早版本）
- Go（仅限 Go 1.18）
- Java（Java 17 或更早版本）
- JavaScript（EMCMAScript 2021 或更早版本）
- JSX（React 17 或更早版本）
- Kotlin（Kotlin 2.0 或更早版本）
- PHP（PHP 8.2 或更早版本）
- Python（Python 3 系列中的 Python 3.11 或更早版本）
- Ruby（仅限 Ruby 2.7 和 3.2）
- Rust
- Scala（Scala 3.2.2 或更早版本）
- Shell
- TSX
- TypeScript（所有版本）

软件组成分析支持的语言

- Go（仅限 Go 1.18）
- Java（Java 17 或更早版本）
- JavaScript（EMCMAScript 2021 或更早版本）
- PHP（PHP 8.2 或更早版本）
- Python（Python 3 系列中的 Python 3.11 或更早版本）
- .Net
- Ruby（仅限 Ruby 2.7 和 3.2）
- Rust

基础设施即代码的语言

- AWS CDK（Python 和 TypeScript）
- CloudFormation（2010–09–09）
- Terraform（1.6.2 或更早版本）

停用 Code Security

有关停用 Code Security 的更多信息，请参阅停用扫描类型。

了解 Amazon Inspector 调查发现

当 Amazon Inspector 检测到 Amazon EC2 实例、Amazon ECR 容器映像和 Lambda 函数中存在可修复或待修复的漏洞时，它就会生成调查发现。它还会针对在第一方应用程序源代码、第三方应用程序依赖关系以及基础设施即代码中检测到的代码漏洞生成调查发现。调查结果是关于影响您的 AWS 资源之一的漏洞的详细报告。

调查结果以漏洞命名，并提供严重性等级、有关受影响 AWS 资源和非 AWS 资源的信息，以及描述如何修复检测到的漏洞的详细信息。Amazon Inspector 会存储所有活动调查发现，直到其得到修复。

当资源被删除、终止或不再符合扫描资格时，Amazon Inspector 会自动关闭与该资源相关的调查发现，并在 3 天后删除相应的调查发现。如果调查发现因任何其他原因被关闭，则会在 30 天后将其删除。

> **Note**
>
> 如果导致漏洞的问题再次出现，Amazon Inspector 将在关闭调查发现后的七天内重新打开已修复的调查发现。

如果禁用 Amazon Inspector，则调查发现将在 24 小时后删除。如果资源终止，则与该资源相关的所有调查发现都将在 3 天后移除。对于附加到不再符合扫描资格的资源的任何调查发现，情况也是如此。如果您的账户被 AWS 暂停，90 天后将删除发现的结果。已停止实例的调查发现仍处于活动状态。

调查发现状态

Amazon Inspector 将调查发现分为以下几种状态。

活跃

Amazon Inspector 将尚未补救的调查发现归类为活动调查发现。

已抑制

Amazon Inspector 将受一项或多项抑制规则约束的调查发现归类为已抑制调查发现。

已关闭

调查发现得到补救后，Amazon Inspector 会将该发现归类为已关闭调查发现。

主题

Amazon Inspector 调查发现类型

本节介绍 Amazon Inspector 中的不同调查发现类型。

主题

程序包漏洞

程序包漏洞调查发现可识别 AWS 环境中暴露于常见漏洞和风险（CVE）的程序包。攻击者可以利用这些未修补的漏洞来破坏数据的保密性、完整性或可用性，或访问其他系统。CVE 系统提供了针对公共已知的信息安全漏洞和风险的参考方法。有关更多信息，请访问 https://www.cve.org/。

Amazon Inspector 可以为 EC2 实例、ECR 容器映像和 Lambda 函数生成程序包漏洞调查发现。程序包漏洞调查发现具有该调查发现类型所特有的额外详细信息，即 Inspector 评分和漏洞情报。

代码漏洞

代码漏洞调查发现有助于识别可被利用的代码行。代码漏洞包括缺少加密、数据泄露、注入缺陷以及弱加密。Amazon Inspector 通过 Lambda 函数扫描及其 Code Security 功能生成代码漏洞调查发现。

Amazon Inspector 会使用自动推理和机器学习来评估 Lambda 函数应用程序代码，以分析应用程序代码的总体安全合规性。它基于与 Amazon Q 合作开发的内部检测器来识别违反政策的行为和漏洞。有关可能的检测的列表，请参阅 Amazon Q 检测器库。

代码扫描可捕获代码片段以突出显示检测到的漏洞。例如，代码片段可能会以纯文本显示硬编码的凭证或其他敏感资料。Amazon Q 会存储与代码漏洞相关的代码片段。默认情况下，您的代码使用 AWS 拥有的密钥进行加密。但是，如果您想更好地控制这些信息，则可以创建客户托管式密钥来加密您的代码。有关更多信息，请参阅对调查发现中的代码进行静态加密。

> **Note**
>
> 组织的委派管理员无法查看属于成员账户的代码片段。

网络可达性

网络可达性调查发现表明，您的环境中存在通往 Amazon EC2 实例的开放网络路径。当您的 TCP 和 UDP 端口可以从 VPC 边缘（如互联网网关，包括应用程序负载均衡器或经典负载均衡器后的实例）、VPC 对等连接或使用虚拟网关的 VPN 到达时，就会出现这些调查发现。这些调查发现突出显示了可能过于宽松的网络配置，例如管理不善的安全组、访问控制列表或互联网网关，或者网络配置可能允许潜在的恶意访问。

Amazon Inspector 仅针对 Amazon EC2 实例生成网络可达性调查发现。启用 Amazon Inspector 后，Amazon Inspector 每 12 小时对网络可达性调查发现进行一次扫描。

Amazon Inspector 在扫描网络路径时会评估以下配置：

- Amazon EC2 实例
- 应用程序负载均衡器
- Direct Connect
- Elastic Load Balancer
- 弹性网络接口
- 互联网网关
- 网络访问控制列表
- 路由表
- 安全组
- 子网
- Virtual Private Cloud
- 虚拟专用网关
- VPC 端点
- VPC 网关端点
- VPC 对等连接
- VPN 连接

查看 Amazon Inspector 调查发现

可以在 Amazon Inspector 控制台中和通过 Amazon Inspector `ListFindings` API 查看调查发现。在 Amazon Inspector 控制台中，可以在控制面板和调查发现屏幕上查看所有调查发现。默认情况下，这些屏幕仅显示您的活跃和关键的调查发现。但是，您可以筛选调查发现，也可以选择按类别查看调查发现。如果您激活了这些集成，您还可以在 S ecurity Hub CSPM 和 Amazon ECR 中查看一些发现。本节中的步骤介绍了如何在 Amazon Inspector 控制台中以及通过 Amazon Inspector `ListFindings` API 查看调查发现。

Console

查看 Amazon Inspector 调查发现

1. 使用您的凭证登录。在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. （可选）从导航窗格中选择控制面板。控制面板显示了环境覆盖率的概览，仅显示您的活跃和关键的调查发现。
3. （可选）从导航窗格中选择调查发现。此屏幕列出了所有活跃调查发现。您可以使用筛选条件来查看特定的调查发现。要从列表中排除调查发现，请创建抑制规则。要查看调查发现的详细信息，请选择调查发现的名称。
4. （可选）从导航窗格中选择下列选项之一，按类别查看调查发现：
   - 按漏洞 – 显示包含最关键调查发现的漏洞。
   - 按账户 – 显示包含最重要调查发现的账户。只有委派管理员才能使用该类别。
   - 按实例 – 显示包含最关键调查发现的 Amazon EC2 实例。此类别不包括有关网络可用性的信息。
   - 按容器映像 – 显示包含最关键调查发现的 Amazon ECR 容器映像。此类别还提供有关您的容器映像的基本信息。它甚至包括详细信息，例如部署了多少 Amazon ECS 任务和 Amazon EKS 容器组（pod）。从该屏幕中，您可以了解过去 24 小时内 tasks/pods 有多少人正在运行并已停止。
   - 按容器存储库 – 显示包含最关键调查发现的容器存储库。
   - 按 Lambda 函数 – 显示包含最关键调查发现的 Lambda 函数。

API

查看 Amazon Inspector 调查发现

- 运行 ListFindings API 操作。在请求中，可以指定 filterCriteria 返回特定的调查发现。

查看 Amazon Inspector 调查发现的详细信息

本节中的步骤介绍了如何查看 Amazon Inspector 调查发现的详细信息。

查看调查发现的详细信息

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v 2/home 中打开 Amazon Inspector 控制台
2. 选择要查看调查发现的区域。
3. 在导航窗格中，选择调查发现以显示调查发现列表
4. （可选）使用筛选栏选择特定的调查发现。有关更多信息，请参阅 筛选 Amazon Inspector 调查发现。
5. 选择一个调查发现，查看其详细信息面板。

调查发现详细信息面板包含调查发现的基本识别特征。这包括调查发现的标题以及已发现漏洞的基本描述、补救建议和严重性评分。有关评分的信息，请参阅了解 Amazon Inspector 调查发现的严重性级别。

调查发现的详细信息因调查发现类型和受影响的资源而异。

所有搜索结果都包含发现结果的 AWS 账户 ID 号、严重性、发现类型、查找结果的创建日期，以及包含该资源详细信息的受资源影响部分。

调查发现类型决定了可用于该调查发现的补救措施和漏洞情报信息。根据调查发现类型，将提供不同的调查发现详细信息。

程序包漏洞

程序包漏洞调查发现适用于 EC2 实例、ECR 容器映像和 Lambda 函数。有关更多信息，请参阅程序包漏洞。

程序包漏洞调查发现还包括 查看 Amazon Inspector 分数并了解漏洞情报详细信息。

此调查发现类型具有以下详细信息：

- 修复可用 – 表示漏洞是否已在受影响程序包的更新版本中修复。具有下列值之一：
  - YES，这意味着所有受影响的程序包都有已修复漏洞的版本。
  - NO，这意味着受影响的程序包没有已修复漏洞的版本。
  - PARTIAL，这意味着一个或多个（但不是全部）受影响的程序包具有已修复漏洞的版本。
- 攻击漏洞可用 – 表示漏洞具有已知的攻击风险。
  - YES，这意味着在您的环境中发现的漏洞具有已知的攻击风险。Amazon Inspector 无法查看环境中攻击漏洞的利用情况。
  - NO，这意味着漏洞没有已知的攻击风险。
- 受影响的程序包 – 列出了调查发现中标记为易受攻击的各个程序包，以及每个程序包的详细信息：
- 文件路径 – 与调查发现关联的 EBS 卷 ID 和分区号。此字段出现在使用 无代理扫描 扫描的 EC2 实例的调查发现中。
- 已安装版本/已修复版本 – 检测到漏洞的当前已安装程序包的版本号。将已安装的版本号与斜杠（/）后的值进行比较。第二个值是修复检测到的漏洞的软件包的版本号，该版本号由与该发现相关的常见漏洞和披露 (CVEs) 或公告提供。如果漏洞已在多个版本中得到修复，则此字段将列出包含修复的最新版本。如果修复不可用，则此值为 `None available`。

  > **Note**
  >
  > 如果调查发现是在 Amazon Inspector 开始将此字段纳入调查发现之前检测到的，则此字段的值为空。不过，可能提供相应修复。

- 程序包管理器 – 用于配置此程序包的程序包管理器。
- 补救 – 如果可通过更新程序包或编程库进行修复，则此部分将包含可以运行的更新命令。您可以复制提供的命令并在环境中运行它。

  > **Note**
  >
  > 补救命令由供应商数据源提供，可能因系统配置而异。请查看调查发现参考资料或操作系统文档，获取更具体的指导。

- 漏洞详细信息 – 针对调查发现中确定的 CVE，提供指向相应 Amazon Inspector 首选来源的链接，例如美国国家漏洞数据库（NVD）、REDHAT 或其他操作系统供应商。此外，您还可以看到

调查发现的严重性评分。有关严重性评分的更多信息，请参阅 了解 Amazon Inspector 调查发现的严重性级别。提供以下评分，以及每项评分的评分向量：

- 漏洞预测评分系统（EPSS）分数
- Inspector 分数
- 来自 Amazon CVE 的 CVSS 3.1
- 来自 NVD 的 CVSS 3.1
- 来自 NVD 的 CVSS 2.0（如果适用，适用于较旧版本）CVEs

- 相关漏洞 – 指定与调查发现相关的其他漏洞。通常 CVEs ，这些是影响相同软件包版本的其他内容，或者是与发现的 CVE CVEs 属于同一组的其他内容，由供应商确定。
- 受影响的资源 – 包括有关注册表、存储库、资源类型、映像 ID 和映像操作系统的信息。它还包括诸如上次推送映像的时间、部署了多少 Amazon ECS 任务和 Amazon EKS 容器组（pod）以及过去 24 小时内最后使用该映像的时间。如果您已部署任何 Amazon ECS 任务和 Amazon EKS 容器组（pod），则可以通过选择该字段的值来查看详细信息。这样将引导您进入一个屏幕，您可以在其中查看诸如集群 ARN、过去 24 小时内最后使用资源的时间、资源正在运行和停止的计数，以及工作负载的名称和类型等信息。

代码漏洞

代码漏洞调查发现仅适用于 Lambda 函数。有关更多信息，请参阅代码漏洞。此调查发现类型具有以下详细信息：

- 修复可用 – 对于代码漏洞，此值始终为 YES。
- 检测器名称 – 用于检测代码漏洞的 Amazon Q 检测器的名称。有关可能的检测的列表，请参阅 Q 检测器库。
- 检测器标签 – 与检测器关联的 Amazon Q 标签，Amazon Q 使用标签对检测进行分类。
- 相关的 CWE — IDs 与代码漏洞相关的常见弱点枚举 (CWE)。
- 文件路径 – 代码漏洞的文件位置。
- 漏洞位置 – 对于 Lambda 代码扫描代码漏洞，此字段显示 Amazon Inspector 发现漏洞的确切代码行。
- 建议的补救措施 – 这提供了如何编辑代码来补救调查发现的建议。

网络可达性

网络可达性调查发现仅适用于 EC2 实例。有关更多信息，请参阅网络可达性。此调查发现类型具有以下详细信息：

- 开放端口范围 – 可以访问 EC2 实例的端口范围。

- 开放网络路径 – 显示 EC2 实例的开放访问路径。选择路径上的项目可获取更多信息。
- 补救 – 推荐关闭开放网络路径的方法。

查看 Amazon Inspector 分数并了解漏洞情报详细信息

Amazon Inspector 会为 Amazon Elastic Compute Cloud（Amazon EC2）实例调查发现创建分数。可以在 Amazon Inspector 控制台中查看 Amazon Inspector 分数和漏洞情报详细信息。Amazon Inspector 分数提供了详细信息，您可以将其与通用漏洞评分系统中的指标进行比较。这些详细信息仅适用于程序包漏洞调查发现。本节介绍了如何解释 Amazon Inspector 分数以及如何了解漏洞情报详细信息。

Amazon Inspector 评分

Amazon Inspector 会为每个 Amazon EC2 调查发现创建一个分数。Amazon Inspector 通过将 CVSS 基本分数信息与来自您的计算环境的信息（例如网络可达性数据和可利用性数据）相关联来确定分数。Amazon Inspector 支持 Amazon、Debian 和 RHEL 供应商。每个供应商都提供 CVSS v3.1 基本分数。对于其他供应商，Amazon Inspector 使用由国家漏洞数据库（NVD）提供的 CVSS 基本分数。

由于 FedRAMP 的要求，Amazon Inspector 使用 CVSS v3.1 基本分数作为默认分数。但是，当 CVSS 4.0 基本分数可用时，它将包含在您的漏洞元数据中。CVSS 4.0 基本分数提供了额外的指标，以改进漏洞评估。您可以在调查发现的漏洞详细信息和导出的调查发现中找到 CVSS 基本分数的来源和版本。

> **Note**
>
> Amazon Inspector 分数不适用于运行 Ubuntu 的 Linux 实例。Ubuntu 使用与 CVSS 分数不同的自定义严重性评级系统。

Amazon Inspector 评分详细信息

打开调查发现的详细信息页面后，您可以选择 Inspector 分数和漏洞情报选项卡。此面板显示基础评分与 Inspector 分数之间的差异。本节介绍 Amazon Inspector 如何根据程序包的 Amazon Inspector 评分和供应商评分的组合来分配严重性评级。如果评分不同，此面板会显示相应原因。

在 CVSS 分数指标部分，您可以看到一个表格，其中包含 CVSS 基础评分指标与 Inspector 分数的对比情况。参与对比的指标是 first.org 维护的 CVSS 规范文档中定义的基础指标。基础指标汇总如下：

攻击向量

可利用漏洞的环境。对于 Amazon Inspector 调查发现，这可以是网络、相邻网络或本地。

攻击复杂性

这描述了攻击者在利用漏洞时面临的难度级别。低评分意味着攻击者只需满足很少或根本不需要满足其他条件即可利用该漏洞。高评分意味着攻击者需要投入大量精力才能成功利用此漏洞进行攻击。

所需的权限

这描述了攻击者利用漏洞所需的权限级别。

用户互动

该指标说明成功利用此漏洞进行攻击是否需要除攻击者之外的其他真人用户。

范围

这说明一个易受攻击组件中的漏洞是否会影响易受攻击组件安全范围以外的组件中的资源。如果此值为不变，则受影响的资源不会影响其他资源。如果此值为已更改，则表明可利用易受攻击的组件来影响由不同安全机构管理的资源。

机密性

这衡量当漏洞被利用时，对资源内数据机密性的影响程度。其范围为从无（不影响机密性）到高（资源中的所有信息都会泄露，或者密码或加密密钥等机密信息会泄露）。

完整性

这衡量当漏洞被利用时，对受影响资源内数据完整性的影响程度。当攻击者修改受影响资源内的文件时，完整性就会受到威胁。评分范围为从无（即攻击者无法通过漏洞修改任何信息）到高（如果漏洞被利用，攻击者将可以修改任意或所有文件，或者可能被修改的文件会产生严重后果）。

可用性

这衡量当漏洞被利用时，对受影响资源可用性的影响程度。评分范围为从无（漏洞完全不影响可用性）到高（如果漏洞被利用，攻击者可以完全拒绝对资源的访问或导致服务不可用）。

漏洞情报

本节总结了亚马逊提供的有关CVE的现有情报以及网络安全和基础设施安全局 (CISA) 等行业标准安全情报来源。

> **Note**
>
> 来自 CISA 或 Amazon 的英特尔并不适用于所有 CVEs人。

可以在控制台中或使用 BatchGetFindingDetails API 查看漏洞情报详细信息。控制台提供以下详细信息：

ATT&CK

本节显示了与 CVE 相关的 MITRE 战术、技巧和程序 (TTPs)。将显示关联 TTPs 项，如果有两个以上适用， TTPs 则可以选择链接以查看完整列表。选择一种战术或技术，可在 MITRE 网站上打开有关相应战术或技术的信息。

CISA

此部分包含与漏洞相关联的日期。美国网络安全和基础设施安全局（CISA）根据活动利用情况的证据，将漏洞添加到已知被利用的漏洞目录中的日期，以及 CISA 预计系统修复漏洞的截止日期。此信息来自 CISA。

已知恶意软件

此部分列出了利用此漏洞的已知利用包和工具。

上次报告时间

此部分显示此漏洞的最后一次已知公开利用日期。

了解 Amazon Inspector 调查发现的严重性级别

当 Amazon Inspector 生成漏洞调查发现时，它会自动为调查发现分配严重性评级。严重性评级有助于您评估调查发现并对其进行优先级排序。调查发现的严重性评级对应于数字分数和等级：信息性、低、中、高和严重。Amazon Inspector 根据调查发现类型确定调查发现的严重性评级。本节介绍了 Amazon Inspector 如何确定每种调查发现类型的严重性评级。

软件程序包漏洞严重性

Amazon Inspector 使用该 NVD/CVSS 分数作为软件包漏洞严重性评分的基础。该 NVD/CVSS 分数是 NVD 发布并由 CVSS 定义的漏洞严重性分数。该 NVD/CVSS 分数由安全指标组成，例如攻击复杂性、漏洞利用代码成熟度和所需的权限。Amazon Inspector 会生成一个从 1 到 10 的数字评分，

以反映漏洞的严重性。Amazon Inspector 将其归类为基础评分，因为它根据漏洞的内在特征反映了漏洞的严重性，而这些特征不会随着时间的推移改变。该评分还假定了不同部署环境中合理的最坏影响。CVSS v3 标准将 CVSS 评分对应以下严重性评级。

评分	评级
0	信息性
0.1—3.9	低
4.0—6.9	中
7.0—8.9	高
9.0—10.0	重大

程序包漏洞调查发现的严重性也可能是未分类。这意味着供应商尚未为检测到的漏洞设置漏洞评分。在这种情况下，我们建议使用发现的参考文献 URLs 来研究该漏洞并做出相应的响应。

程序包漏洞调查发现包括以下评分和相关的评分向量，这些都属于调查发现的详细信息：

- EPSS 分数
- Inspector 分数
- 来自 Amazon CVE 的 CVSS 3.1
- 来自 NVD 的 CVSS 3.1
- 来自 NVD 的 CVSS 2.0（如适用）

代码漏洞严重性

对于代码漏洞调查发现，Amazon Inspector 使用生成该调查发现的 Amazon Q 检测器定义的严重性级别。使用 CVSS v3 评分系统为每个检测器分配一个严重性。

网络可达性严重性

Amazon Inspector 根据暴露的服务、端口和协议以及开放路径的类型来确定网络可达性漏洞的严重性。下表定义了这些严重性评级。开放路径评级列中的值表示来自虚拟网关、对等 AWS Direct Connect 网络和网络的 VPCs开放路径。所有其他暴露的服务、端口和协议的严重性评级均为“提醒”。

服务	TCP 端口	UDP 端口	互联网路径评级	开放路径评级
DHCP	67、68、546、547	67、68、546、547	中	信息性
Elasticsearch	9300、9200	NA	中	信息性
FTP	21	21	高	中
全局目录 LDAP	3268	NA	中	信息性
全局目录 LDAP over TLS	3269	NA	中	信息性
HTTP	80	80	低	信息性
HTTPS	443	443	低	信息性
Kerberos	88、464、543、544、749、751	88、464、749、750、751、752	中	信息性
LDAP	389	389	中	信息性
LDAP over TLS	636	NA	中	信息性
MongoDB	27017、27018、27019、28017	NA	中	信息性
MySQL	3306	NA	中	信息性
NetBIOS	137、139	137、138	中	信息性
NFS	111、2049、4045、1110	111、2049、4045、1110	中	信息性
Oracle	1521、1630	NA	中	信息性
PostgreSQL	5432	NA	中	信息性

打印服务	515	NA	高	中
RDP	3389	3389	中	低
RPC	111、135、530	111、135、530	中	信息性
SMB	445	445	中	信息性
SSH	22	22	中	低
SQL Server	1433	1434	中	信息性
Syslog	601	514	中	信息性
Telnet	23	23	高	中
WINS	1512、42	1512、42	中	信息性

管理 Amazon Inspector 中的调查发现

借助 Amazon Inspector，可以用不同的方式管理调查发现。可以按状态筛选调查发现。可以根据筛选条件搜索调查发现。可以创建抑制规则来将一些调查发现排除在调查发现列表之外。您也可以将调查结果导出到 AWS Security Hub CSPM亚马逊和亚马逊 EventBridge简单存储服务 (Amazon S3) Service。

主题

筛选 Amazon Inspector 调查发现

可以使用筛选条件筛选 Amazon Inspector 调查发现。如果某项调查发现与您的筛选条件不符，Amazon Inspector 会将该调查发现排除在视图之外。本节介绍如何使用筛选条件筛选 Amazon Inspector 调查发现。

在 Amazon Inspector 控制台中创建筛选条件

在每个调查发现视图中，您都可以使用筛选功能来查找具有特定特征的调查发现。移至其他选项卡视图时，筛选条件将被移除。

筛选条件包含一个条件，其中包括配对的筛选属性与筛选值。不符合筛选条件的调查发现将从调查发现列表中排除。例如，要查看与您的管理员帐户关联的所有结果，您可以选择 AWS 账户 ID 属性并将其与您的十二位数 AWS 账户 ID 的值配对。

有些筛选条件适用于所有调查发现，而有些筛选条件仅适用于特定的资源类型或调查发现类型。

对调查发现视图应用筛选条件

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择调查发现。默认视图会显示所有状态为活动的调查发现。

3. 要按条件筛选调查发现，请选择添加筛选条件栏，查看该视图适用的所有筛选条件的列表。在不同的视图中可使用的筛选条件不同。
4. 从列表中选择您要筛选的条件。
5. 在标准输入窗格中输入所需的筛选值以定义条件。
6. 选择应用，将该筛选条件应用于当前结果。您可以再次选择筛选条件输入栏，继续添加其他筛选条件。
7. （可选）要查看隐藏或已关闭的调查发现，请在筛选条件栏中选择活动，然后选择隐藏或已关闭。选择显示全部，可在同一个视图中查看活动、隐藏和已关闭的调查发现。

抑制 Amazon Inspector 调查发现

可以创建抑制规则来隐藏符合条件的调查发现。例如，可以基于严重性评级创建抑制规则来隐藏调查发现。如果 Amazon Inspector 生成的调查发现与抑制规则相符，Amazon Inspector 就会抑制该调查发现并将其隐藏在视图之外。Amazon Inspector 会存储抑制的调查发现，直到它们得到修复。修复被抑制的调查发现后，Amazon Inspector 就会关闭该调查发现。可以在控制台中查看抑制的调查发现。

您可以创建抑制规则来对最重要的调查发现进行优先级排序。抑制规则对调查发现没有任何影响，因为它们只会将调查发现隐藏在视图之外。无法创建关闭或修复调查发现的抑制规则。您还可以使用 Amazon EventBridge 规则抑制不想要的发现。 AWS Security Hub CSPM 本节中的步骤介绍了如何创建、查看、编辑和删除抑制规则。

> **Note**
>
> 只有组织的委派管理员才能创建和管理抑制规则。

创建抑制规则

您可以创建抑制规则来筛选默认显示的调查发现列表。您可以使用 CreateFilterAPI 并指定SUPPRESS为的值，以编程方式创建禁止规则。`action`

> **Note**
>
> 只有独立账户和 Amazon Inspector 授权的管理员才能创建和管理抑制规则。组织中的成员不会在导航窗格中看到抑制规则的选项。

要创建抑制规则（控制台）

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择抑制规则。然后，选择创建规则。
3. 对于每个条件，请执行以下操作：
   - 选择筛选栏以查看可以添加到抑制规则中的筛选条件的列表。
   - 为您的抑制规则选择筛选条件。
4. 添加完条件后，输入规则的名称，还可选择输入相应描述。
5. 选择保存规则。Amazon Inspector 会立即应用新的抑制规则，并隐藏所有符合条件的调查发现。

查看隐藏的调查发现

默认情况下，Amazon Inspector 不会在 Amazon Inspector 控制台中显示隐藏的调查发现。不过，您可以查看特定规则抑制的调查发现。

要查看隐藏的调查发现

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择抑制规则。
3. 在抑制规则列表中，选择规则的标题。

编辑抑制规则

您可以随时更改抑制规则。

要修改抑制规则

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 从导航窗格中，选择抑制规则。
3. 选择要更改的抑制规则的名称，然后选择编辑。
4. 进行您想要的更改，然后选择保存。

删除抑制规则

抑制规则可以删除。删除抑制规则后，Amazon Inspector 将不再隐藏符合规则标准且未被其他规则抑制的新调查发现和现有调查发现。

删除抑制规则后，符合该规则条件的新调查发现和现有调查发现的状态均变为活动。这意味着它们默认显示在 Amazon Inspector 控制台中。此外，Amazon Inspector 将这些发现 EventBridge 作为事件发布 AWS 给 Security Hub CSPM 和亚马逊。

要删除抑制规则

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择抑制规则。
3. 选中要删除的抑制规则标题旁边的复选框。
4. 选择删除，然后确认您的选择以永久删除规则。

导出 Amazon Inspector 调查发现报告

调查发现报告是一个 CSV 或 JSON 文件，提供了详细的调查发现快照。您可以将调查结果报告导出到 AWS Security Hub CSPM亚马逊和亚马逊 EventBridge简单存储服务 (Amazon S3)。配置调查发现报告时，需要指定要在报告中包含哪些调查发现。默认情况下，调查发现报告包含所有活动调查发现的数据。如果您是组织的委派管理员，调查发现报告则包括贵组织中所有成员账户的数据。要自定义调查发现报告，请创建筛选条件并将其应用于该报告。

当您导出调查结果报告时，Amazon Inspector 会使用您指定的加密您的调查结果数据。 AWS KMS key Amazon Inspector 对调查发现数据进行加密后，它会将调查发现报告存储在您指定的 Amazon S3 存储桶中。您的 AWS KMS 密钥必须与您的 Amazon S3 存储桶 AWS 区域 相同。您的 AWS KMS 密钥策略必须允许 Amazon Inspector 使用它，您的 Amazon S3 存储桶策略必须允许 Amazon Inspector 向其添加对象。导出调查发现告后，您可以从 Amazon S3 存储桶下载该报告或将其转移到新位置。您还可以将 Amazon S3 存储桶用作其他导出的调查发现报告的存储库。

本节介绍如何在 Amazon Inspector 控制台中导出调查发现报告。以下任务要求您验证权限、配置 Amazon S3 存储桶、配置 AWS KMS key、配置和导出调查结果报告。

> **Note**
>
> 如果您使用 Amazon Inspector CreateFindingsReportAPI 导出调查结果报告，则只能查看您的有效发现。如果要查看抑制或关闭的调查发现，则必须在筛选条件中指定 SUPPRESSED 或 CLOSED。

任务

步骤 1：验证您的权限

> **Note**
>
> 首次导出调查发现报告后，第 1 到第 3 步为可选步骤。执行这些步骤取决于您是否要使用相同的 Amazon S3 存储桶以及 AWS KMS key 用于其他导出的调查结果报告。如果您想在完成步骤 1-3 后以编程方式导出调查结果报告，请使用 Amazon Inspector CreateFindingsReportAPI 的操作。

在从 Amazon Inspector 导出调查发现报告之前，请确认您拥有导出调查发现报告以及配置用于加密和存储报告的资源所需的权限。要验证您的权限，请使用 AWS Identity and Access Management (IAM) 查看附加到您的 IAM 身份的 IAM 策略。然后，将这些策略中的信息与以下导出调查发现报告时您需要执行的操作的列表进行比较。

Amazon Inspector

对于 Amazon Inspector，请确认您可以执行以下操作：

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

这些操作允许您检索账户的调查发现数据，并将这些数据导出到调查发现报告中。

如果您计划以编程方式导出大型报告，则还可以验证您是否可以执行以下操作：`inspector2:GetFindingsReportStatus`，用于检查报告的状态；`inspector2:CancelFindingsReport`，用于取消正在进行的导出。

AWS KMS

对于 AWS KMS，请确认允许您执行以下操作：

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

这些操作允许您检索和更新您希望 Amazon Inspector 用来加密报告的 AWS KMS key 密钥策略。

要使用 Amazon Inspector 控制台导出报告，还要确认是否允许您执行以下 AWS KMS 操作：

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

这些操作允许您检索和显示有关您账户的 AWS KMS keys 的信息。然后，您可以选择其中一种密钥来加密报告。

如果您计划创建新的 KMS 密钥来加密报告，则还需要能够执行 `kms:CreateKey` 操作。

Amazon S3

对于 Amazon S3，请验证您是否可以执行以下操作：

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

这些操作允许您创建和配置 Amazon Inspector 用于存储报告的 S3 存储桶。它们还允许您在存储桶中添加和删除对象。

如果要使用 Amazon Inspector 控制台导出报告，还需要验证您是否可以执行 `s3:ListAllMyBuckets` 和 `s3:GetBucketLocation` 操作。这些操作允许您检索和显示有关您账户的 S3 存储桶的信息。然后，您可以选择其中一个存储桶来存储报告。

如果您无法执行一项或多项必要的操作，请在继续下一步之前向 AWS 管理员寻求帮助。

步骤 2：配置 S3 存储桶

验证权限后，就可以配置用于存储调查发现报告的 S3 存储桶了。它可以是您自己账户的现有存储桶，也可以是其他账户拥有 AWS 账户 且允许您访问的现有存储桶。如果您想将报告存储在新存储桶中，请在继续操作之前创建存储桶。

S3 存储桶必须与您要导出的调查结果数据 AWS 区域 相同。例如，如果您在美国东部（弗吉尼亚州北部）区域使用 Amazon Inspector，并且想要导出该区域的调查发现数据，则存储桶也必须位于美国东部（弗吉尼亚州北部）区域。

此外，存储桶的策略必须允许 Amazon Inspector 向存储桶添加对象。本主题说明了如何更新存储桶策略，并提供了要添加到策略中的语句的示例。有关添加和更新存储桶策略的详细信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用存储桶策略。

如果您想将报告存储在其他账户拥有的 S3 存储桶中，请与该存储桶的所有者合作，更新存储桶策略。同时还要获取存储桶的 URI。导出报告时，需要输入此 URI。

更新存储桶策略

1. 使用您的证书登录，然后在 https://console.aws.amazon.com/s3 上打开 Amazon S3 控制台。
2. 在导航窗格中，选择存储桶。
3. 选择要存储调查发现报告的 S3 存储桶。
4. 选择 Permissions（权限）选项卡。
5. 在存储桶策略部分中，选择编辑。
6. 将以下示例语句复制到剪贴板：

   JSON

```
{
    "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:us-
east-1:111122223333:report/*"
                }
            }
        }
    ]
}
```

7. 在 Amazon S3 控制台的存储桶策略编辑器中，将上述语句粘贴到策略中，以将其添加到策略中。

   添加语句时，请确保语法有效。存储桶策略使用 JSON 格式。这意味着需要在语句之前或之后添加一个逗号，具体取决于在策略中添加语句的位置。如果将语句添加在最后，请在前一语句的右大括号后面添加一个逗号。如果将语句添加为第一个语句，或添加在两个现有语句之间，请在语句的右大括号后面添加一个逗号。

8. 使用适合您环境的正确值更新语句，其中：

   - *amzn-s3-demo-bucket*是存储桶的名称。
   - *111122223333*是您的账户 ID AWS 账户。
   - *Region*是你 AWS 区域 在其中使用 Amazon Inspector 并希望允许 Amazon Inspector 向存储桶添加报告。例如，us-east-1 表示美国东部（弗吉尼亚州北部）区域。

> **Note**
>
> 如果您在手动启用的环境中使用 Amazon Inspector AWS 区域，还需要在该Service字段的值中添加相应的区域代码。此字段指定了 Amazon Inspector 的服务主体。例如，如果您在中东（巴林）区域使用 Amazon Inspector，该区域的区域代码为 me-south-1，则需要将语句中的 inspector2.amazonaws.com 替换为 inspector2.me-south-1.amazonaws.com。

请注意，示例语句定义了使用两个 IAM 全局条件键的条件：

- a@@ ws: SourceAccount — 此条件仅允许 Amazon Inspector 为您的账户向存储桶中添加报告。它可以防止 Amazon Inspector 为其他账户向存储桶中添加报告。更具体地说，该条件指定了哪个账户可以将存储桶用于 `aws:SourceArn` 条件指定的资源和操作。

  要为其他账户在存储桶中存储报告，请在此条件中添加其他每个账户的账户 ID。例如：

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

- a@@ ws: SourceArn — 此条件根据要添加到存储桶中的对象的来源限制对存储桶的访问权限。它可以 AWS 服务 防止其他人向存储桶添加对象。它还可以防止 Amazon Inspector 在为您的账户执行其他操作时向存储桶添加对象。更具体地说，只有当对象是调查发现报告，并且这些报告由条件中指定的账户在条件中指定的区域创建时，该条件才允许 Amazon Inspector 向存储桶添加对象。

  要允许 Amazon Inspector 对其他账户执行指定操作，请将每个额外账户的亚马逊资源名称 (ARNs) 添加到此条件中。例如：

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

  `aws:SourceAccount` 和 `aws:SourceArn` 条件指定的账户应匹配。

  这两个条件都有助于防止 Amazon Inspector 在 Amazon S3 事务期间被用作混淆代理。您可以从存储桶策略中删除这些条件，但我们并不建议您这样做。

9. 完成存储桶策略更新后，选择保存更改。

步骤 3：配置 AWS KMS key

验证权限并配置 S3 存储桶后，应确定 AWS KMS key 您希望 Amazon Inspector 用来加密调查发现报告的 。密钥必须是客户管理的对称加密 KMS 密钥。此外，密钥必须与您配置用于存储报告的 S3 存储桶 AWS 区域 相同。

密钥可以是您自己账户中的现有 KMS 密钥，也可以是其他账户拥有的现有 KMS 密钥。如果要使用新的 KMS 密钥，请在继续之前创建密钥。如果您希望使用其他账户拥有的现有密钥，请获取该密钥的 Amazon 资源名称 (ARN)。从 Amazon Inspector 中导出报告时，需要输入此 ARN。有关创建和查看 KMS 密钥设置的信息，请参阅 AWS Key Management Service 开发人员指南中的管理密钥。

确定要使用哪个 KMS 密钥后，向 Amazon Inspector 授予使用该密钥的权限。否则，Amazon Inspector 将无法加密和导出报告。要授予 Amazon Inspector 使用密钥的权限，请更新密钥的密钥策略。有关密钥策略和管理 KMS 密钥访问权限的详细信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS KMS密钥策略。

> **Note**
>
> 以下步骤用于更新现有密钥以允许 Amazon Inspector 使用它。如果您没有现有密钥，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

更新密钥策略

1. 使用您的凭据登录，然后在 https://console.aws.amazon.com/km s 处打开 AWS KMS 控制台。
2. 在导航窗格中，选择客户托管密钥。
3. 选择要用于加密报告的 KMS 密钥。该密钥必须为对称加密 (SYMMETRIC_DEFAULT) 密钥。
4. 在密钥策略选项卡上，选择编辑。如果您没有看到带有编辑按钮的密钥策略，则必须先选择切换到策略视图。
5. 将以下示例语句复制到剪贴板：

```
{
    "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. 在 AWS KMS 控制台的密钥策略编辑器中，将前面的语句粘贴到密钥策略中以将其添加到策略中。

   添加语句时，请确保语法有效。密钥策略使用 JSON 格式。这意味着需要在语句之前或之后添加一个逗号，具体取决于在策略中添加语句的位置。如果将语句添加在最后，请在前一语句的右大括号后面添加一个逗号。如果将语句添加为第一个语句，或添加在两个现有语句之间，请在语句的右大括号后面添加一个逗号。

7. 使用适合您环境的正确值更新语句，其中：

   - *111122223333*是您的账户 ID AWS 账户。
   - *Region*是你想要允许 Amazon Inspector 使用密钥对报告进行加密的。 AWS 区域 例如，us-east-1 表示美国东部（弗吉尼亚州北部）区域。

   > **Note**
   >
   > 如果您在手动启用的环境中使用 Amazon Inspector AWS 区域，还需要在该Service字段的值中添加相应的区域代码。例如，如果您在中东（巴林）区域使用 Amazon Inspector，请将 inspector2.amazonaws.com 替换为 inspector2.me-south-1.amazonaws.com。

   与上一步中存储桶策略的示例语句一样，此示例中的 Condition 字段也使用两个 IAM 全局条件键：

   - a@@ ws: SourceAccount — 此条件仅允许 Amazon Inspector 对您的账户执行指定操作。更具体地说，它决定了哪个账户可以为 aws:SourceArn 条件指定的资源和操作执行指定的操作。

     要允许 Amazon Inspector 为其他账户执行指定操作，请在此条件中添加其他每个账户的账户 ID。例如：

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

- a@@ w SourceArn s: — 此条件会 AWS 服务 阻止其他人执行指定的操作。它还可以防止 Amazon Inspector 在为您的账户执行其他操作时使用密钥。换句话说，只有当对象是调查发现报告，并且这些报告由条件中指定的账户在条件中指定的区域创建时，该条件才允许 Amazon Inspector 使用密钥加密 S3 对象。

  要允许 Amazon Inspector 对其他账户执行指定操作，请 ARNs 为每增加一个账户添加此条件。例如：

  ```
  "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` 和 `aws:SourceArn` 条件指定的账户应匹配。

这些条件有助于防止 Amazon Inspector 在与之进行交易时被用作困惑不解的副手 AWS KMS。您可以从语句中删除这些条件，但我们并不建议您这样做。

8. 完成密钥策略更新后，选择保存更改。

步骤 4：配置和导出调查发现报告

> **Note**
>
> 每次只能导出一份调查发现报告。如果当前正在导出报告，必需等到导出完成后再导出其他调查发现报告。

验证权限并配置资源以加密和存储调查发现报告后，就可以配置和导出报告了。

要配置和导出调查发现报告

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中的调查发现下，选择所有调查发现。

3. （可选）使用调查发现表上方的筛选栏，添加筛选条件，指定要在报告中包含哪些调查发现。添加条件时，Amazon Inspector 会更新表格，使其仅包含符合条件的调查发现。该表提供了报告所含数据的预览。

> **Note**
>
> 建议添加筛选条件。如果您不这样做，则该报告将包含当前 AWS 区域 状态为 “有效” 的所有发现的数据。如果您是某组织的 Amazon Inspector 管理员，则这将包含贵组织中所有成员账户的调查发现数据。
>
> 如果报告包含所有或多个调查发现的数据，则生成和导出报告可能需要很长时间，而且一次只能导出一份报告。

4. 选择导出调查发现。
5. 在导出设置部分的导出文件类型中，指定报告的文件格式：

   - 要创建包含数据的 JavaScript 对象表示法 (.json) 文件，请选择 JSON。

     如果您选择 JSON 选项，则报告将包含每个调查发现的所有字段。有关可能的 JSON 字段的列表，请参阅 Amazon Inspector API 参考中的调查发现数据类型。

   - 要创建包含相应数据的逗号分隔值 (.csv) 文件，请选择 CSV。

     如果您选择 CSV 选项，则报告将仅包含每个调查发现的部分字段，即报告调查发现关键属性的大约 45 个字段。这些字段包括：调查发现类型、标题、严重性、状态、描述、首次查看、上次查看、修复可用、 AWS 账户 ID、资源 ID、资源标签和补救措施。除了这些字段之外，还会记录每个发现的评分细节和参考文献 URLs 。以下是调查发现报告中 CSV 标题的示例：

AWS 账户编号	严重性	修复可用	查找类型	标题	说明	正在查找 ARN	初见	最后一次露面	上次更新时间	Resource ID	容器镜像标签	Region	平台	资源标签	受影响的包裹	Package 已安装版本	已在版本中修复	Package 补救	文件路径	网络路径	年龄（天）	修复	Inspector 分数	Inspector 分数向量	Status	漏洞 ID	供应商	供应商严重性	供应商咨询	供应商公告已发布	NVD CVSS3 分数	NVD CVSS3 矢量图	NVD CVSS2 分数	NVD CVSS2 矢量图	供应商 CVSS3 分数	供应商 CVSS3 矢量图	供应商 CVSS2 分数	供应商 CVSS2 矢量图	资源类型	阿米	资源公有 IPV4	资源私有 IPV4	资源 IPV6	资源 Vpc	端口范围	漏洞可利用	上次被利用的时间为	Lambda 层	Lambda Package 类型	Lambda 上次更新时间为	参考网址

6. 在导出位置下，针对 S3 URI，指定要存储报告的 S3 存储桶：

- 要将报告存储在您的账户拥有的存储桶中，请选择浏览 S3。Amazon Inspector 会列出您的账户的 S3 存储桶。选择所需的存储桶所在的行，然后单击选择。

> **Tip**
>
> 要同时为报告指定 Amazon S3 路径前缀，请在 S3 URI 框中的值后面附加斜杠 (/) 和前缀。然后，Amazon Inspector 会在将报告添加到存储桶时添加前缀，Amazon S3 会生成由该前缀指定的路径。
> 例如，如果您想使用自己的 AWS 账户 ID 作为前缀，并且您的账户 ID 为 111122223333，请在 S3 URI 框中的值后面追加该**/111122223333**值。
> 前缀类似于 S3 存储桶内的目录路径。它使您可以将相似的对象组合到一个存储桶中，就像将相似文件一起存储在文件系统的一个文件夹中一样。有关详细信息，请参阅 Amazon Simple Storage Service 用户指南中的使用文件夹在 Amazon S3 控制台中组织对象。

- 要将报告存储在其他账户拥有的存储桶中，请输入相应存储桶的 URI，例如 **s3://DOC-EXAMPLE_BUCKET**，其中 DOC-EXAMPLE_BUCKET 是存储桶的名称。存储桶所有者可以在存储桶属性中帮您找到这些信息。

7. 对于 KMS 密钥 AWS KMS key ，请指定要用于加密报告的：

   - 要使用自己账户中的密钥，请从列表中选择相应密钥。该列表显示了您账户的客户托管对称加密 KMS 密钥。
   - 要使用其他账户拥有的密钥，请输入相应密钥的 Amazon 资源名称 (ARN)。密钥所有者可以在密钥属性中帮您找到这些信息。有关详细信息，请参阅 AWS Key Management Service 开发人员指南中的查找密钥 ID 和 ARN。

8. 选择导出。

Amazon Inspector 生成调查发现报告，使用您指定的 KMS 密钥对其进行加密，然后将其添加到您指定的 S3 存储桶中。根据您选择在报告中包含的调查发现数量，此过程可能需要几分钟或几小时。导出完成后，Amazon Inspector 会显示一条消息，表明您的调查发现报告已成功导出。（可选）在消息中选择查看报告，可导航到 Amazon S3 中的报告。

请注意，每次只能导出一份报告。如果当前正在导出报告，请等到导出完成后再尝试导出其他报告。

排查导出错误

如果导出调查发现报告时出现错误，Amazon Inspector 会显示一条描述错误的消息。您可以使用本主题中的信息作为指南，找出错误的可能原因和解决方案。

例如，验证 S3 存储桶是否处于当前存储桶中， AWS 区域 并且该存储桶的策略允许 Amazon Inspector 向该存储桶添加对象。此外，请确认当前区域已启 AWS KMS key 用，并确保密钥策略允许 Amazon Inspector 使用该密钥。

修复错误后，请尝试再次导出报告。

不能有多个报告错误

如果您要创建报告时，Amazon Inspector 已经在生成报告，则会收到一条错误消息，其内容为原因：无法同时处理多个报告。之所以出现此错误，是因为 Amazon Inspector 每次只能为一个账户生成一份报告。

要解决错误，您可以等待其他报告完成或取消报告，然后再请求新报告。

您可以使用操作来检查报告的状态，此GetFindingsReportStatus操作会返回当前正在生成的任何报告的报告 ID。

如果需要，您可以使用操作提供的报告 ID 通过该`GetFindingsReportStatus`操作取消当前正在进行的CancelFindingsReport导出。

使用亚马逊创建对 Amazon Inspector 调查结果的自定义回复 EventBridge

Amazon Inspector 在亚马逊中 EventBridge为新生成的调查结果和汇总的调查结果创建一个事件。Amazon Inspector 还会针对调查发现的任何状态更改创建一个事件。这意味着，当您采取诸如重启资源或更改与资源关联的标签之类的操作时，Amazon Inspector 就会针对调查发现创建一个新事件。当 Amazon Inspector 为更新的调查发现创建新事件时，调查发现 `id` 保持不变。

> **Note**
>
> 如果您的账户是 Amazon Inspector 委托管理员账户，则会将事件 EventBridge 发布到您的账户和事件发生地的成员账户。

在 Amazon Inspector 中使用 EventBridge 事件时，您可以自动执行任务，以帮助您应对发现的安全问题。要接收有关基于 EventBridge 事件的 Amazon Inspector 调查结果的通知，您必须创建 EventBridge 规则并为 Amazon Inspector 指定目标。该 EventBridge 规则 EventBridge 允许发送有关 Amazon Inspector 发现的通知，目标则指定将通知发送到何处。

Amazon Inspector 将事件发送 AWS 区域 到你当前使用亚马逊检查器的默认事件总线。这意味着您必须为激活 Amazon Inspector 并将 Amazon Inspector 配置为接收 EventBridge 事件的每个 AWS 区域位置配置事件规则。Amazon Inspector 尽最大努力发出事件。

本节为您提供事件架构的示例，并介绍如何创建 EventBridge 规则。

事件架构

以下是 EC2 调查发现事件的 Amazon Inspector 事件格式示例。有关其他调查发现类型和事件类型的示例架构，请参阅EventBridge 架构。


```
{
    "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"
    }
}
```


创建 EventBridge 规则以通知您 Amazon Inspector 的调查结果

为了提高 Amazon Inspector 调查结果的可见性，您可以使用 EventBridge 设置发送到消息中心的自动查找提醒。本主题向您展示如何向电子邮件、Slack 或 Amazon Chime 发送严重性为 CRITICAL 和 HIGH 的调查发现提醒。您将学习如何设置 Amazon 简单通知服务主题，然后将该主题关联到 EventBridge 事件规则。

步骤 1：设置 Amazon SNS 主题和端点

要设置自动警报，必须首先在 Amazon Simple Notification Service 中设置一个主题并添加一个端点。有关更多信息，请参阅 SNS 指南。

此过程可确定要将 Amazon Inspector 调查发现数据发送到何处。可以在 EventBridge 事件规则创建期间或之后将 SNS 主题添加到事件规则中。

Email setup

创建 SNS 主题

1. 在 v3/home 上登录亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/

2. 从导航窗格中选择主题，然后选择创建主题。
3. 在创建主题部分中，选择标准。接下来，输入主题名称，如 **`Inspector_to_Email`**。其他详细信息是可选的。
4. 选择创建主题。这将打开一个包含新主题详细信息的新面板。
5. 在订阅部分中，选择创建订阅。
6. a. 从协议菜单中选择电子邮件。
   b. 在端点字段中，添加您想要用于接收通知的电子邮件地址。

   > **Note**
   >
   > 创建订阅后，您需要通过电子邮件客户端确认订阅。

   c. 选择创建订阅。
7. 在收件箱中查收订阅消息，然后选择确认订阅。

Slack setup

创建 SNS 主题

1. 在 v3/home 上登录亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/
2. 从导航窗格中选择主题，然后选择创建主题。
3. 在创建主题部分中，选择标准。接下来，输入主题名称，如 **`Inspector_to_Slack`**。其他详细信息是可选的。选择创建主题以完成端点的创建。

在聊天应用程序客户端中配置 Amazon Q 开发者版

1. 导航到聊天应用程序控制台中的 Amazon Q 开发者版，地址为https://console.aws.amazon.com/chatbot/。
2. 从配置的客户端面板中选择配置新的客户端。
3. 选择 Slack，然后选择配置以确认。

> **Note**
>
> 在选择 Slack 时，您必须通过选择允许来确认 Amazon Q 开发者版在聊天应用程序中访问您的通道的权限。

4. 选择配置新通道以打开配置详细信息窗格。

   a. 输入通道的名称。

   b. 对于 Slack 通道，选择您要使用的通道。

   c. 在 Slack 中，右键单击通道名称并选择复制链接，复制私有通道的通道 ID。

   d. 在聊天应用程序中的 Amazon Q Developer 窗口中，将您从 Slack 复制的频道 ID 粘贴到私人频道 ID 字段中。 AWS 管理控制台

   e. 在权限中，如果您还没有角色，请选择使用模板创建 IAM 角色。

   f. 对于策略模板，请选择通知权限。这是聊天应用程序中的 Amazon Q 开发者版的 IAM 策略模板。该策略为 CloudWatch 警报、事件和日志以及 Amazon SNS 主题提供了必要的读取和列出权限。

   g. 对于频道护栏政策，请选择 AmazonInspector 2。ReadOnlyAccess

   h. 选择您之前创建 SNS 主题的区域，然后选择您创建的用于向 Slack 通道发送通知的 Amazon SNS 主题。

5. 选择配置。

Amazon Chime setup

创建 SNS 主题

1. 在 v3/home 上登录亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/
2. 从导航窗格中选择主题，然后选择创建主题。
3. 在创建主题部分中，选择标准。接下来，输入主题名称，如 **`Inspector_to_Chime`**。其他详细信息是可选的。选择创建主题以完成。

在聊天应用程序客户端中配置 Amazon Q 开发者版

1. 导航到聊天应用程序控制台中的 Amazon Q 开发者版，地址为https://console.aws.amazon.com/chatbot/。
2. 从已配置的客户端面板中选择配置新客户端。
3. 选择 Chime，然后选择配置以确认。
4. 在配置详细信息窗格中，输入通道的名称。
5. 在 Amazon Chime 中打开所需的聊天室。

   a. 选择右上角的齿轮图标，然后选择管理 Webhook 和自动程序。

b. 选择复制 URL 以将 Webhook URL 复制到剪贴板。

6. 在聊天应用程序中的 Amazon Q 开发者窗口中，将您复制的 URL 粘贴到 Webhook 网址字段中。 AWS 管理控制台
7. 在权限中，如果您还没有角色，请选择使用模板创建 IAM 角色。
8. 对于策略模板，请选择通知权限。这是聊天应用程序中的 Amazon Q 开发者版的 IAM 策略模板。它为 CloudWatch 警报、事件和日志以及 Amazon SNS 主题提供了必要的读取和列出权限。
9. 选择您之前创建 SNS 主题的区域，然后选择您创建的用于向 Amazon Chime 聊天室发送通知的 Amazon SNS 主题。
10. 选择配置。

步骤 2：为 Amazon Inspector 的调查结果创建 EventBridge 规则

1. 使用您的凭证登录。
2. 打开亚马逊 EventBridge 控制台，网址为https://console.aws.amazon.com/events/。
3. 从导航窗格中选择规则，然后选择创建规则。
4. 输入规则名称，另还可选择输入描述。
5. 选择具有事件模式的规则，然后选择下一步。
6. 在事件模式窗格中，选择自定义模式（JSON 编辑器）。
7. 将下面的 JSON 粘贴到编辑器中。

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"],
  "detail": {
    "severity": ["HIGH", "CRITICAL"],
    "status": ["ACTIVE"]
  }
}
```

> **Note**
>
> 此模式会针对 Amazon Inspector 检测到的各种严重性为 `CRITICAL` 或 `HIGH` 的活动调查发现发送通知。

输入完事件模式后，选择下一步。

8. 在选择目标页面上，选择 AWS 服务。然后，对于选择目标类型，选择 SNS 主题。
9. 对于选择主题，请选择您在第 1 步中创建的 SNS 主题的名称。然后选择下一步。
10. 根据需要添加可选标签，然后选择下一步。
11. 检查规则，然后选择创建规则。

EventBridge 适用于 Amazon Inspector 多账户环境

如果您是 Amazon Inspector 的授权管理员， EventBridge 则规则会根据您的成员账户中的适用调查结果显示在您的账户上。如果您通过 EventBridge 管理员帐户设置发现通知（如上一节所述），您将收到有关多个账户的通知。换句话说，除了您自己账户生成的调查发现和事件的通知外，您还会收到您的成员账户生成的调查发现和事件的通知。

您可以使用调查发现的 JSON 详细信息中的 `accountId` 来识别产生 Amazon Inspector 调查发现的成员账户。

使用 Amazon Inspector 中的控制面板

该控制面板针对 Amazon Inspector 扫描的资源提供了汇总统计数据快照。使用控制面板了解环境覆盖率和重要调查发现。

> **Note**
>
> 如果您的账户是组织的委派管理员账户，则控制面板会显示关于您账户以及组织中所有其他账户的信息。

本主题介绍了如何查看该控制面板以及了解构成该控制面板的组件。

主题

查看控制面板

该控制面板显示了环境覆盖率和重要调查发现的概览。控制面板每 5 分钟自动刷新数据。您可以通过选择屏幕右上角附近的刷新图标，来手动刷新数据。可以通过选择相应项目来查看该项目的支持数据。

> **Note**
>
> 如果您的账户是组织的委派管理员账户，则可以通过在账户字段中输入成员账户 ID 来查看成员账户的汇总统计数据。

查看控制面板：

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 从导航窗格选择控制面板。

了解控制面板组件并解释数据

该控制面板的每个部分都提供了对关键指标或调查发现数据的见解，因此您可以了解当前 AWS 区域内 AWS 资源的漏洞状况。

环境覆盖范围

环境覆盖范围部分提供有关 Amazon Inspector 扫描的资源的统计数据。在该部分中，您可以查看 Amazon Inspector 扫描的 Amazon EC2 实例、Amazon ECR 映像和 AWS Lambda 函数的数量和百分比。如果您作为 Amazon Inspector 委托管理员通过 AWS Organizations 管理多个账户，您还将看到组织账户总数、激活 Amazon Inspector 的账户数量以及由此产生的组织覆盖率数据。您还可以使用此部分来确定哪些资源不在 Amazon Inspector 的覆盖范围内。这些资源可能包含漏洞，这些漏洞可能会被人利用，使您的组织面临风险。有关更多信息，请参阅评估 Amazon Inspector 对您 AWS 环境的覆盖范围。

选择一个覆盖范围组后，可进入所选组的账户管理页面。账户管理页面显示了有关 Amazon Inspector 覆盖哪些账户、Amazon EC2 实例和 Amazon ECR 存储库的详细信息。

覆盖范围组如下所示：

- 账户
- 实例
- 容器存储库
- 容器映像
- Lambda

重要调查发现

重要调查发现部分提供了环境中重要漏洞的数量以及环境中所有调查发现的总数。在本节中，数量按资源和评测类型显示。有关重要调查发现以及 Amazon Inspector 如何确定重要性的更多信息，请参阅了解 Amazon Inspector 调查发现。

选择一个重要调查发现组后，会进入所有调查发现页面，并自动应用筛选条件，显示与所选分组匹配的所有重要调查发现。

重要调查发现组如下所示：

- Amazon Inspector 代码扫描结果
- Amazon EC2 实例调查发现
- Amazon ECR 容器映像调查发现

- Lambda 函数调查发现

基于风险的补救

基于风险的补救部分显示前五个存在严重漏洞的程序包，这些漏洞会影响环境中的大部分资源。修复这些程序包可以显著减少环境面临的关键风险数量。选择程序包名称以查看相关的漏洞详细信息和受影响的资源。

包含最重要调查发现的账户

包含最重要调查发现的账户部分显示环境中包含最重要调查发现的前五个 AWS 账户，以及该账户的调查发现总数。只有当 Amazon Inspector 通过 AWS Organizations 配置为多账户扫描时，才能通过委托管理员账户查看此部分。此视图可帮助委托管理员了解组织内的哪些账户面临的风险最大。

选择账户 ID 以查看有关受影响成员账户的更多信息。

包含最重要调查发现的 Amazon ECR 存储库

包含最重要调查发现的 Elastic Container Registry (ECR) 存储库部分显示环境中包含最重要容器映像调查发现的前五个 Amazon ECR 存储库。该视图显示存储库名称、AWS 账户标识符、存储库创建日期、严重漏洞数量和漏洞总数。此视图可帮助您确定哪些存储库面临的风险最大。

选择存储库名称以查看有关受影响存储库的更多信息。

包含最重要调查发现的容器映像

包含最重要调查发现的容器映像部分显示环境中包含最重要调查发现的前五个容器映像。该视图显示映像标签数据、存储库名称、映像摘要、AWS 账户标识符、严重漏洞数量和漏洞总数。此视图可帮助应用程序所有者确定哪些容器映像可能需要重建和重新启动。

选择容器映像可查看有关受影响的容器映像的更多信息。

包含最重要调查发现的实例

包含最重要调查发现的实例部分显示包含最重要调查发现的前五个 Amazon EC2 实例。该视图显示实例标识符、AWS 账户标识符、亚马逊机器映像（AMI）标识符、严重漏洞数量和漏洞总数。此视图可帮助基础设施所有者确定哪些实例可能需要修补。

选择实例 ID 以查看有关受影响的 Amazon EC2 实例的更多信息。

包含最重要调查发现的亚马逊机器映像 (AMI)

包含最重要调查发现的亚马逊机器映像 (AMI) 部分显示环境中包含最重要调查发现的前五个 AMI。该视图显示 AMI 标识符、AWS 账户标识符、在环境中运行的受影响 EC2 实例的数量、AMI 创建日

期、AMI 操作系统平台、严重漏洞的数量和漏洞总数。此视图可帮助基础设施所有者确定哪些 AMI 可能需要重建。

选择受影响实例以查看有关从受影响的 AMI 启动的实例的更多信息。

包含最重要调查发现的 AWS Lambda 函数

包含最重要调查发现的 AWS Lambda 函数部分显示环境中包含最重要调查发现的前五个 Lambda 函数。该视图显示 Lambda 函数名称、AWS 账户标识符、运行时系统环境、严重漏洞的数量、高危漏洞的数量和漏洞总数。此视图可帮助基础设施所有者确定哪些 Lambda 函数可能需要修复。

选择函数名称以查看有关受影响的 AWS Lambda 函数的更多信息。

包含最重要扫描结果的 Amazon Inspector 代码扫描

包含最重要代码漏洞的项目部分显示包含重要调查发现的前五个项目。您可以选择一个项目来查看有关调查发现的详细信息。当选择一个项目时，将定向到调查发现所在的存储库。“调查发现”选项卡显示您的调查发现的名称及其严重性评级。它显示了使用哪种类型的分析来生成您的调查发现。它还会显示您的调查发现已有多久及其状态。

搜索 Amazon Inspector 漏洞数据库

您可以在 Amazon Inspector 漏洞数据库中搜索通用漏洞披露（CVE）。Amazon Inspector 使用漏洞数据库中的信息来生成与 CVE ID 相关的详细信息。您可以在 CVE 详细信息屏幕上查看这些详细信息。Amazon Inspector 会跟踪漏洞数据库中的软件漏洞并生成调查发现。Amazon Inspec CVEs tor 仅支持 CVE 详细信息屏幕的 “检测平台” 部分中列出的平台。本节介绍如何使用 CVE ID 搜索 Amazon Inspector 漏洞数据库。

> **Note**
>
> 目前，CVE 搜索不支持 Microsoft Windows。

搜索漏洞数据库

本节介绍如何在控制台中以及使用 Amazon Inspector API 搜索漏洞数据库。

> **Note**
>
> 您必须在当前版本中激活 Amazon Inspector， AWS 区域 然后才能搜索漏洞数据库。

Console

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v 2/home 中打开 Amazon Inspector 控制台
2. 从导航窗格中，选择漏洞数据库搜索。
3. 在搜索栏中，输入 CVE ID，然后选择搜索。

API

运行 Amazon Inspector SearchVulnerabilitiesAPI，并按`filterCriteria`以下格式提供单个 CVE ID：`CVE-<year>-<ID>`。

了解 CVE 详细信息

本节介绍如何解释 CVE 详细信息页面。

CVE 详细信息

CVE 详细信息部分包括以下信息：

- CVE 描述和 ID
- CVE 严重性
- 通用漏洞评分系统（CVSS）和漏洞预测评分系统（EPSS）得分
- 检测平台

> **Note**
>
> 如果此字段为空，则 Amazon Inspector 不支持检测您的 CVE ID。

- 常见缺陷枚举（CWE）
- 供应商创建和更新日期

漏洞情报

漏洞情报部分提供了威胁情报数据，例如漏洞利用目标和上次已知的公开漏洞利用日期。

它还提供了来自网络安全和基础设施安全局（CISA）的数据，其中包括补救措施、CVE 被添加到已知被利用漏洞目录的日期以及 CISA 期望联邦机构修复 CVE 的日期。

参考

参考部分提供了资源链接，可供获取有关 CVE 的更多信息。

使用 Amazon Inspector 导出 SBOM

软件物料清单（SBOM）是您代码库中所有开源和第三方软件组件的嵌套清单。Amazon Inspector 为环境中的各个资源提供 SBOM。可以使用 Amazon Inspector 控制台或 Amazon Inspector API 为资源生成 SBOM。可以为 Amazon Inspector 支持和监控的所有资源导出 SBOM。导出的 SBOM 将提供有关软件供应的信息。可以通过评测 AWS 环境的覆盖率来查看资源的状态。本部分介绍如何配置和导出 SBOM。

一些软件组件和程序包管理器使用版本范围或动态引用，而不是固定版本来实现依赖关系。这种做法会产生未解析的哈希值，其中 Amazon Inspector 会识别哈希或 jar 文件，但无法将其映射到特定的名称和版本以进行漏洞检测。Amazon Inspector 现在会将这些未解析的哈希值包含在软件物料清单（SBOM）导出中。虽然无法对这些程序包进行漏洞扫描，但它们的哈希值可在导出的组件列表中找到。

> **Note**
>
> 目前，Amazon Inspector 不支持为 Windows Amazon EC2 实例导出 SBOM。

Amazon Inspector

Amazon Inspector 支持以 CycloneDX 1.4 和 SPDX 2.3 兼容格式导出 SBOM。Amazon Inspector 将 SBOM 以 JSON 文件格式导出到您选择的 Amazon S3 存储桶。

> **Note**
>
> 从 Amazon Inspector 导出的 SPDX 格式与使用 SPDX 2.3 的系统兼容，但它们不包含无权利保留协议 (CC0) 字段。这是因为包含此字段将允许用户重新分发或编辑材料。

来自 Amazon Inspector 的 CycloneDX 1.4 SBOM 格式示例

```
                {
"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"
        }
      ]
    }
  ]
}
```

来自 Amazon Inspector 的 SPDX 2.3 SBOM 格式示例

```
{
 "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 筛选条件

导出 SBOM 时，可以使用筛选条件，为特定资源子集创建报告。如果您不提供筛选条件，则会导出所有活动、受支持的资源的 SBOM。而且，如果您是委托管理员，这还包括所有成员的资源。可使用以下筛选条件：

- AccountID — 此筛选条件可用于导出与特定账户 ID 关联的资源的 SBOM。
- EC2 实例标签 — 此筛选条件可用于导出带有特定标签的 EC2 实例的 SBOM。
- 函数名称 — 此筛选条件可用于导出特定 Lambda 函数的 SBOM。
- 映像标签 — 此筛选条件可用于导出带有特定标签的容器映像的 SBOM。
- Lambda 函数标签 — 此筛选条件可用于导出带有特定标签的 Lambda 函数的 SBOM。
- 资源类型 — 此筛选条件可用于筛选资源类型：EC2/ECR/Lambda。
- 资源 ID — 此筛选条件可用于导出特定资源的 SBOM。

- 存储库名称 — 此筛选条件可用于为特定存储库中的容器映像生成 SBOM。

配置和导出 SBOM

要导出 SBOM，必须先配置 Amazon S3 存储桶和允许 Amazon Inspector 使用的 AWS KMS 密钥。您可以使用筛选条件为资源的特定子集导出 SBOM。要为 AWS 组织中的多个账户导出 SBOM，请在以 Amazon Inspector 委托管理员的身份登录后执行以下步骤。

先决条件

- Amazon Inspector 主动监测的受支持资源。
- 配置了策略的 Amazon S3 存储桶，允许 Amazon Inspector 向存储桶添加对象。有关配置策略的信息，请参阅配置导出权限。
- 配置了策略的 AWS KMS 密钥，允许 Amazon Inspector 使用该策略来加密报告。有关配置策略的信息，请参阅配置用于导出的 AWS KMS 密钥。

> **Note**
>
> 如果您之前配置了 Amazon S3 存储桶和用于调查发现导出的 AWS KMS 密钥，则可以将这些存储桶和密钥用于 SBOM 导出。

选择您的首选访问方法来导出 SBOM。

Console

1. 使用您的凭证登录，然后打开 Amazon Inspector 控制台，网址为：https://console.aws.amazon.com/inspector/v2/home。
2. 使用页面右上角的 AWS 区域选择器选择要为其导出 SBOM 的资源所在的区域。
3. 在导航窗格中，选择导出 SBOM。
4. （可选）在导出 SBOM 页面中，使用添加筛选条件菜单选择要为其创建报告的资源子集。如果未提供筛选条件，Amazon Inspector 将导出所有活动资源的报告。如果您是委托管理员，这将包括您组织中的所有活动资源。
5. 在导出设置下，选择需要的 SBOM 格式。
6. 输入 Amazon S3 URI 或选择浏览 Amazon S3，选择一个 Amazon S3 位置来存储 SBOM。
7. 输入为 Amazon Inspector 配置的密钥，用于加密报告。AWS KMS

API

- 要以编程方式为资源导出 SBOM，请使用 Amazon Inspector API 的 CreateSbomExport 操作。

  在请求中，使用 `reportFormat` 参数指定 SBOM 输出格式，选择 `CYCLONEDX_1_4` 或 `SPDX_2_3`。`s3Destination` 参数是必填的，而且您必须指定一个配置了策略的 S3 存储桶，以允许 Amazon Inspector 对其进行写入。（可选）使用 `resourceFilterCriteria` 参数将报告的范围限制在特定的资源。

AWS CLI

- 要使用 AWS Command Line Interface 为您的资源导出 SBOM，请运行以下命令：

  ```
  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
  ```

  在您的请求中，将 *FORMAT* 替换为您选择的格式：`CYCLONEDX_1_4` 或 `SPDX_2_3`。然后，将 s3 目标的 *user input placeholders* 替换为要导出到的 S3 存储桶的名称、用于 S3 中输出的前缀，以及用于加密报告的 KMS 密钥的 ARN。

用于 Amazon Inspector 事件的 Amazon EventBridge 事件架构

Amazon EventBridge 可将来自应用程序和其他 AWS 服务的实时数据流传输到 AWS Lambda 函数、Amazon Simple Notification Service 主题和 Amazon Kinesis Data Streams 中的数据流等目标。为了支持与其他应用程序、服务和系统的集成，Amazon Inspector 会将调查发现作为事件发布到 EventBridge。您可以使用 Amazon Inspector 发布有关调查发现、覆盖率和扫描的事件。本节提供了 EventBridge 事件的示例架构。

主题

用于 Amazon Inspector 的 Amazon EventBridge 基本架构

以下是 Amazon Inspector 的 EventBridge 事件的基本架构示例。事件详情因事件类型而异。

```
{
    "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 调查发现事件架构示例

以下是 Amazon Inspector 调查发现的 EventBridge 事件的架构示例。当 Amazon Inspector 发现您的某个资源中存在软件漏洞或网络问题时，就会创建调查发现事件。有关创建针对此类事件的通知的指南，请参阅使用亚马逊创建对 Amazon Inspector 调查结果的自定义回复 EventBridge。

以下字段可识别调查发现事件：

- `detail-type` 设置为 `Inspector2 Finding`。
- `detail` 描述了调查发现。
- `detail.resources.tags` 是存储键值数据的位置。

您可以筛选选项卡，查看针对不同资源和调查发现类型的调查发现事件架构。

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**
>
> 详细信息值以对象形式返回单个调查发现的 JSON 详细信息。它不会返回整个调查发现响应语法，该语法支持数组中的多个调查发现。

Amazon Inspector 初始扫描完成事件架构示例

以下是 Amazon Inspector 完成初始扫描事件的 EventBridge 事件架构示例。当 Amazon Inspector 完成对您的某个资源的初始扫描时，会创建此事件。

以下字段可识别初始扫描完成事件：

- `detail-type` 字段设置为 `Inspector2 Scan`。
- `detail` 对象包含一个 `finding-severity-counts` 对象，该对象详细说明了适用严重性类别中调查发现的数量，例如 `CRITICAL`、`HIGH` 和 `MEDIUM`。

从选项中进行选择，按资源类型查看不同的初始扫描事件架构。

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 覆盖率事件架构示例

以下是 Amazon Inspector 覆盖率事件的 EventBridge 事件架构示例。当 Amazon Inspector 扫描资源的覆盖率发生变化时，会创建此事件。以下字段可识别覆盖率事件：

- `detail-type` 字段设置为 `Inspector2 Coverage`。
- `detail` 对象包含一个 `scanStatus` 对象，用于指示资源的新扫描状态。

```
{
    "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 自动启用架构示例

当 Amazon Inspector 无法支持组织中的成员数量时，会将自动启用事件发送给委派管理员。以下字段可识别自动启用事件：

- `detail-type` 字段设置为 `Inspector2 AutoEnable`。
- `detail` 对象描述了自动启用事件失败的原因。

```
{
    "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"
        }
}
```

适用于 Linux 和 Windows 的 Amazon Inspector SSM 插件

本主题介绍适用于 Linux 和 Windows 实例的 Amazon Inspector SSM 插件。

适用于 Linux 的 Amazon Inspector SSM 插件

Amazon Inspector 使用 Amazon Inspector SSM 插件对 Linux 实例执行深度检查扫描。Amazon Inspector SSM 插件会自动安装在 Linux 实例的 `/opt/aws/inspector/bin` 目录中。可执行文件的名称是 `inspectorssmplugin`。

Amazon Inspector 使用 Systems Manager Distributor 在您的实例中部署该插件。要执行深度检查扫描，Systems Manager Distributor 和 Amazon Inspector 必须支持您的 Amazon EC2 实例的操作系统。有关 Systems Manager Distributor 支持的操作系统的信息，请参阅《AWS Systems Manager 用户指南》中的支持的软件包平台和架构。

Amazon Inspector 会创建文件目录来管理 Amazon Inspector SSM 插件收集的用于深度检查的数据。这些文件目录包括 `/opt/aws/inspector/var/input` 和 `/opt/aws/inspector/var/output`。

`/opt/aws/inspector/var/output` 中的 `packages.txt` 文件存储了深度检查发现的程序包的完整路径。如果 Amazon Inspector 在您的实例上多次检测到同一个程序包，则 `packages.txt` 文件会列出发现该程序包的每个位置。

Amazon Inspector 将该插件的日志存储在 `/var/log/amazon/inspector` 目录中。

卸载 Amazon Inspector SSM 插件

如果 `inspectorssmplugin` 文件被无意中删除，则 SSM 关联 `InspectorLinuxDistributor-do-not-delete` 将在下一个扫描间隔尝试重新安装 `inspectorssmplugin`。

如果您停用 Amazon EC2 扫描，则该插件将自动从所有 Linux 主机上卸载。

适用于 Windows 的 Amazon Inspector SSM 插件

Amazon Inspector 需要使用 Amazon Inspector SSM 插件才能扫描 Windows 实例。Amazon Inspector SSM 插件会自动安装在位于 `C:\Program Files\Amazon\Inspector` 的 Windows 实例上，且可执行二进制文件名为 `InspectorSsmPlugin.exe`。

创建以下文件位置是为了存储 Amazon Inspector SSM 插件收集的数据：

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

> **Note**
>
> 默认情况下，Amazon Inspector SSM 插件的运行优先级低于正常水平。

> **Note**
>
> 可以结合默认主机管理配置设置使用 Windows 实例。但是，您必须创建或使用配置了 `ssm:PutInventory` 和 `ssm:GetParameter` 权限的角色。

卸载 Amazon Inspector SSM 插件

如果 `InspectorSsmPlugin.exe` 文件被无意中删除，则 `InspectorDistributor-do-not-delete` 关联将在下一个 Windows 扫描间隔重新安装 `InspectorSsmPlugin.exe` 文件。如果要卸载 Amazon Inspector SSM 插件，可以使用 `AmazonInspector2-ConfigureInspectorSsmPlugin` 文档中的卸载操作。但是，如果停用 Amazon EC2 扫描，则将自动从所有 Windows 主机上卸载 Amazon Inspector SSM 插件。

> **Note**
>
> 如果在停用 Amazon Inspector 之前卸载了 SSM Agent，那么 Amazon Inspector SSM 插件仍会保留在 Windows 主机上，但不会向 Amazon Inspector SSM 插件发送数据。有关更多信息，请参阅停用 Amazon Inspector。

Amazon Inspector SBOM 生成器

软件物料清单（SBOM）是一个包含构建软件所需的组件、库和模块的正式结构化列表。Amazon Inspector SBOM 生成器（Sbomgen）是一种工具，可为存档、容器映像、目录、本地系统以及已编译 Go 文件和 Rust 二进制文件生成 SBOM。Sbomgen 可扫描其中包含已安装程序包相关信息的文件。Sbomgen 找到相关文件后，它便会提取程序包名称、版本和其他元数据。然后，Sbomgen 将程序包元数据转换为 CycloneDX SBOM。您可以使用Sbomgen以文件形式生成 CycloneDX SBOM，也可以在 STDOUT 中生成，然后发送到 Amazon SBOMs Inspector 进行漏洞检测。您也可以Sbomgen将其用作 CI/CD 集成的一部分，它会自动扫描容器映像，作为部署管道的一部分。

支持的程序包类型

Sbomgen 会收集以下程序包类型的清单：

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

支持的容器映像配置检查

Sbomgen 可扫描独立的 Dockerfile 并根据现有映像生成历史记录来查找安全问题。有关更多信息，请参阅 Amazon Inspector Dockerfile 检查。

安装 Sbomgen

Sbomgen 仅适用于 Linux 操作系统。

如果希望 Sbomgen 分析本地缓存的映像，则必须安装 Docker。如果要分析作为 `.tar` 文件导出的映像或托管在远程容器注册表中的映像，则无需 Docker。

Amazon Inspector 建议在至少具有以下硬件规格的系统上运行 Sbomgen：

- 4 核 CPU
- 8 GB RAM

要安装 Sbomgen，请执行以下操作

1. 从架构的正确 URL 中下载最新 Sbomgen zip 文件：

   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

   或者，您可以下载先前版本的 Amazon Inspector SBOM 生成器 zip 文件。
2. 使用以下命令解压缩下载的文件：

   ```
   unzip inspector-sbomgen.zip
   ```
3. 检查提取的目录中是否包含以下文件：

   - `inspector-sbomgen`— 这是你要执行的生成工具 SBOMs。
   - `README.txt` – 这是说明如何使用 Sbomgen 的文档。
   - `LICENSE.txt` – 此文件包含 Sbomgen 的软件许可证。
   - `licenses` – 此文件夹包含 Sbomgen 所使用的第三方程序包的许可证信息。
   - `checksums.txt` – 此文件提供 Sbomgen 工具的哈希值。
   - `sbom.json` – 这是 Sbomgen 工具的 CycloneDX SBOM。
   - `WhatsNew.txt` - 此文件包含汇总的变更日志，因此您可以快速查看各 Sbomgen 版本之间的重大更改和改进。
4. （可选）使用以下命令验证该工具的真实性和完整性：

   ```
   sha256sum < inspector-sbomgen
   ```

   - 将结果与 `checksums.txt` 文件的内容进行比较。

5. 使用以下命令为该工具授予可执行权限。

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

6. 使用以下命令验证是否已成功安装 Sbomgen：

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

您应该可以看到类似于如下所示的输出内容：

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

使用 Sbomgen

本部分介绍可以使用 Sbomgen 的不同方式。可以通过内置示例进一步了解如何使用 Sbomgen。要查看这些示例，请运行 `list-examples` 命令：

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

为容器映像生成 SBOM 并输出结果

您可以使用Sbomgen SBOMs 为容器镜像生成并将结果输出到文件中。可以使用 `container` 子命令启用此功能。

示例命令

在以下代码片段中，可以将 *`image:tag`* 替换为您自己的映像 ID，将 *`output_path.json`* 替换为要保存输出的路径。

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

> **Note**
>
> 扫描时间和性能取决于映像大小和层数。映像较小不仅可以提高 Sbomgen 性能，还可以减少潜在的攻击面。映像较小还可以缩短映像构建、下载和上传的时间。

Sbomgen与一起使用时 ScanSbom，Amazon Inspector Scan API 不会处理 SBOMs 包含超过 5,000 个包裹的包裹。在这种情况下，Amazon Inspector Scan API 会返回 HTTP 400 响应。

如果映像包含批量媒体文件或目录，请考虑使用 `--skip-files` 参数将它们排除在 Sbomgen 之外。

示例：常见的错误情况

容器映像扫描可能因以下错误而失败：

- `InvalidImageFormat` – 当扫描具有损坏 TAR 标头、清单文件或配置文件的格式错误的容器映像时发生。
- `ImageValidationFailure`— 当容器映像组件的校验和或内容长度验证失败时发生，例如内容长度标头不匹配、清单摘要不正确或校验和校验失败。 SHA256
- `ErrUnsupportedMediaType` – 当映像组件包含不受支持的媒体类型时发生。有关支持的媒体类型的信息，请参阅支持的操作系统和媒体类型。

Amazon Inspector 不支持 `application/vnd.docker.distribution.manifest.list.v2+json` 媒体类型。但是，Amazon Inspector 确实支持清单列表。当扫描使用清单列表的映像时，您可以通过 `--platform` 参数显式指定要使用哪个平台。如果未指定 `--platform` 参数，则 Amazon Inspector SBOM 生成器会根据清单的运行平台自动选择清单。

从目录和存档生成 SBOM

您可以使用Sbomgen SBOMs 从目录和档案中生成。可以使用 `directory` 或 `archive` 子命令启用此功能。如果您想从项目文件夹（例如下载的 Git 存储库）生成 SBOM，Amazon Inspector 建议使用此特征。

示例命令 1

以下代码片段显示了从目录文件生成 SBOM 的子命令。

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

示例命令 2

以下代码片段显示了从存档文件生成 SBOM 的子命令。仅支持 `.zip`、`.tar` 和 `.tar.gz` 存档格式。

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

从 Go 或 Rust 已编译二进制文件生成 SBOM

您可以使用Sbomgen SBOMs 从编译后的文件Go和Rust二进制文件中生成。可以通过 `binary` 子命令启用此功能：

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

从挂载卷生成 SBOM

你可以使用 Amazon Inspector SBOM 生成器 SBOMs 从已安装的卷中生成。可以使用 `volume` 子命令启用此功能。我们建议您在要分析存储卷（例如已挂载到系统的 Amazon EBS 卷）时使用此功能。与目录子命令不同的是，挂载卷扫描可以检测操作系统程序包和操作系统信息。

您可以扫描亚马逊 EBS 卷，方法是将其连接到安装了 Amazon Inspector SBOM 生成器的亚马逊实例，然后将其安装在该 EC2 实例上。对于其他亚马逊 EC2 实例当前正在使用的 Amazon EBS 卷，您可以创建该卷的 Amazon EBS 快照，然后根据该快照创建新的 Amazon EBS 卷以进行扫描。有关 Amazon EBS 的更多信息，请参阅《Amazon Elastic Block Store 用户指南》中的什么是 Amazon EBS？。

示例命令

以下代码片段显示了一个从挂载卷生成 SBOM 的子命令。`--path` 参数应指定其中挂载卷的根目录。

```
# generate SBOM from mounted volume
./inspector-sbomgen volume --path /mount/point/of/volume/root
```

示例命令

以下代码片段显示了一个子命令，它可以从挂载的卷生成 SBOM，同时使用 `--exclude-suffix` 参数排除特定的文件路径。当卷包含批量文件（例如日志文件或媒体文件）时，`--exclude-suffix` 参数特别有用。其路径以指定后缀结尾的文件和目录将被排除在扫描范围之外，这样可以减少扫描时间和内存使用量。

```
# generate SBOM from mounted volume with exclusions
./inspector-sbomgen volume --path /mount/point/of/volume/root \
--exclude-suffix .log \
--exclude-suffix cache
```

目标卷中的所有文件路径都将标准化为其原始路径。例如，当扫描挂载在 /mnt/volume 中的卷（其中包含 /mnt/volume/var/lib/rpm/rpmdb.sqlite 中的文件）时，路径将在生成的 SBOM 中标准化为 /var/lib/rpm/rpmdb.sqlite。

将 SBOM 发送给 Amazon Inspector 进行漏洞识别

除了生成 SBOM 外，您还可以使用 Amazon Inspector Scan API 中的单个命令发送 SBOM 进行扫描。Amazon Inspector 会先评估 SBOM 的内容是否存在漏洞，然后再将调查发现返回给 Sbomgen。根据您的输入，会显示调查发现或将其写入文件。

> **Note**
>
> 您必须拥有 AWS 账户 具有读取权限的活动用户InspectorScan-ScanSbom才能使用此功能。

要启用此功能，请将 --scan-sbom 参数传递给 Sbomgen CLI。也可以将 --scan-sbom 参数传递给以下任何一个 Sbomgen 子命令：archive、binary、container、directory、localhost。

> **Note**
>
> Amazon Inspector Scan API 无法 SBOMs 处理超过 5,000 个包裹。在这种情况下，Amazon Inspector Scan API 会返回 HTTP 400 响应。

您可以使用以下 AWS CLI 参数通过 AWS 个人资料或 IAM 角色向 Amazon Inspector 进行身份验证：

```
--aws-profile profile
--aws-region region
--aws-iam-role-arn role_arn
```

您还可以通过向 Sbomgen 提供以下环境变量，来向 Amazon Inspector 进行身份验证。

```
AWS_ACCESS_KEY_ID=$access_key \
AWS_SECRET_ACCESS_KEY=$secret_key \
```

```
AWS_DEFAULT_REGION=$region \
./inspector-sbomgen arguments
```

要指定响应格式，请使用 `--scan-sbom-output-format cyclonedx` 参数或 `--scan-sbom-output-format inspector` 参数。

示例命令 1

此命令可为最新 Alpine Linux 版本创建 SBOM，然后扫描 SBOM，并将漏洞结果写入 JSON 文件。

```
./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
```

示例命令 2

此命令使用 AWS 证书作为环境变量向 Amazon Inspector 进行身份验证。

```
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
```

示例命令 3

此命令可使用 IAM 角色的 ARN 向 Amazon Inspector 进行身份验证。

```
./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
```

使用其他扫描器增强检测功能

Amazon Inspector SBOM 生成器会根据所使用的命令应用预定义的扫描器。

默认扫描器组

每个 Amazon Inspector SBOM 生成器子命令都会自动应用以下默认扫描器组。

- 对于`directory`子命令：二进制、 programming-language-packages、dockerfile 扫描器组
- 对于`localhost`子命令：os， programming-language-packages，生态系统外扫描仪组
- 对于`container`子命令：os、、extra-cosystems programming-language-packages、dockerfile、二进制扫描器组

特殊扫描器

要包括默认扫描器组之外的扫描器，请使用 `--additional-scanners` 选项，后面加上要添加的扫描器的名称。以下是一个展示如何执行此操作的示例命令。

```
# Add WordPress installation scanner to directory scan
./inspector-sbomgen directory --path /path/to/directory/ --additional-scanners
 wordpress-installation -o output.json
```

以下是显示如何使用逗号分隔列表添加多个扫描器的命令示例。

```
./inspector-sbomgen container --image image:tag --additional-scanners scanner1,scanner2
 -o output.json
```

通过调整要扫描的最大文件大小来优化容器扫描

在分析和处理容器映像时，默认情况下 Sbomgen 会扫描 200 MB 或更小的文件。大于 200 MB 的文件很少包含程序包元数据。当您清点超过 200 MB 的 Go 或 Rust 二进制文件时，可能会遇到未命中数。要调整大小限制，请使用 `--max-file-size` 参数。这样您就可以增加包含大文件的限制，并通过排除大文件来降低限制以减少资源用量。

示例

以下示例说明了如何使用 --max-file-size 参数来增加文件大小。

```
# 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
```

调整此设置有助于控制磁盘用量、内存消耗和总体扫描持续时间。

禁用进度指示器

Sbomgen显示一个旋转的进度指示器，该指示器可能会导致 CI/CD 环境中出现过多的斜杠字符。

```
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
```

可以使用以下 --disable-progress-bar 参数禁用进度指示器：

```
./inspector-sbomgen container --image alpine:latest \
--outfile /tmp/sbom.json \
--disable-progress-bar
```

使用 Sbomgen 向私有注册表进行身份验证

通过提供您的私有注册表身份验证凭证，您可以 SBOMs 从私有注册表中托管的容器生成。可通过以下方法提供这些凭证：

使用缓存的凭证进行身份验证（推荐）

对于此方法，请先向容器注册表进行身份验证。例如，如果您使用的是 Docker，则可以使用 Docker login 命令向容器注册表进行身份验证：docker login。

1. 向容器注册表进行身份验证。例如，如果您使用的是 Docker，则可以使用 Docker `login` 命令向注册表进行身份验证：
2. 向容器注册表进行身份验证后，在注册表中的容器映像上使用 Sbomgen。要使用以下示例，请将 *`image:tag`* 替换为要扫描的映像的名称：

```
./inspector-sbomgen container --image image:tag
```

使用交互式方法进行身份验证

对于这种方法，您需要提供用户名作为参数，Sbomgen 会在需要时提示您输入安全密码。

要使用以下示例，请将 *`image:tag`* 替换为要扫描的映像的名称，将 *`your_username`* 替换为有权访问该映像的用户名：

```
./inspector-sbomgen container --image image:tag --username your_username
```

使用非交互式方法进行身份验证

对于这种方法，请将您的密码或注册表令牌存储在 `.txt` 文件中。

> **Note**
>
> 当前用户应该只能读取此文件。该文件应还包含一行密码或令牌。

要使用以下示例，请将 *`your_username`* 替换为您的用户名，将 *`password.txt`* 替换为包含一行密码或令牌的 `.txt` 文件，将 *`image:tag`* 替换为要扫描的映像的名称：

```
INSPECTOR_SBOMGEN_USERNAME=your_username \
INSPECTOR_SBOMGEN_PASSWORD=`cat password.txt` \
./inspector-sbomgen container --image image:tag
```

来自 Sbomgen 的输出内容示例

下面是使用 Sbomgen 清点的容器映像的 SBOM 示例。

容器映像 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"
        }
      ]
    }
  ]
}
```

先前版本的 Amazon Inspector SBOM 生成器

此主题包括指向最新版本和以前版本的 Amazon Inspector SBOM 生成器的链接。有关安装 Sbomgen 的信息，请参阅安装 Sbomgen。

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.11.2 1.11.2	bef68671bc532e4fb5 29500b62d7af836012 3cd967308d41ad0ce8 f43f43f7762fb 4f11d7037efa443f44 2c4edf7ba28774c4fa 706fba7622e4fba645 bb3ad3958c9	
Linu AMD64 Linu ARM64	1.11.1 1.11.1	809eb7cb80d24fbf6f fdd124438d53a90763 2c222e924913ebd610 44ca9494ca949490	

平台	版本	SHA-256 校验和	
		057f9e4c9970aeda4b da0685e7e02436fd52 23fbe81cec65138551 c63ed77ba0	
Linu AMD64 Linu ARM64	1.11.0 1.11.0	5172a5556cf46f9fbc 5cf1d35bd382919fb6 b41aca1ec938db3a75 530060b0cf c9e2da7b076dc89dc3 9a962a962a7dd9c7d1 fd29230a4eec7eb95f 951d6a179093d0	
Linu AMD64 Linu ARM64	1.10.1 1.10.1	9e33622a7874adfe71 9ab7db75a1e44f4b5f ae3573374068b501c8 9f0accfcfe 78d5a7f800fc26ba86 adab5b634431a91c00 7075e06d6ce46e5068 7d5156184e	
Linu AMD64 Linu ARM64	1.10.0 1.10.0	0b7a553d7d2d17c40a 62f1a11013bc46fa2c 3814f407c11130e15a f3f3fe3fe3fe313769 5ce9e315a4f8f90ff5 eed7ab058efc8dbff6 593dbf653d3d3d3fc4 55f1c37e882ec6466	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.9.1 1.9.1	d0ef4c14fec6c42e70ae55b3e44d17d027132947596e8ef861c0efc3c0efc3c0e5a871 2d8145011c13f5611fc30f4510785d53e98b911717f6dbe69616af4d4b0df61f	
Linu AMD64 Linu ARM64	1.9.0 1.9.0	78b377b2730eb15476173e40885454ae191e953663af3e0928dddfb8608f4655 985bdc06d25eccb87c4a81995c8a2d3c78e1c02beea309a620b2de4954767591	
Linu AMD64 Linu ARM64	1.8.3 1.8.3	54eed5a772f68320f3906bec5920e3a19da904abdace10f985b87859015eef89 febd74a397fb0cdd3356072503f08465ab872d1620d59a2ab7d83bdb076c929d	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.8.2 1.8.2	2e4e3c754e230046349dd975feb48fa953ea5a2de190cbbc17c1c85043936b5a 449a49e222a2bdffe0353435d7b04b0556b35a391c7b9714ce46d1a5382bc3e2	
Linu AMD64 Linu ARM64	1.8.1 1.8.1	9ff7958e298d2b228b0c7617f0a9a873254587fc26aee9826c37273650b389e9 6737584fd2c7d24b56777d02846d1737f47d0121344baea217a3e5368fd98fcc	
Linu AMD64 Linu ARM64	1.8.0 1.8.0	ef32e7fb4ee0af1e47d6b528b47293fc7127c7a7539f7354e84452626a4c204d 0b82ddc691a517bb8fc6ccd67b80ca566b117a1bb410c05764c9b7e3ba76c510	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.7.3 1.7.3	3fba95d44aaea55ad06d3c7635a671662c483474578376d3f11e84474f8de25f 1f4b52e3d80de87b92b563a78bac4a2d898e7af82db5b6791d899d516e97cfbb	
Linu AMD64 Linu ARM64	1.7.2 1.7.2	c44ba9bf1cfeb3ea2d6d0b15d25816800a5045a438474f2f77c390bac41ae4cb d37c5b1605bf82260da0b0f36311c83b1646a4327c3fd8169ba4b3a978470c9c	
Linu AMD64 Linu ARM64	1.7.1 1.7.1	b0beb602a6ae439d4e307bd99682bc8a419fd7d5e78a278bfc718eb18e00b05e 95ff2d9df2fcd1982dd705df1e763f57a0b499b6fe06801e9a80869e2e464831	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.7.0 1.7.0	a6316c2ecd5fde7091d1099335f45f0e2400b3977c92ee4d72bd1eb359320e61 9751ba5e5c6c6c0aef7d29b1c4adbd4088da3a07bb77eaa7de3f04aa33ad8562	
Linu AMD64 Linu ARM64	1.6.3 1.6.3	b6a309e879aaa78d7d8e224eb5214df5fd415244d370885e6c8876db5a4181d2 59ed0b7eb7d1eadadb691f058d32634a03a856ba03ac2ddb8cd3599ceb55cb9a	
Linu AMD64 Linu ARM64	1.6.2 1.6.2	8d8ba06535be614a4d44b1bd74c66d1fd4874ff9ab788ad5e23aa5229db9c687 2bd7b4a88b9c6b041a6ff82f7f9bc116b76cf410bf6eb896fc8d68e717b55f2a	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.6.1 1.6.1	3e3d62dc794b31d9d2de1904592cf42f25e9f42c30eb90cc53385a60b42f1a63 ad89f670908fb0b48bca0242f3ac58e7179f6fabfcc9a2b3fd0e5c3d79e27539	
Linu AMD64 Linu ARM64	1.6.0 1.6.0	ffe671c2c1d1c2142a4af056d1c179eaffbc3925f5afaa6f3d655bd495ce5e1c a733c0b00c7225369c68ad47c57846b4546e2c9f47580ab98394baefc765c134	
Linu AMD64 Linu ARM64	1.5.5 1.5.5	ebcfbe565631de5bc61b1d55d70a2d15b965f628678a2b60cffd01cd0c3443f1 a8e018ceee3a76dd4271f966015c216438b11ee807fcd970753e786baa335b56	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.5.4 1.5.4	aa8c1ffacc563b8797 5497f53eddec0b2939 7a898fac19f4902b8a cb7eeb347b c6ba98d441aa88d3d3 150449c098cd13ce3b aeccee45ad4c9a1326 f8bb8f87fc	
Linu AMD64 Linu ARM64	1.5.3 1.5.3	d493c23121101c9c3d f888e717bf81d7f7b8 1809754f3492e1ae52 f02b089b68 8dfa5c97b3bd45da48 7706e95d1894290f53 b113247bbb89b9fac1 6dab8184b6	
Linu AMD64 Linu ARM64	1.5.2 1.5.2	ff6233d7da9f7e9635 89a0eb8f07bee2ca37 5360365cb6b6e35458 5cf1371910 fd31efb6031754b2bc 8414d7fe9dd14a0677 67704145af0559b350 0cc437c7ee	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.5.1 1.5.1	391fcc52117fed79cae6e92a9e225732166a6df2582aa7f6b5230149761f6732 f9bc90d18724f93db0f5ca3b79136adb7b49fa33fa179a5e87b4d512f256b56b	
Linu AMD64 Linu ARM64	1.5.0 1.5.0	d7b6cb84053358e462d76488d019140ecd05ad405217a60a96b727fb062880fe 067dcf5c302160a5270f89aed3f941bb0571dcb8a59f75dddb1b7747c2a82ec7	
Linu AMD64 Linu ARM64	1.4.0 1.4.0	c8ca73761afd742e1deb98b04eb5714c9c2a574b652a763b18e23560e66aea24 188d9757782278653e65605aaf186feda104345ba2f9de438873e568f1ff6204	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.3.2 1.3.2	57dd5d135 600e84690706cfe958 60e78149988d37cf81 429ce97b9256d179fb 4 91526ecdafc6cc3718 fabe75b2693ace5eff b9c0af3327b484b7f5 a154929997	
Linu AMD64 Linu ARM64	1.3.1 1.3.1	097ec83907c459a36d e11c92d016fffd64f1 c33fd4bcbf2af465e0 979b0d9237 aa93a3d402abc4a986 a9ad9d3de8fcca81ee 25a55596ac6dc4502e d1d6819502	
Linu AMD64 Linu ARM64	1.3.0 1.3.0	21439f92c314daf136 832ca6676a65d28876 8aa69fc6dcd2014a30 38b2701eeb 4a41779b0c3b32242e edef288de6c1bf40fd a0d4246b32fd0cd8d4 e51e58f94b	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.2.1 1.2.1	e022e95e59f1790949bca8dbbb6478a5d3fb677ccd45aa4ba30ebd91ae86ad65 824acc5bb5b0210954fe9ab089d9461453a4975d34292cc0c676837c3a7279b4	
Linu AMD64 Linu ARM64	1.2.0 1.2.0	9625b1a8ae1937ca2179c2535a0ffceca934138e0b66feac9ba3e34ffaa22ec5 7f387e560b41571fb52efd9e620bf2b9e3a067ca781e88aaa977b2b8acdebf35	
Linu AMD64 Linu ARM64	1.1.1 1.1.1	6809b7e46675c66e3af354c53433dc46c4d1ddaf258e05ba15e38e784ea0285e 6361e59fb2448c66c4698ea33979ecaaefc2af4420034aabbbe741242f60dbdd	

平台	版本	SHA-256 校验和	
Linu AMD64 Linu ARM64	1.1.0 1.1.0	f84c8815413d451490b38509950235f88713c0c61c7259a4831934995664bd8f aaffefb5e44195dc55d5fd3289e511720f64c130644cbd58103cf7f36e96f058	
Linu AMD64 Linu ARM64	1.0.0 1.0.0	cc126e24962f1a6497cf17679b3e3b73be68963c47e3968a56e73caacf045b5c 5d5bf97a4acfeaaa73ad6c918738188e0c822e475ef37a334e49d77ba907b08a	

Amazon Inspector SBOM 生成器全面操作系统集合

Amazon Inspector SBOM 生成器可扫描不同的操作系统，以确保对系统组件进行可靠而详细的分析。生成 SBOM 可以帮助您了解操作系统的构成，从而可识别系统托管程序包中的漏洞。本主题介绍了 Amazon Inspector SBOM 生成器支持的不同操作系统程序包集合的主要特征。有关 Amazon Inspector 支持的操作系统的信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。

支持的操作系统构件

Amazon Inspector SBOM 生成器支持以下操作系统构件：

平台	二元	来源	流
Alma Linux	不适用	支持	是

平台	二元	来源	流
Alpine Linux	是	是	不适用
Amazon Linux	不适用	是	不适用
CentOS	不适用	是	不适用
Chainguard	支持	是	不适用
Debian	支持	是	不适用
Distroless	支持	是	不适用
Fedora	不适用	是	不适用
MinimOS	支持	是	不适用
OpenSUSE	不适用	是	不适用
Oracle Linux	不适用	是	不适用
Photon OS	不适用	是	不适用
RHEL	不适用	支持	是
Rocky Linux	不适用	支持	是
SLES	不适用	是	不适用
Ubuntu	支持	是	不适用
Windows	不适用	不适用	不适用

基于 APK 的操作系统程序包集合

本节包括基于 APK 的操作系统程序包集合所支持的平台和主要特征。有关更多信息，请参阅 Alpine Linux 网站上的 Alpine Package Keeper。

支持的平台

以下是支持的平台。

- Alpine Linux

> **Note**
>
> 对于基于 APK 的系统，Amazon Inspector SBOM 生成器会从 `/lib/apk/db/` 文件中收集程序包元数据。

主要 功能

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的版本
- 源程序包识别 – 识别每个已安装程序包的源程序包

示例

以下片段是 APK 数据库文件的示例。

```
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
```

基于 DPKG 的操作系统程序包集合

本节包括基于 DPKG 的操作系统程序包集合所支持的平台和主要特征。有关更多信息，请参阅 Debian 网站上的 Debian 程序包。

支持的平台

支持以下平台。

- Debian
- Ubuntu

> **Note**
>
> 对于基于 DPKG 的系统，Amazon Inspector SBOM 生成器会从 /var/lib/dpkg/status 文件中收集程序包元数据。

主要 功能

以下是基于 DPKG 的操作系统程序包的主要特征。

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的版本
- 源程序包识别 – 识别每个已安装程序包的源程序包

示例

以下片段是 /var/lib/dpkg/ 文件的示例。

```
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/
```

基于 RPM 的操作系统程序包集合

本节包括基于 RPM 的操作系统程序包集合所支持的平台和主要特征。有关更多信息，请参阅 RPM 网站上的 RPM 程序包管理器。

支持的平台

支持以下平台。

- Alma Linux
- Amazon Linux
- CentOS
- Fedora
- OpenSUSE
- Oracle Linux
- PhotonOS
- RedHat Enterprise Linux
- Rocky Linux
- SUSE Linux Enterprise Server

> **Note**
>
> 对于基于 RPM 的系统，Amazon Inspector SBOM 生成器会从 `/var/lib/rpm` 文件中收集程序包元数据。

主要 功能

以下是基于 RPM 的操作系统程序包集合的主要特征。

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的版本
- 源程序包识别 – 识别每个已安装程序包的源程序包
- 流支持 – 提取每个已安装程序包的流元数据

示例

以下是 RPM 数据库文件片段的示例。

```
/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
```

Windows 操作系统版本集

与基于 Linux 的操作系统不同，Windows 不对操作系统本身使用软件包管理系统。Amazon Inspector SBOM 生成器仅收集 Windows 操作系统的版本信息。对于 Windows 应用程序扫描，请改用windows-apps扫描仪。windows-apps扫描仪收集有关在 Windows 系统上安装的应用程序的信息。有关更多信息，请参阅Microsoft applications 生态系统集合。

主要 功能

- 操作系统版本集合-从 Windows 注册表中提取 Windows 操作系统版本。提取的操作系统版本用于Windows操作系统的漏洞检测。

注册表项和值

以下 Windows 注册表项和值用于收集操作系统名称和版本信息。

- 注册表密钥

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion
```

- 注册表值

- ProductName— 操作系统名称和版本（例如"Windows Server 2025 Datacenter"）
- CurrentMajorVersionNumber— 操作系统的主要版本
- CurrentMinorVersionNumber— 操作系统的次要版本
- CurrentBuild— 操作系统的内部版本号
- UBR— 操作系统的修订号

Chainguard 映像程序包集合

本节包括 Chainguard 映像程序包集合支持的平台和主要特征。有关更多信息，请参阅 Chainguard 网站上的映像。

支持的平台

支持以下平台

- Wolfi Linux

> **Note**
>
> 对于 Chainguard 映像，Amazon Inspector SBOM 生成器会从 `/lib/apk/db/installed` 文件中收集程序包元数据。

主要 功能

以下是主要特征。

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的版本
- 源程序包识别 – 识别每个已安装程序包的源程序包

示例

以下片段是 Chainguard 映像文件的示例。

```
P:wolfi-keys
```

```
V:1-r8
A:x86_64
L:MIT
T:Wolfi signing keyring
o:wolfi-keys
```

Distroless 映像程序包集合

Distroless 容器是在 Linux 发行版中排除程序包管理器、Shell 和其他实用程序的容器映像。Distroless 容器仅包含运行应用程序以及提高性能和安全性所需的基本依赖关系。

> **Note**
>
> 对于 Distroless 映像，Amazon Inspector SBOM 生成器会从 `/var/lib/dpkg/status.d` 文件中收集程序包元数据。仅支持基于 Debian 和基于 Ubuntu 的发行版。可以通过 `/etc/os-release` 文件系统中的 NAME 字段来识别它们，该字段显示“Debian”或“Ubuntu”。

主要 功能

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的版本

示例

以下是 Distroless 映像文件的示例。

```
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 程序包集合

本节包括 Minimus 映像程序包集合支持的平台和主要特征。有关更多信息，请参阅 Minimus 网站。

支持的平台

支持以下平台。

- MinimOS

> **Note**
>
> 对于 Minimus 映像，Amazon Inspector SBOM 生成器会从 `/lib/apk/db/installed` 文件中收集程序包元数据。

主要 功能

以下是主要特征。

- 程序包名称集合 – 提取每个已安装程序包的名称
- 版本集合 – 提取每个已安装程序包的名称
- 源程序包识别 – 识别每个已安装程序包的源程序包

以下是 Minimus 映像文件的片段。

```
P:ca-certificates-bundle
V:20241121-r1
A:aarch64
L:MPL-2.0 AND MIT
T:
o:ca-certificates
```

编程语言依赖关系集合

Amazon Inspector SBOM 生成器支持不同的编程语言和框架，它们构成了强大而详细的依赖关系集合。生成 SBOM 可以帮助您了解软件的构成，从而可识别漏洞并保持与安全标准的合规性。Amazon Inspector SBOM 生成器支持以下编程语言和文件格式。

Go 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Go	Go	`go.mod`	不适用	不适用	不适用	不适用	支持
		`go.sum`	不适用	不适用	不适用	不适用	是
		`Go Binaries`	是	不适用	不适用	不适用	是
		`GOMODCACHE`	不适用	不适用	不适用	不适用	否

go.mod/go.sum

使用 `go.mod` 和 `go.sum` 文件来定义和锁定 Go 项目中的依赖关系。Amazon Inspector SBOM 生成器会根据 Go 工具链版本以不同方式管理这些文件。

主要 功能

- 从 `go.mod` 收集依赖关系（如果 Go 工具链版本为 1.17 或更高版本）
- 从 `go.sum` 收集依赖关系（如果 Go 工具链版本为 1.17 或更低版本）
- 解析 `go.mod` 以识别所有已声明的依赖关系和依赖关系版本

示例 **`go.mod`** 文件

以下是 `go.mod` 文件的示例。

```
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
)
```

示例 **go.sum** 文件

以下是 go.sum 文件的示例。

```
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**
>
> 这些文件中的每一个都会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。 GitHub

Go 二进制文件

Amazon Inspector SBOM 生成器从已编译的 Go 二进制文件中提取依赖关系，以提供关于使用中代码的保障。

> **Note**
>
> Amazon Inspector SBOM 生成器支持从使用官方 Go 编译器构建的 Go 二进制文件中捕获和评估工具链版本。有关更多信息，请参阅 Go 网站上的下载并安装。如果您使用来自其他供应商

（例如 Red Hat）的 Go 工具链，由于分发和元数据可用性可能存在差异，因此评估结果可能不准确。

主要 功能

- 直接从 Go 二进制文件中提取依赖关系信息
- 收集二进制文件中嵌入的依赖关系
- 检测并提取用于编译二进制文件的 Go 工具链版本。

GOMODCACHE

Amazon Inspector SBOM 生成器会扫描 Go 模块缓存，以收集有关已安装依赖关系的信息。此缓存会存储下载的模块，以确保在不同的构建中使用相同的版本。

主要 功能

- 扫描 GOMODCACHE 目录以识别缓存的模块
- 提取详细的元数据，包括模块名称、版本和源 URLs

示例结构

以下是 GOMODCACHE 结构的示例。

```
~/go/pkg/mod/
### github.com/gin-gonic/gin@v1.7.2
### golang.org/x/crypto@v0.0.0-20210616213533-5cf6c0f8e123
```

> **Note**
>
> 此结构会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Java 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Java	Maven	已编译的 Java 应用程序（.jar/.war/.ear） pom.xml	不适用 不适用	不适用 不适用	支持 是	不适用 不适用	支持 是

> **Note**
>
> 我们的漏洞评估功能仅支持 Maven Central 存储库。当前不支持第三方存储库，例如 JBoss Enterprise Maven Repository。

Amazon Inspector SBOM 生成器通过分析已编译的 Java 应用程序和 `pom.xml` 文件来执行 Java 依赖关系扫描。在扫描已编译的应用程序时，扫描器会生成 SHA–1 哈希以进行完整性验证，提取嵌入的 `pom.properties` 文件，并解析嵌套的 `pom.xml` 文件。

SHA–1 哈希集合（适用于已编译的 .jar、.war、.ear 文件）

Amazon Inspector SBOM 生成器会尝试收集项目中所有 `.ear`、`.jar` 和 `.war` 文件的 SHA–1 哈希，以保证已编译的 Java 构件的完整性和可追溯性。

主要 功能

- 为所有已编译的 Java 构件生成 SHA–1 哈希

示例构件

以下是 SHA–1 构件的示例。

```
{
  "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**
>
> 此构件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

pom.properties

pom.properties 文件用于 Maven 项目中以存储项目元数据，包括程序包名称和程序包版本。Amazon Inspector SBOM 生成器解析此文件以收集项目信息。

主要 功能

- 解析和提取程序包构件、程序包组和程序包版本

示例 **pom.properties** 文件

以下是文件 pom.properties 的示例。

```
#Generated by Maven
#Tue Mar 16 15:44:02 UTC 2021

version=1.6.0
groupId=net.datafaker
artifactId=datafaker
```

> **Note**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

排除嵌套 **pom.xml** 解析

如果要在扫描已编译的 Java 应用程序时排除 pom.xml 解析，请使用 --skip-nested-pomxml 参数。

pom.xml

pom.xml 文件是 Maven 项目的核心配置文件。它包含有关项目和项目依赖关系的信息。Amazon Inspector SBOM 生成器解析 pom.xml 文件以收集依赖关系，扫描存储库中的独立文件和已编译 .jar 文件内的文件。

主要 功能

- 解析并提取 pom.xml 文件中的程序包构件、程序包组和程序包版本。

支持的 Maven 作用域和标签

依赖关系是通过以下 Maven 作用域收集的：

- compile
- 提供的
- 运行时
- 测试

- 系统
- 导入

依赖关系是通过以下 Maven 标签收集的：<optional>true</optional>。

带有作用域的示例 **pom.xml** 文件

以下是带有作用域的 pom.xml 文件的示例。

```
<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>
```

没有作用域的示例 **pom.xml** 文件

以下是没有作用域的 pom.xml 文件的示例。

```
<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**
>
> 这些文件中的每一个都会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。 GitHub

JavaScript 依赖扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Javascript	Node Modules	node_modules/*/package.json	不适用	不适用	是	是	是
	NPM		不适用	支持	不适用	不适用	否
	PNPM		不适用	是	不适用	不适用	否
	YARN	package-lock.json (v1, v2, and v3) / npm-shrinkwrap.json	不适用	是	不适用	不适用	否

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
		pnpm-lock.yaml yarn.lock					

package.json

package.json 文件是 Node.js 项目的核心组件。它包含有关已安装程序包的元数据。Amazon Inspector SBOM 生成器会扫描此文件，以识别程序包名称和程序包版本。

主要 功能

- 解析 JSON 文件结构以提取程序包名称和版本
- 识别具有私有值的私有程序包

示例 **package.json** 文件

以下是文件 package.json 的示例。

```
{
"name": "arrify",
"private": true,
"version": "2.0.1",
"description": "Convert a value to an array",
"license": "MIT",
"repository": "sindresorhus/arrify"
}
```

> **Note**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

package-lock.json

package-lock.json 文件由 npm 自动生成，用于锁定为项目安装的依赖关系的确切版本。它通过存储所有依赖关系及其子依赖关系的确切版本来确保在不同环境中的一致性。该文件可以区分常规依赖关系和开发依赖关系。

主要 功能

- 解析 JSON 文件结构以提取程序包名称和程序包版本
- 支持开发依赖关系检测

示例 **package-lock.json** 文件

以下是文件 package-lock.json 的示例。

```
"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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

npm-shrinkwrap.json

npm 会自动生成 package-lock.json 和 npm-shrinkwrap.json 文件，用于锁定为项目安装的依赖关系的确切版本。这样，通过存储所有依赖关系及子依赖关系的确切版本，可确保在不同环境中的一致性。这些文件可区分常规依赖关系和开发依赖关系。

主要 功能

- 解析 JSON 文件结构的 package-lock 版本 1、2 和 3，以提取程序包名称和版本
- 支持开发人员依赖关系检测（package-lock.json 会捕获生产和开发依赖关系，允许工具识别开发环境中使用了哪些程序包）
- npm-shrinkwrap.json 文件优先于 package-lock.json 文件

示例

以下是文件 package-lock.json 的示例。

```
"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 文件由 pnpm 生成，用于维护已安装依赖关系版本的记录。它还单独跟踪开发依赖关系。

主要 功能

- 解析 YAML 文件结构以提取程序包名称和版本
- 支持开发依赖关系检测

示例

以下是文件 pnpm-lock.yaml 的示例。

```
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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

yarn.lock

Amazon Inspector SBOM 生成器会尝试收集项目中 `.ear`、`.jar` 和 `.war` 文件的 SHA–1 哈希，以保证已编译的 Java 构件的完整性和可追溯性。

主要 功能

- 为所有已编译的 Java 构件生成 SHA–1 哈希

示例 SHA–1 构件

以下是 SHA–1 构件的示例。

```
"@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**
>
> 此构件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

.NET 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
.NET	.NET Core	*.deps.json	不适用	不适用	不适用	不适用	支持
	Nuget	Packages.config	不适用	不适用	不适用	不适用	是
	Nuget	packages.lock.json	不适用	不适用	是	不适用	是
	.NET	.csproj	不适用	不适用	不适用	不适用	是

Packages.config

`Packages.config` 文件是较旧版本的 Nuget 用来管理项目依赖关系的 XML 文件。它列出了项目引用的所有程序包，包括特定版本。

主要 功能

- 解析 XML 结构以提取软件包 IDs 和版本

示例

以下是文件 `Packages.config` 的示例。

```
<?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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

*.deps.json

`*.deps.json` 文件由 .NET Core 项目生成，包含有关所有依赖关系的详细信息，包括路径、版本和运行时依赖关系。此文件可确保运行时具有用于加载正确版本的依赖关系的必要信息。

主要 功能

- 解析 JSON 结构以获取全面的依赖关系详细信息

- 在 libraries 列表中提取程序包名称和版本。

示例 **.deps.json** 文件

以下是文件 .deps.json 的示例。

```
{
"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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

packages.lock.json

packages.lock.json 文件由较新版本的 Nuget 用来锁定 .NET 项目依赖关系的确切版本，以保证在不同环境中一致使用相同的版本。

主要 功能

- 解析 JSON 结构以列出锁定的依赖关系
- 同时支持直接依赖关系和传递依赖关系
- 提取程序包名称和已解析的版本

示例 **packages.lock.json** 文件

以下是文件 packages.lock.json 的示例。

```
{
"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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

.csproj

`.csproj` 文件采用 XML 编写，是 .NET 项目的项目文件。它包括对 Nuget 程序包、项目属性和构建配置的引用。

主要 功能

- 解析 XML 结构以提取程序包引用

示例 **`.csproj`** 文件

以下是文件 `.csproj` 的示例。

```
<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>
```

示例 **.csproj** 文件

以下是文件 .csproj 的示例。

```
<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**
>
> 这些文件中的每一个都会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。 GitHub

PHP 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
PHP	Composer	composer.lock /vendor/composer/i	不适用 不适用	不适用 不适用	支持 是	不适用 不适用	支持 是

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
		nstalled.json					

composer.lock

会在运行 composer install 或 composer update 命令时自动生成 `composer.lock` 文件。该文件可保证在每个环境中均安装相同版本的依赖关系。这样可提供一致且可靠的构建过程。

主要 功能

- 解析结构化数据的 JSON 格式
- 提取依赖关系名称和版本

示例 **composer.lock** 文件

以下是文件 `composer.lock` 的示例。

```
{
"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**
>
> 该文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

/vendor/composer/installed.json

`/vendor/composer/installed.json` 文件位于 `vendor/composer` 目录中，提供了所有已安装程序包及程序包版本的详尽列表。

主要 功能

- 解析结构化数据的 JSON 格式
- 提取依赖关系名称和版本

示例 **`/vendor/composer/installed.json`** 文件

以下是文件 `/vendor/composer/installed.json` 的示例。

```
{
"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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Python 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Python	pip	requirements.txt	不适用	不适用	不适用	不适用	支持
	Poetry	Poetry.lock	不适用	不适用	不适用	不适用	是
	Pipenv	Pipfile.lock	不适用	不适用	不适用	不适用	是
	Egg/Wheel	.egg-info/PKG-INFO	不适用	不适用	不适用	不适用	是

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
		`.dist-info/METADATA`					

requirements.txt

`requirements.txt` 文件是在 Python 项目中广泛使用的一种格式，用于指定项目依赖关系。此文件中的每一行都包含一个带有其版本约束的程序包。Amazon Inspector SBOM 生成器会解析此文件以准确识别和编目依赖关系。

主要 功能

- 支持版本说明符（== 和 ~= ）
- 支持注释和复杂的依赖关系行

> **Note**
>
> 不支持版本说明符 <= 和 =>。

示例 **`requirements.txt`** 文件

以下是文件 `requirements.txt` 的示例。

```
flask==1.1.2
requests==2.24.0
numpy==1.18.5
foo~=1.2.0
# Comment about a dependency
scipy. # invalid
```

> **Note**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Pipfile.lock

Pipenv 是一款集所有打包世界精华（捆绑、固定和解除固定）于一身的工具。`Pipfile.lock` 可锁定依赖关系的确切版本，以便实现确定性构建。Amazon Inspector SBOM 生成器读取此文件，以列出依赖关系及其已解析的版本。

主要 功能

- 解析依赖关系解析的 JSON 格式
- 支持默认依赖关系和开发依赖关系

示例 **`Pipfile.lock`** 文件

以下是文件 `Pipfile.lock` 的示例。

```
{
"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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Poetry.lock

Poetry 是一款适用于 Python 的依赖关系管理和打包工具。`Poetry.lock` 文件可锁定依赖关系的确切版本，以便实现一致的环境。Amazon Inspector SBOM 生成器会从该文件中提取详细的依赖关系信息。

主要 功能

- 解析结构化数据的 TOML 格式
- 提取依赖关系名称和版本

示例 **`Poetry.lock`** 文件

以下是文件 `Poetry.lock` 的示例。

```
[[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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Egg/Wheel

对于全局安装的 Python 程序包，Amazon Inspector SBOM 生成器支持解析在 `.egg-info/PKG-INFO` 和 `.dist-info/METADATA` 目录中找到的元数据文件。这些文件提供了有关已安装程序包的详细元数据。

主要 功能

- 提取程序包名称和版本
- 同时支持 egg 和 wheel 两种格式

示例 **PKG-INFO/METADATA** 文件

以下是文件 `PKG-INFO/METADATA` 的示例。

```
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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Ruby 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Ruby	Bundler	Gemfile.lock .gemspec globall installed Gems	不适用 不适用 不适用	不适用 不适用 不适用	是 不适用 不适用	不适用 不适用 不适用	支持 是 是

Gemfile.lock

Gemfile.lock 文件可锁定所有依赖关系的确切版本，以确保在每个环境中均使用相同的版本。

主要 功能

- 解析 Gemfile.lock 文件以识别依赖关系和依赖关系版本
- 提取详细的程序包名称和程序包版本

示例 **Gemfile.lock** 文件

以下是文件 Gemfile.lock 的示例。

```
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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

.gemspec

`.gemspec` 文件是包含关于 Gem 的元数据的 RubyGem 文件。Amazon Inspector SBOM 生成器可解析此文件以收集有关 Gem 的详细信息。

主要 功能

- 解析并提取 Gem 名称和 Gem 版本

> **Note**
>
> 不支持参考规范。

示例 **`.gemspec`** 文件

以下是文件 `.gemspec` 的示例。

```
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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

全局安装的 Gem

Amazon Inspector SBOM 生成器支持扫描全局安装的 Gem，这些 Gem 位于标准目录中，例如 Amazon EC2/Amazon ECR 中的 `/usr/local/lib/ruby/gems/<ruby_version>/gems/` 和 Lambda 中的 `ruby/gems/<ruby_version>/gems/`。这样可确保识别和编目所有全局安装的依赖关系。

主要 功能

- 识别并扫描标准目录中所有全局安装的 Gem
- 提取每个全球安装的 Gem 的元数据和版本信息

示例目录结构

以下是目录结构的示例。

```
.
### /usr/local/lib/ruby/3.5.0/gems/
```

```
### actrivesupport-6.1.4
### concurrent-ruby-1.1.9
### i18n-1.8.10
```

> **Note**
>
> 此结构会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Rust 依赖关系扫描

编程语言	软件包管理器	支持的构件	工具链支持	开发依赖关系	传递依赖关系	私有标记	以递归方式
Rust	Cargo.toml	Cargo.toml Cargo.lock Rust binary (built with cargo-auditable)	不适用 不适用 是	不适用 不适用 不适用	不适用 是 不适用	不适用 不适用 不适用	支持 是 是

Cargo.toml

`Cargo.toml` 文件是 Rust 项目的清单文件。

主要 功能

- 解析并提取 Cargo.toml 文件以识别项目程序包名称和版本。

示例 **Cargo.toml** 文件

以下是文件 Cargo.toml 的示例。

```
[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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

Cargo.lock

Cargo.lock 文件会锁定依赖关系版本，以确保无论何时构建项目均使用相同的版本。

主要 功能

- 解析 Cargo.lock 文件以识别所有依赖关系和依赖关系版本。

示例 **Cargo.lock** 文件

以下是文件 Cargo.lock 的示例。

```
# 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**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

具有 cargo-auditable 的 Rust 二进制文件

Amazon Inspector SBOM 生成器收集来自使用 cargo-auditable 库构建的 Rust 二进制文件的依赖关系。这样，通过启用从已编译二进制文件中提取依赖关系，可提供额外的依赖关系信息。

主要 功能

- 直接从使用 cargo-auditable 库构建的 Rust 二进制文件中提取依赖关系信息
- 检索二进制文件中包含的依赖关系的元数据和版本信息

> **Note**
>
> 此文件会生成包含程序包 URL 的输出。此 URL 可用于在生成软件物料清单时指定有关软件包的信息，并且可以包含在 ScanSbomAPI 中。有关更多信息，请参阅网站上的软件包网址。GitHub

不受支持的构件

本节介绍了不受支持的构件。

Java

Amazon Inspector SBOM 生成器仅支持对来源于主流 Maven 存储库的依赖关系进行漏洞检测。不支持私有或自定义 Maven 存储库，例如 Red Hat Maven 和 Jenkins。为了准确检测漏洞，请确保 Java 依赖关系是从主流 Maven 存储库中拉取的。来自其他存储库的依赖关系不会包含在漏洞扫描中。

JavaScript

esbuild 捆绑包

对于 esbuild 压缩捆绑包，Amazon Inspector SBOM 生成器不支持对使用 esbuild 的项目进行依赖关系扫描。由 esbuild 生成的源映射不包含生成准确 Sbomgen 所需的足够元数据（依赖关系名称和版本）。要获得可靠的结果，请在捆绑过程之前扫描原始项目文件，例如 `node_modules/directory` 和 `package-lock.json`。

package.json

Amazon Inspector SBOM 生成器不支持扫描根级别 package.json 文件以获取依赖关系信息。此文件仅指定程序包名称和版本范围，但不包括完全解析的程序包版本。要获得准确的扫描结果，请使用 `package.json` 或包含已解析版本的其他锁定文件（例如 `yarn.lock` 和 `pnpm.lock`）。

Dotnet

当在 `PackageReference` 中使用浮动版本或版本范围时，如果不执行程序包解析，就更难确定项目中使用的确切程序包版本。浮动版本和版本范围允许开发人员指定可接受程序包版本，而不是固定版本的范围。

Go 二进制文件

Amazon Inspector SBOM 生成器不会扫描使用配置为排除构建 ID 的构建标志构建的 Go 二进制文件。这些编译标志阻止 Amazon Inspector SBOM 生成器将二进制文件准确地映射到其原始来源。由于无法提取程序包信息，因此不支持不明确的 Go 二进制文件。要进行准确的依赖关系扫描，请确保 Go 二进制文件是使用默认设置（包括构建 ID）构建的。

Rust 二进制文件

Amazon Inspector SBOM 生成器只有在使用 cargo-auditable 库构建二进制文件时才会扫描 Rust 二进制文件。未利用此库的 Rust 二进制文件缺少进行准确依赖关系提取所需的必要元数据。Amazon Inspector SBOM 生成器从 Rust 1.7.3 开始提取已编译的 Rust 工具链版本，但仅适用于 Linux 环境中的二进制文件。为了进行全面扫描，请在 Linux 上使用 cargo-auditable 构建 Rust 二进制文件。

> **Note**
>
> 即使提取了工具链版本，也不支持对 Rust 工具链本身的漏洞检测。

Amazon Inspector SBOM 生成器全面生态系统集合

Amazon Inspector SBOM 生成器是一款用于创建软件物料清单（SBOM）并对操作系统和编程语言支持的程序包执行漏洞扫描的工具。它支持对核心操作系统以外的各种生态系统进行扫描，确保对基础设施组件进行可靠而详细的分析。通过生成 SBOM，您可以了解现代技术堆栈的构成，识别生态系统组件中的漏洞，并了解第三方软件。

支持的生态系统

生态系统集合将 SBOM 生成扩展到通过操作系统程序包管理器安装的程序包之外。此操作通过收集以替代方法（例如手动安装）部署的应用程序来完成。Amazon Inspector SBOM 生成器支持扫描以下生态系统：

生态系统	应用程序
7-Zip	7-Zip存档器（版本 21.07 及更高版本）
Apache	Apache httpd Apache tomcat

生态系统	应用程序
Atlassian	Jira Core Confluence Jira Software Jira Service Management
Curl	Curl Libcurl
Elasticsearch	Elasticsearch
Google	Chrome
Java	JDK JRE Amazon Corretto
Jenkins	Jenkins（版本 2.400.* 及更高版本）
MariaDB 和 MySQL	MariaDB Server（10.6+、11.x、12.x）
	Oracle MySQL Server Server(8.0、8.4、9.4+)

生态系统	应用程序
Microsoft applications	PowerShell NuGet CLI Visual Studio Code Microsoft Edge SharePoint Server Microsoft Defender Exchange Server Visual Studio .NET Runtime ASP.NET Core Runtime Microsoft Teams Outlook for Windows Microsoft Office Microsoft 365
Nginx	Nginx
Node	Node
Node.JS	node
OpenSSH	OpenSSH（版本 9 和 10）
OpenSSL	OpenSSL
Oracle	Oracle Database Server
PHP	PHP（版本 8.1 及更高版本）

生态系统	应用程序
WordPress	core plugin theme

7-Zip 生态系统集合

受支持的应用程序

- 7 Zip 存档器（版本 21.07 或更高版本）

主要 功能

- 检查7-Zip二进制文件以提取嵌入式版本信息。

> **Note**
> 具体而言，它会从二进制文件中搜索产品版本值。

支持的平台 — Windows

- `C:/Program Files/7-Zip/7z.exe`
- `C:/Program Files/7-Zip/7za.exe`
- `C:/Program Files/7-Zip/7zz.exe`
- `C:/Program Files/7-Zip/7zr.exe`
- `C:/Program Files (x86)/7-Zip/7z.exe`
- `C:/Program Files (x86)/7-Zip/7za.exe`
- `C:/Program Files (x86)/7-Zip/7zz.exe`
- `C:/Program Files (x86)/7-Zip/7zr.exe`

PURL 示例

以下是 7-Zip 的示例程序包 URL。

```
pkg:generic/7zip/7zip@25.01
```

Apache 生态系统集合

本节提供有关 Apache httpd 和 Apache tomcat 应用程序的详细信息。

Apache httpd

受支持的应用程序

- Apache httpd

> **Note**
>
> 漏洞评估仅适用于 Apache httpd 版本 2.0 及更高版本。

主要 功能

- 解析 `/include/ap_release.h` 文件以提取安装宏，其中包含主要标识符字符串、次要标识符字符串和补丁标识符字符串。

支持的平台

Amazon Inspector SBOM 生成器会跨平台扫描常见安装路径中的安装：

Unix

- `/usr/local/apache2/include/`

Windows

- `/Apache24/include/`
- `/Program Files/Apache24/include/`
- `/Program Files (x86)/Apache24/include/`

示例 **ap_release.h** 文件

以下是 ap_release.h 文件中内容的示例。

```
//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 示例

以下是 Apache httpd 应用程序的示例程序包 URL。

```
Sample PURL: pkg:generic/apache/httpd@2.4.1
```

Apache tomcat

受支持的应用程序

- Apache tomcat

> **Note**
>
> 漏洞评估仅适用于 Apache tomcat 版本 9.0 及更高版本。

主要 功能

- 解压缩 catalina.jar 文件以提取 META-INF/MANIFEST.MF 文件中的安装宏，该文件包含版本字符串。

支持的平台

Amazon Inspector SBOM 生成器会跨平台扫描常见安装路径中的安装：

Linux

- /opt/tomcat/lib/
- /usr/share/tomcat/lib
- /var/lib/tomcat/lib/

macOS

- /Library/Tomcat/lib/
- /usr/local/tomcat/lib

Windows

- /Program Files/Apache Software Foundation
- /Program Files (x86)/Apache Software Foundation/

示例 **catalina.jar/META-INF/MANIFEST.MF** 文件

以下是 catalina.jar/META-INF/MANIFEST.MF 文件中内容的示例。

```
//truncated

Implementation-Title: Apache Tomcat
Implementation-Vendor: Apache Software Foundation
Implementation-Version: 10.1.31

//truncated
```

PURL 示例

以下是 Apache tomcat 应用程序的示例程序包 URL。

```
Sample PURL: pkg:generic/apache/tomcat@10.1.31
```

Atlassian 生态系统集合

本节提供有关Atlassian服务器产品和应用程序的详细信息。

Atlassian Server Products

受支持的应用程序

- Jira Core
- Confluence

主要 功能

- Jira Core— 解析 Maven POM 属性atlassian-jira-webapp以提取版本信息。
- Confluence— 解析 Maven POM 属性confluence-webapp以提取版本信息。

支持的平台

Amazon Inspector SBOM 生成器会扫描常见安装路径中的安装：

Linux

- /opt/atlassian/jira/atlassian-jira/META-INF/maven/com.atlassian.jira/atlassian-jira-webapp/pom.properties
- /opt/atlassian/confluence/confluence/META-INF/maven/com.atlassian.confluence/confluence-webapp/pom.properties

PURL 示例

以下是Atlassian服务器 URLs 产品的示例软件包。

```
// Jira Core
pkg:generic/atlassian/jira-core@10.0.1?distro=linux

// Confluence
pkg:generic/atlassian/confluence@9.2.7?distro=linux
```

Atlassian Applications

受支持的应用程序

- Jira Software
- Jira Service Management

主要 功能

- Jira Software— 通过 `jira-software-application` JAR 进行检测并从 Maven POM 属性中提取版本。
- Jira Service Management— 通过 `jira-servicedesk-application` JAR 进行检测并从 Maven POM 属性中提取版本。

支持的平台

Amazon Inspector SBOM 生成器会扫描常见安装路径中的安装：

Linux

- `/opt/atlassian/jira/atlassian-jira/WEB-INF/application-installation/jira-software-application/jira-software-application-*.jar`
- `/opt/atlassian/jira/atlassian-jira/WEB-INF/application-installation/jira-servicedesk-application/jira-servicedesk-application-*.jar`

PURL 示例

以下是Atlassian应用程序 URLs 的示例包。

```
// Jira Software
pkg:generic/atlassian/jira-software@10.3.9?distro=linux
```

```
// Jira Service Management
pkg:generic/atlassian/jira-service-management@10.3.9?distro=linux
```

Curl 生态系统集合

本节提供有关Curl和Libcurl应用程序的详细信息。

Curl

受支持的应用程序

- Curl

支持的平台

- Unix – Linux 和 macOS
  - /usr/local/bin/curl

主要特点 — Curl

- 检查curl二进制文件以提取嵌入式版本信息。

> **Note**
>
> 具体而言，它会在二进制可执行文件`.rodata`部分（Linux 上的 ELF 二进制文件）、部分（Windows 上的 PE 二进制文件）或 __cstring `.rdata` 部分（macOS 上的 MacHO 二进制文件）中搜索版本字符串。

Curl version string

以下是嵌入在Curl二进制文件中的版本字符串的示例：

```
curl/8.14.1
```

从字符串中提取版本`8.14.1`以标识`Curl`版本。

示例 PURL（卷曲）

以下是一个 Curl 版本文件的示例程序包 URL。

```
Sample PURL: pkg:generic/curl/curl@8.14.1
```

Libcurl

受支持的应用程序

- Libcurl

支持的平台

- Unix – Linux 和 macOS
  - /usr/local/bin/curl/curlver.h

主要特点 — Libcurl

- 检查curlver.h以提取嵌入式版本信息Libcurl。

> **Note**
>
> 具体而言，它从定义的LIBCURL_VERSION_MAJORLIBCURL_VERSION_MINOR、和LIBCURL_VERSION_PATCH变量中提取版本。

Libcurl version string

以下是curlver.h文件中版本变量的示例：

```
#define LIBCURL_VERSION_MAJOR 8
    #define LIBCURL_VERSION_MINOR 14
    #define LIBCURL_VERSION_PATCH 1
```

从这些行中提取版本8.14.1以标识Libcurl版本。

示例 PURL (Libcurl)

以下是一个 Libcurl 版本文件的示例程序包 URL。

```
Sample PURL: pkg:generic/curl/libcurl@8.14.1
```

Elasticsearch 生态系统集合

受支持的应用程序

- Elasticsearch

> **Note**
>
> 漏洞评估仅适用于 7.17.0 Elasticsearch 版。

主要 功能

- Version— 解压缩`elasticsearch-<specific.version>.jar`文件以提取包含Elasticsearch版本字符串`META-INF/MANIFEST.MF`的文件中的安装宏。

支持的平台

- Linux— `/etc/elasticsearch/lib` `/opt/elasticsearch/lib/`、和 `/usr/share/elasticsearch/lib/`
- macOS – `/usr/local/var/lib/elasticsearch/lib/`
- Windows— `/elasticsearch/` `/Program Files (x86)/Elastic/elasticsearch/lib/`、和 `/Program Files/Elastic/elasticsearch/lib/`

示例 **elasticsearch-<specific.version>.jar/META-INF/MANIFEST.MF** 文件

以下是一个`elasticsearch-<specific.version>.jar/META-INF/MANIFEST.MF`文件示例。

```
//truncated

Manifest-Version: 1.0
Module-Origin: git@github.com:elastic/elasticsearch.git
X-Compile-Elasticsearch-Version: 8.19.0-SNAPSHOT
X-Compile-Lucene-Version: 9.12.1
X-Compile-Elasticsearch-Snapshot: true
```

```
//truncated
```

PURL 示例

以下是elasticsearch-<specific.version>.jar/META-INF/MANIFEST.MF文件的软件包 URL 示例。

```
pkg:generic/elastic/elasticsearch@8.19.0-SNAPSHOT
```

Google 生态系统集合

受支持的应用程序

- Google Chrome
- Puppeteer（支持 puppeteer 库；不包括 puppeteer-core）

> **Note**
>
> Puppeteer 支持 puppeteer 库。不包括 Puppeteer core。

支持的构件

Amazon Inspector 从以下途径收集 Google Chrome 信息：

- `chrome/VERSION` 文件（构建源）
- `chrome.exe` 文件（Windows Chrome 安装）
- `puppeteer` 文件（安装）

对于每一个支持的构件，Sbomgen 会解析并收集 chrome 文件或 puppeteer 文件。对于 puppeteer 安装，会根据 puppeteer 版本收集相应的 Chromium 版本。有关更多信息，请参阅 Puppeteer 网站上的支持的浏览器。

当 `PUPPETEER_SKIP_CHROMIUM_DOWNLOAD` 环境变量被设置为 `true` 时，评估将被跳过，并且 `skip_chromium_download=true` 限定符将被添加到 Puppeteer 程序包 URL 中。

示例 **`chrome/VERSION`** 版本文件

以下是 `chrome/VERSION` 版本文件的示例。

```
MAJOR=130
MINOR=0
BUILD=6723
PATCH=58
```

PURL 示例

以下是一个 chrome/VERSION 版本文件的示例程序包 URL。

```
Sample PURL: pkg:generic/google/chrome@131.0.6778.87
```

示例 **puppeteer** 版本文件

以下是 puppeteer 版本文件的示例。

```
{
"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 示例

以下是一个 puppeteer 版本文件的示例程序包 URL。

```
Sample PURL: pkg:generic/google/puppeteer@23.9.0
```

PURL 示例

以下是 puppeteer 版本文件的带有跳过限定符的示例程序包 URL。

```
pkg:generic/google/puppeteer@22.15.0?distro=linux&skip_chromium_download=true
```

Java 生态系统集合

受支持的应用程序

- Oracle JDK
- Oracle JRE
- Amazon Corretto

主要 功能

- 提取 Java 安装的字符串。
- 识别包含 Java 运行时的目录路径。
- 将供应商标识为 Oracle JDK、Oracle JRE 和 Amazon Corretto。

Amazon Inspector SBOM 生成器会扫描以下安装路径和平台上的 Java 安装：

- macOS: `/Library/Java/JavaVirtualMachines`
- Linux 32-bit: `/usr/lib/jvm`
- Linux 64-bit: `/usr/lib64/jvm`
- Linux (generic): `/usr/java and /opt/java`

示例 Java 版本信息

以下是 Oracle Java 发行版的示例。

```
// 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 示例

以下是 Oracle Java 发行版的示例程序包 URL。

```
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
```

Jenkins 生态系统集合

受支持的应用程序

- Jenkins 核心

> **Note**
>
> 漏洞评估适用于 2.400.* 及更高Jenkins版本。

主要 功能

- 通过读取包含版本字符串`jenkins.war`的文件从`META-INF/MANIFEST.M`文件中提取Jenkins版本信息。

Amazon Inspector SBOM 生成器在跨平台的常见安装路径中查找 Jenkins 的安装：

Linux

- `/usr/share/jenkins/jenkins.war`
- /usr/share/java/jenkins.war

macOS

- `/opt/homebrew/opt/jenkins-lts/libexec/jenkins.war`

Windows

- `/Program Files/Jenkins/Jenkins.war`
- `/Program Files (x86)/Jenkins/Jenkins.war`

示例文件

以下是不同版本jenkins.war/META-INF/MANIFEST.MF的文件示例。

```
Manifest-Version: 1.0
Created-By: Maven WAR Plugin 3.4.0
Build-Jdk-Spec: 21
Implementation-Title: Jenkins war
Main-Class: executable.Main
Implementation-Version: 2.516.2
Jenkins-Version: 2.516.2
```

```
Manifest-Version: 1.0
Jenkins-Version: 2.414.1
Implementation-Title: Jenkins
Implementation-Version: 2.414.1
Built-By: kohsuke
Created-By: Apache Maven 3.8.6
```

样本 PURLs

以下是 Jenkins LTS 版本 URLs 的 2.516.2 版和自动化服务器版本的 2.414 版的软件包。Jenkins

```
LTS: pkg:generic/jenkins/jenkins-core-lts@2.516.2.1
Regular: pkg:generic/jenkins/jenkins-core@2.414
```

MariaDB和MySQL生态系统集合

MariaDB

受支持的应用程序

- MariaDB Server（10.6+、11.x、12.x）

主要 功能

- 使用数据库特定的模式从数据库服务器二进制文件和头文件中提取版本信息。
- 标识包含数据库服务器安装的目录路径。
- 使用数据驱动的文件类型检测自动区分MariaDB和MySQL安装。

SBOM 生成器在跨平台的常见MariaDB安装路径中查找安装：

Linux

- `/usr/bin/mariadbd`
- `/usr/sbin/mariadbd`
- `/usr/local/bin/mariadbd`

macOS

- `C:/Program Files (x86)/MariaDB/include/mysql/mariadb_version.h (MariaDB)`
- `C:/Program Files/MariaDB/include/mysql/mariadb_version.h (MariaDB)`

Windows

- `C:/Program Files (x86)/MariaDB/include/mysql/mariadb_version.h (MariaDB)`
- `C:/Program Files/MariaDB/include/mysql/mariadb_version.h (MariaDB)`

PURL 示例

以下是MariaDB服务器的软件包 URL 示例。

```
# MariaDB Server

pkg:generic/mysql/mariadb-server@10.11.8
```

MySQL 生态系统集合

受支持的应用程序

- Oracle MySQL Server Server(8.0、8.4、9.4+)

主要 功能

- 使用数据库特定的模式从数据库服务器二进制文件和头文件中提取版本信息。
- 标识包含数据库服务器安装的目录路径。
- 使用数据驱动的文件类型检测自动区分MySQL和MariaDB安装。

SBOM 生成器在跨平台的常见MySQL安装路径中查找安装：

Linux

- `/usr/local/bin/mysqld`
- `/usr/bin/mysqld`
- `/usr/sbin/mysqld`

macOS

- `/usr/local/mysql/include/mysql_version.h (MySQL)`

Windows

- `C:/Program Files/MySQL/MySQL Server/include/mysql_version.h (MySQL)`
- `C:/Program Files (x86)/MySQL/MySQL Server/include/mysql_version.h (MySQL)`

PURL 示例

以下是MySQL服务器的软件包 URL 示例。

```
# Oracle MySQL Server

pkg:generic/mysql/mysql-server@8.0.43
```

Microsoft applications 生态系统集合

支持的微软应用程序

- PowerShell
- NuGet CLI
- Visual Studio Code
- Microsoft Edge
- SharePoint Server
- Microsoft Defender
- Exchange Server

- Visual Studio
- .NET Runtime
- ASP.NET Core Runtime
- Microsoft Teams
- Outlook for Windows
- Microsoft Office
- Microsoft 365

主要 功能

- PowerShell— 检查`pwsh.exe`文件以提取嵌入式版本信息。
- NuGet CLI— 检查`nuget.exe`文件以提取嵌入式版本信息。
- Visual Studio Code— 检查`Code.exe`文件以提取嵌入式版本信息。
- Microsoft Edge— 检查`msedge.exe`文件以提取嵌入式版本信息。
- SharePoint Server— 检查`Microsoft.SharePoint.dll`文件以提取嵌入式版本信息。
- Microsoft Defender— 检查`MsMpEng.exe`文件以提取嵌入式版本信息。
- Exhange Server— 检查`Exsetup.exe`文件以提取嵌入式版本信息。
- Visual Studio— 解析`state.json`文件以从`catalogInfo.productDisplayVersion`字段中检索版本字符串。
- .NET Runtime— 在安装路径中搜索`Microsoft.NETCore.App.deps.json`文件并从以下文件路径模式中提取版本字符串。

```
Microsoft.NETCore.App/<VERSION>/Microsoft.NETCore.App.deps.json
```

- ASP.NET Runtime— 在安装路径中搜索`Microsoft.AspNetCore.App.deps.json`文件并从以下文件路径模式中提取版本字符串。

```
Microsoft.AspNetCore.App/<VERSION>/Microsoft.AspNetCore.App.deps.json
```

- Outlook for Windows— 解析 Windows 注册表，并从以下注册表项中提取版本。

```
HKLM\SOFTWARE\Classes\Local Settings\Software\Microsoft
\Windows\CurrentVersion\AppModel\PackageRepository\Packages
\Microsoft.OutlookForWindows_<VERSION>_<ARCH>__8wekyb3d8bbwe
```

- Microsoft Teams— 解析 Windows 注册表，并从以下注册表项中提取版本。

```
HKLM\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion
\AppModel\PackageRepository\Packages\MSTeams_<VERSION>_<ARCH>__8wekyb3d8bbwee
```

- Microsoft Office 365 / Microsoft 365— 解析 Windows 注册表，并从以下注册表项和值中提取版本。
  - 注册表密钥

```
KEY_LOCAL_MACHINES\SOFTWARE\Microsoft\Office\ClickToRun\Configuration
```

  - 注册表值
    - VersionToReport— 微软 Office 版本
    - ProductReleaseIds— 产品清单 IDs。这用于识别已安装的 Office 产品。有关产品的更多信息 IDs，请参见product IDsMicrosoft网站。
- Microsoft Office Suite— 通过检查以下可执行文件来收集已安装的每个 Office 应用程序：
  - `EXCEL.EXE` – Microsoft Excel
  - `WINWORD.EXE` – Microsoft Word
  - `POWERPNT.EXE` – Microsoft PowerPoint
  - `OUTLOOK.EXE` – Microsoft Outlook

  Windows 注册表中的版本号用作每个已安装的 Office 应用程序的权威版本号。

示例 **`state.json`** 文件

以下是用于收集已安装Visual Studio版本`state.json`的文件示例。

```
{
    "icon": {
        "mimeType": "image/svg+xml",
        "fileName": "product.svg"
    },
    "updateDate": "2025-11-06T05:05:35.6517471Z",
    "installDate": "2025-11-06T05:05:35.6527436Z",
    "enginePath": "C:\\Program Files (x86)\\Microsoft Visual Studio\\Installer\
\resources\\app\\ServiceHub\\Services\\Microsoft.VisualStudio.Setup.Service",
    "installationName": "VisualStudio/17.14.19+36623.8",
    "catalogInfo": {
        "id": "VisualStudio/17.14.19+36623.8",
        "buildBranch": "d17.14",
```

```
        "buildVersion": "17.14.36623.8",
        "localBuild": "build-lab",
        "manifestName": "VisualStudio",
        "manifestType": "installer",
        "productDisplayVersion": "17.14.19",
// truncated
```

示例 PURL

以下是每个软件包的 URL 示例Microsoft Applications。

```
// PowerShell
Sample PURL: pkg:generic/microsoft/powershell@7.5.3

// NuGet CLI
Sample PURL: pkg:generic/microsoft/nuget@6.14.0

// Visual Studio Code
Sample PURL: pkg:generic/microsoft/visualstudiocode@1.104.2

// Microsoft Edge
Sample PURL: pkg:generic/microsoft/edge@140.0.3485.94

// SharePoint Server
Sample PURL: pkg:generic/microsoft/sharepoint@23.38.219.1

// Microsoft Defender
Sample PURL: pkg:generic/microsoft/defender@4.18.23110.3

// Exchange Server
Sample PURL: pkg:generic/microsoft/exchangeserver@15.2.2562.17

// Visual Studio
Sample PURL: pkg:generic/microsoft/visualstudio@17.14.19

// .NET Runtime
Sample PURL: pkg:generic/microsoft/dotnet@8.0.18

// ASP.NET Core Runtime
Sample PURL: pkg:generic/microsoft/aspdotnet@8.0.18

// Microsoft Teams
Sample PURL: pkg:generic/microsoft/teams@25241.203.3947.4411
```

```
// Outlook for Windows
Sample PURL: pkg:generic/microsoft/outlookforwindows@1.2025.916.400

// Microsoft 365 / Office 365
Sample PURL: pkg:generic/microsoft/office@16.0.19127.20264?
product_ids=O365HomePremRetail

// Microsoft Word
Sample PURL: pkg:generic/microsoft/word@16.0.19127.20264

// Microsoft Excel
Sample PURL: pkg:generic/microsoft/excel@16.0.19127.20264

// Microsoft PowerPoint
Sample PURL: pkg:generic/microsoft/powerpoint@16.0.19127.20264

// Microsoft Outlook
Sample PURL: pkg:generic/microsoft/outlook@16.0.19127.20264
```

Nginx 生态系统集合

受支持的应用程序

- Nginx

支持的平台

以下是支持的平台。

Linux

- /usr/sbin/nginx
- /usr/local/nginx
- /usr/local/etc/nginx
- /usr/local/nginx/nginx
- /usr/local/nginx/sbin/nginx
- /etc/nginx/nginx

Windows

- C:\nginx\nginx.exe
- C:\nginx-x.y.z\nginx.exe（x.y.z 为任意版本）

macOS

- /usr/local/etc/nginx/nginx

主要 功能

该集合可检查二进制文件，以提取嵌入式版本信息。它会在二进制可执行文件 `.rodata` 部分（针对 Linux 上的 ELF 二进制文件）、`.rdata` 部分（针对 Windows 上的 PE 二进制文件）或 `__cstring` 部分（针对 MachO 二进制文件）中搜索版本字符串。

示例版本字符串

以下是嵌入在 Nginx 二进制文件中的版本字符串的示例。

```
nginx version: nginx/1.27.5
```

版本 `1.27.5` 被提取以识别 Nginx 版本。

PURL 示例

以下是 Nginx 的示例程序包 URL。

```
Sample PURL: pkg:generic/nginx/nginx@1.27.5
```

Node.JS 运行时集合

受支持的应用程序

- Node.JS 的节点运行时二进制文件

支持的平台

以下是支持的平台。（* 为任意版本）

Linux

- /usr/local/bin/node
- /usr/bin/node
- /nodejs/bin/node
- ~/.nvm/versions/node/*/bin/node
- ~/.local/share/fnm/node-versions/*/installation/bin/node
- ~/.asdf/installs/nodejs/*/bin/node
- ~/.local/share/mise/installs/node/*/bin/node
- ~/.volta/tools/image/node/*/bin/node

Windows

- C:\Program Files\ n odejs\ n ode.exe
- C:\Program Files (x86)\ n odejs\ n ode.exe
- ~\AppData\ Roaming\ fnm\ n 节点版本\ *\ 安装\ n ode.exe

macOS

- /opt/homebrew/Cellar/node/*/bin/node

主要 功能

该集合可检查二进制文件，以提取嵌入式版本信息。它会在二进制可执行文件 `.rodata` 部分（针对 Linux 上的 ELF 二进制文件）、`.rdata` 部分（针对 Windows 上的 PE 二进制文件）或 `__ctring` 部分（针对 MachO 二进制文件）中搜索版本字符串。

示例版本字符串

以下是嵌入在Node.JS运行时二进制文件中的版本字符串的示例。

```
node.js/v24.11.1
```

提取`24.11.1`版本以标识Node.JS运行时版本。

PURL 示例

以下是 Node.JS 的示例程序包 URL。

```
Sample PURL: pkg:generic/nodejs/node@24.11.1
```

OpenSSH 生态系统集合

受支持的应用程序

- OpenSSH（版本 9）
- OpenSSH（版本 10）

支持的平台 Linux/macOS

- `/usr/sbin/sshd`
- `/usr/local/sbin/sshd`

支持的平台 Windows

- `C:/Windows/System32/OpenSSH/sshd.exe`
- `C:/Program Files/OpenSSH/sshd.exe`
- `C:/Program Files (x86)/OpenSSH/sshd.exe`
- `C:/OpenSSH/sshd.exe`

主要 功能

- 检查 `sshd` 二进制文件以提取嵌入的版本信息。
- 查找二进制可执行文件 `.rodata` 部分（针对 Linux 上的 ELF 二进制文件）、`__cstring` 部分（针对 MacOs 上的 Mach-O 二进制文件）或 `.rdata` 部分（针对 Windows 上的 PE 二进制文件）中的版本字符串。

示例版本字符串

以下是嵌入在 OpenSSH 二进制文件中的版本字符串的示例。

```
OpenSSH_9.9p2
```

版本 9.9p2 被提取以识别 OpenSSH 版本。

PURL 示例

以下是 OpenSSH 的示例程序包 URL。

```
Sample PURL: pkg:generic/openssh/openssh@9.9p2
```

OpenSSL 生态系统集合

受支持的应用程序

对 OpenSSL 库和开发程序包的支持仅限于使用官方 OpenSSL 为 3.0.0 及以上版本构建的软件。该软件还必须遵循语义版本控制。不支持自定义或分支的 OpenSSL 版本以及低于 3.0.0 的版本。

Amazon Inspector SBOM 生成器为每个已安装的 OpenSSL 实例提取关键程序包信息。

主要 功能

- 从 OpenSSL 标头文件中提取基本 SEMVER 版本字符串
- 识别包含 OpenSSL 安装的目录路径

Amazon Inspector SBOM 生成器通过扫描各平台上的常见安装路径中的 `openssl.h` 文件来查找 OpenSSL 安装。

Linux/Unix 的示例安装路径

以下是 Linux/Unix 的示例安装路径。

```
/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 生成器通过解析 `openssl.h` 文件并查找版本定义来提取版本信息。

```
# define OPENSSL_VERSION_MAJOR  3
# define OPENSSL_VERSION_MINOR  4
```

```
# define OPENSSL_VERSION_PATCH  0
```

PURL 示例

以下是 OpenSSL 版本的示例程序包 URL。

```
Sample PURL: pkg:generic/openssl/openssl@3.4.0
```

Oracle 数据库服务器集合

受支持的应用程序

- Oracle Database

支持的平台 Linux

- `/opt/oracle`
- `/u01/app/oracle`

> **Note**
>
> 漏洞评估仅适用于 Oracle 数据库服务器版本 19 及更高版本。

主要 功能

- 检查 Oracle 二进制文件以提取嵌入的版本信息。
- 在二进制可执行文件 `.rodata` 部分（针对 Linux 上的 ELF 二进制文件）中查找版本字符串。
- 版本信息遵循包含 RDBMS 版本字符串的特定格式。

示例版本字符串

以下是嵌入在 Oracle Database 二进制文件中的版本字符串的示例：

```
RDBMS_23.7.0.25.01DBRU_LINUX.X64_240304
```

版本 `23.7.0.25.01` 被提取以识别 Oracle Database 版本。

PURL 示例

以下是 Oracle Database 的示例程序包 URL。

```
Sample PURL: pkg:generic/oracle/database@23.7.0.25.01
```

PHP 生态系统集合

受支持的应用程序

- PHP（版本 8.1 及更高版本）

主要 功能

- 使用嵌入式版本字符串从PHP二进制可执行文件中提取版本信息。
- 标识包含PHP二进制文件的目录路径。
- 自动检测标准PHP二进制文件和版本化安装，例如`php8.1`、`php8.2`和。`php8.3`

Amazon Inspector SBOM 生成器在跨平台的常见安装路径中查找安装：PHP

Linux

- `usr/bin/php8.1 through /usr/bin/php8.9`
- `/usr/sbin/php8.1 through /usr/sbin/php8.9`
- `/usr/local/bin/php, /usr/bin/php, /usr/sbin/php`
- `/usr/local/bin/php8.1 through /usr/local/bin/php8.9`（版本化二进制文件）

macOS

- `/opt/homebrew/bin/php`
- `/usr/bin/php`
- `/usr/local/bin/php`

Windows

- `C:/php/php.exe`

- C:/php8.1/php.exe through C:/php8.9/php.exe（版本化目录）

PHP版本提取示例

Amazon Inspector SBOM 生成器使用以下模式搜索嵌入式版本字符串，从而从PHP二进制文件中提取版本信息。

```
X-Powered-By: PHP/8.4.12
```

8.4.12是从此模式中提取的，用于标识PHP版本。

PURL 示例

以下是PHP模式的包网址示例。

```
pkg:generic/php/php@8.4.12
```

WordPress 生态系统集合

支持的组件

- WordPress 核心
- WordPress 插件
- WordPress 主题

主要 功能

- WordPress 核心 – 解析 /wp-includes/version.php 文件以从 $wp_version 变量中提取版本值。
- WordPress 插件 – 解析 /wp-content/plugins/<WordPress Plugin>/readme.txt 文件或 /wp-content/plugins/<WordPress Plugin>/readme.md 文件以提取 Stable 标签作为版本字符串。
- WordPress 主题 – 解析 /wp-content/themes/<WordPress Theme>/style.css 文件以从版本元数据中提取版本。

示例 **version.php** 文件

以下是 WordPress 核心 `version.php` 文件的示例。

```
// 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 示例

以下是 WordPress 核心的示例程序包 URL。

```
Sample PURL: pkg:generic/wordpress/core/wordpress@6.5.5
```

示例 **`readme.txt`** 文件

以下是 WordPress 插件 `readme.txt` 文件的示例。

```
=== 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 示例

以下是 WordPress 插件的示例程序包 URL。

```
Sample PURL: pkg:generic/wordpress/plugin/exclusive-addons-for-elementor@1.0.0
```

示例 **`style.css`** 文件

以下是 WordPress 主题 `style.css` 文件的示例。

```
/*
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 示例

以下是 WordPress 主题的示例程序包 URL。

```
Sample PURL: pkg:generic/wordpress/theme/avada@1.0.0
```

亚马逊 Inspector SBOM 生成器 SSL/TLS 证书扫描

本节介绍如何使用 Amazon Inspector SBOM 生成器来清点 SSL/TLS 证书。通过在预定义的位置以及用户提供的目录中搜索证书来Sbomgen清 SSL/TLS 点证书。该功能旨在使用户能够清点 SSL/TLS 证书并识别过期的证书。CA 证书也将出现在输出清单中。

使用 Sbomgen 证书扫描

您可以使用`--scanners certificates`参数启用 SSL/TLS 证书清单收集。证书扫描可以与其他任何扫描器结合使用。默认情况下，不启用证书扫描。

根据所扫描的构件，Sbomgen 会在不同的位置搜索证书。在所有情况下，Sbomgen 都会尝试提取具有以下扩展名的文件中的证书。

```
.pem
.crt
.der
.p7b
.p7m
.p7s
.p12
.pfx
```

本地主机构件类型

如果启用了证书扫描器且构件类型为本地主机，则 Sbomgen 会以递归方式在 `/etc/*/ssl`、`/opt/*/ssl/certs`、`/usr/local/*/ssl` 和 `/var/lib/*/certs` 中查找证书，其中 `*` 不为空。无论命名什么目录，都将以递归方式搜索用户提供的目录。通常， CA/system 证书不会放在这些路径中。这些证书通常位于名为 `pki`、`ca-certs` 或 `CA` 的文件夹中。它们也可能出现在默认的本地主机扫描路径中。

目录和容器构件

扫描目录或容器构件时，Sbomgen 会搜索位于构件上任意位置的证书。

证书扫描命令示例

以下内容包含一个证书扫描命令示例。一个生成的 SBOM 仅包含本地目录中的证书。另一个生成的 SBOM 包含证书以及本地目录中的 Alpine、Debian 和 Rhel 程序包。另一个生成的 SBOM 包含在常见证书位置中找到的证书。

```
# generate SBOM only containing certificates in a local directory
./inspector-sbomgen directory --path ./project/ --scanners certificates

# generate SBOM only containing certificates and Alpine, Debian, and Rhel OS packages
 in a local directory
./inspector-sbomgen directory --path ./project/ --scanners certificates,dpkg,alpine-
apk,rhel-rpm

# generate SBOM only containing certificates, taken from common localhost certificate
 locations
./inspector-sbomgen localhost --scanners certificates
```

示例文件组件

以下内容包含两个证书查找组件示例。证书到期后，您可以查看标识到期日期的额外属性。

```
{
      "bom-ref": "comp-2",
      "type": "file",
      "name": "certificate:expired.pem",
      "properties": [
            {
                "name": "amazon:inspector:sbom_generator:certificate_finding:IN-
CERTIFICATE-001",
                "value": "expired:2015-06-06T11:59:59Z"
            },
            {
                "name": "amazon:inspector:sbom_generator:source_path",
                "value": "/etc/ssl/expired.pem"
            }
      ]
},
{
      "bom-ref": "comp-3",
      "type": "file",
      "name": "certificate:unexpired.pem",
      "properties": [
            {
                "name": "amazon:inspector:sbom_generator:source_path",
```

```
                "value": "/etc/ssl/unexpired.pem"
            }
      ]
}
```

漏洞响应组件示例

运行带有 `--scan-sbom` 标志的 Amazon Inspector SBOM 生成器会将生成的 SBOM 发送到 Amazon Inspector 进行漏洞扫描。以下是证书查找漏洞响应组件的示例。

```
{
    "advisories": [
        {
            "url": "https://aws.amazon.com/inspector/"
        },
        {
            "url": "https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/
sec_protect_data_transit_encrypt.html"
        }
    ],
    "affects": [
        {
            "ref": "comp-2"
        }
    ],
    "analysis": {
        "state": "in_triage"
    },
    "bom-ref": "vuln-1",
    "created": "2025-04-17T18:48:20Z",
    "cwes": [
        324,
        298
    ],
    "description": "Expired Certificate: The associated certificate(s) are no longer
 valid. Replace certificate in order to reduce risk.",
    "id": "IN-CERTIFICATE-001",
    "properties": [
        {
            "name": "amazon:inspector:sbom_scanner:priority",
            "value": "standard"
        },
        {
```

```
            "name": "amazon:inspector:sbom_scanner:priority_intelligence",
            "value": "unverified"
        }
    ],
    "published": "2025-04-17T18:48:20Z",
    "ratings": [
        {
            "method": "other",
            "severity": "medium",
            "source": {
                "name": "AMAZON_INSPECTOR",
                "url": "https://aws.amazon.com/inspector/"
            }
        }
    ],
    "source": {
        "name": "AMAZON_INSPECTOR",
        "url": "https://aws.amazon.com/inspector/"
    },
    "updated": "2025-04-17T18:48:20Z"
}
```

Amazon Inspector SBOM 生成器许可证集合

Amazon Inspector SBOM 生成器可帮助跟踪软件物料清单（SBOM）中的许可证信息。它收集来自不同操作系统和编程语言的支持程序包的许可证信息。通过生成的 SBOM 中的标准化许可证表达式，您可以了解自己的许可义务。

收集许可证信息

示例命令

以下示例演示了如何收集目录中的许可证信息。

```
./inspector-sbomgen directory --path /path/to/your/directory/ --collect-licenses
```

SBOM 组件示例

以下示例显示了生成的 SBOM 中的组件条目。

```
"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",
    }
  ]
```

支持的程序包

许可证集合支持以下编程语言和操作系统程序包。

目标	软件包管理器	许可证信息源	Type
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	OS

目标	软件包管理器	许可证信息源	Type
		• /usr/lib/sysimage/rpm/Packages.db • /var/lib/rpm/rpmdb.sqlite • /var/lib/rpm/Packages • /var/lib/rpm/Packages.db	
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

目标	软件包管理器	许可证信息源	Type
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
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

目标	软件包管理器	许可证信息源	Type
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
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

目标	软件包管理器	许可证信息源	Type
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
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

目标	软件包管理器	许可证信息源	Type
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
Debian	DPKG	`/usr/share/doc/*/copyright`	OS
Ubuntu	DPKG	`/usr/share/doc/*/copyright`	OS
Node.js	Javascript	`node_modules/*/package.json`	编程语言
PHP	编辑器程序包	• `composer.lock` • `/vendor/composer/installed.json`	编程语言
Go	Go	`LICENSE`	编程语言

目标	软件包管理器	许可证信息源	Type
Python	Python/Egg/Wheel	• `.dist-info/METADATA` • `.egg-info` • `.egg-info/PKG-INFO`	编程语言
Ruby	RubyGem	`*.gemspec`	编程语言
Rust	crate	`Cargo.toml`	编程语言

许可证表达式标准化

SPDX 许可证表达式格式提供了开源软件中许可条款的准确表示形式。Amazon Inspector SBOM 生成器通过本节中描述的规则将所有许可证信息标准化为 SPDX 许可证表达式。这些规则提供了许可信息的一致性和兼容性。

SPDX 短格式标识符映射

所有许可证名称均映射到 SPDX 短格式标识符。例如，`MIT License` 缩短为 `MIT`。

多个许可证组合

您可以将多个许可证与 `AND` 运算符结合使用。以下示例命令展示了如何格式化处理您的命令。

```
MIT AND Apache-2.0
```

自定义许可证前缀

自定义许可证的前缀为 `LicenseRef`，例如 `LicenseRef-CompanyPrivate`。

自定义异常前缀

自定义异常的前缀为 `AdditionRef-`，例如 `AdditionRef-CustomException`。

什么是程序包 URL？

程序包 URL 或 PURL 是一种标准化格式，用于在不同的程序包管理系统中识别软件程序包、组件和库。该格式使跟踪、分析和管理软件项目中的依赖关系变得更加容易，尤其是在生成软件物料清单 (SBOMs) 时。

PURL 结构

PURL 结构类似于 URL，由多个部分组成：

- `pkg` – 文本前缀
- `type` – 程序包类型
- `namespace` – 分组
- `name` – 程序包名称
- `version` – 程序包版本
- `qualifiers` – 额外的键值对
- `subpath` – 程序包中的文件路径

PURL 示例

以下是 PURL 可能呈现形式的示例。

```
pkg:<type>/<namespace>/<name>@<version>?<qualifiers>#<subpath>
```

通用 PURL

通用 PURL 用于表示不属于已建立程序包生态系统的软件程序包和组件，例如 npm、pypi 或 maven。它可以识别软件组件，并捕获可能与特定程序包管理系统不一致的元数据。通用 PURL 可用于各种软件项目，从已编译的二进制文件到平台，不一而足，例如 Apache 和 WordPress。可将其应用于各种使用案例，包括编译后的二进制文件、Web 平台和自定义软件发行版。

关键使用案例

- 支持编译后的二进制文件，对 Go 和 Rust 很有用
- 支持 Web 平台，例如 Apache 和 WordPress，其中程序包可能与传统程序包管理器无关联。

- 通过允许组织参考内部开发的软件或缺少正式程序包的系统，从而支持自定义旧版软件。

格式示例

以下是通用 PURL 格式的示例。

```
pkg:generic/<namespace>/<name>@<version>?<qualifiers>
```

通用 PURL 格式的其他示例

以下是通用 PURL 格式的其他示例。

编译后的 Go 二进制文件

以下表示使用 Go 编译的 `inspector-sbomgen binary`。

```
pkg:generic/inspector-sbomgen?go_toolchain=1.22.5
```

编译后的 Rust 二进制文件

以下表示使用 Rust 编译的 `myrustapp` 二进制文件。

```
pkg:generic/myrustapp?rust_toolchain=1.71.0
```

Apache 项目

以下指的是 Apache 命名空间下的一个 http 项目。

```
pkg:generic/apache/httpd@1.0.0
```

WordPress 软件

以下指的是核心 WordPress 软件。

```
pkg:generic/wordpress/core/wordpress@6.0.0
```

WordPress 主题

以下指的是自定义 WordPress 主题。

```
pkg:generic/wordpress/theme/mytheme@1.0.0
```

WordPress 插件

以下指的是自定义 WordPress 插件。

```
pkg:generic/wordpress/plugin/myplugin@1.0.0
```

在 Amazon Inspector SBOM 生成器中处理未解析或非标准版本引用

Amazon Inspector SBOM 生成器通过直接从源文件中识别依赖关系来定位和解析系统中支持的构件。它不是程序包管理器，不会解析版本范围、根据动态引用推断版本或处理注册表查询。它仅收集项目源构件中定义的依赖关系。在许多情况下，程序包清单中的依赖关系（例如 `package.json`、`pom.xml` 或 `requirements.txt`）是使用未解析或基于范围的版本指定的。本主题包括这些依赖关系的外观示例。

建议

Amazon Inspector SBOM 生成器从源构件中提取依赖关系，但不解析或解释版本范围或动态引用。为了更准确地扫描漏洞 SBOMs，我们建议在项目依赖项中使用已解析的语义版本标识符。

Java

对于 Java，Maven 项目可以使用版本范围来定义 `pom.xml` 文件中的依赖关系。

```
<dependency>
    <groupId>org.inspector</groupId>
    <artifactId>inspector-api</artifactId>
    <version>(,1.0]</version>
</dependency>
```

该范围指定任何不超过 1.0（含）的版本均可接受。但是，如果某个版本不是已解析的版本，则 Amazon Inspector SBOM 生成器将不会收集该版本，因为它无法映射到特定版本。

JavaScript

对于 JavaScript，`package.json` 文件可以包含类似于以下内容的版本范围：

```
"dependencies": {
    "ky": "^1.2.0",
    "registry-auth-token": "^5.0.2",
    "registry-url": "^6.0.1",
    "semver": "^7.6.0"
}
```

^ 运算符指定大于等于指定版本的任何版本均可接受。但是，如果指定的版本不是已解析的版本，则 Amazon Inspector SBOM 生成器将不会收集该版本，因为这样做可能会导致漏洞检测期间出现误报。

Python

对于 Python，`requirements.txt` 文件可以包含带有布尔表达式的条目。

```
requests>=1.0.0
```

>= 运算符指定任何大于或等于 `1.0.0` 的版本均可接受。由于此特定表达式未指定确切的版本，因此 Amazon Inspector SBOM 生成器无法可靠地收集用于漏洞分析的版本。

Amazon Inspector SBOM 生成器不支持非标准或不确定的版本标识符，例如测试版、最新版或快照。

```
pkg:maven/org.example.com/testmaven@1.0.2%20Beta-RC-1_Release
```

> **Note**
>
> 使用非标准后缀（例如 Beta-RC-1_Release）不符合标准语义版本控制，并且无法在 Amazon Inspector 检测引擎中评估其漏洞。

将 CycloneDX 命名空间与 Amazon Inspector 配合使用

Amazon In CycloneDX spector 为您提供可与之配合使用的命名空间和属性名称。 SBOMs本节介绍中可能添加到组件中的所有自定义 key/value 属性CycloneDX SBOMs。有关更多信息，请参阅 GitHub 网站上的 CycloneDX property taxonomy。

amazon:inspector:sbom_scanner 命名空间分类

Amazon Inspector Scan API 使用 amazon:inspector:sbom_scanner 命名空间并具有以下属性：

属性	描述
amazon:inspector:sbom_scanner:cisa_kev_date_added	表示相应漏洞何时被添加到 CISA 已知被利用的漏洞目录中。
amazon:inspector:sbom_scanner:cisa_kev_date_due	表示根据 CISA 已知被利用的漏洞目录，漏洞修复程序的到期时间。
amazon:inspector:sbom_scanner:critical_vulnerabilities	在 SBOM 中发现的严重性为“严重”的漏洞总数。
amazon:inspector:sbom_scanner:exploit_available	表示是否存在针对给定漏洞的漏洞利用。
amazon:inspector:sbom_scanner:exploit_last_seen_in_public	表示针对给定漏洞的漏洞利用最后一次公开出现的时间。
amazon:inspector:sbom_scanner:fixed_version: *component_bom_ref*	为给定漏洞提供指定组件已修复漏洞的版本。
amazon:inspector:sbom_scanner:high_vulnerabilities	在 SBOM 中发现的严重性为“高”的漏洞总数。
amazon:inspector:sbom_scanner:info	为给定组件提供扫描上下文，例如：“已扫描组件：未发现漏洞。”
amazon:inspector:sbom_scanner:is_malicious	指示 OpenSSF 是否将受影响的组件识别为恶意组件。
amazon:inspector:sbom_scanner:low_vulnerabilities	在 SBOM 中发现的严重性为“低”的漏洞总数。
amazon:inspector:sbom_scanner:medium_vulnerabilities	在 SBOM 中发现的严重性为“中”的漏洞总数。

属性	描述
amazon:inspector:sbom_scanner:path	生成主题程序包信息的文件的路径。
amazon:inspector:sbom_scanner:priority	修复给定漏洞的建议优先级。值按降序排列，分别为“立即”、“紧急”、“中等”和“标准”。
amazon:inspector:sbom_scanner:priority_intelligence	用于确定给定漏洞优先级的情报质量。这些值包括“已验证”或“未验证”。
amazon:inspector:sbom_scanner:warning	为未扫描给定组件的原因提供上下文，例如：“已跳过组件：未提供 purl。”

amazon:inspector:sbom_generator 命名空间分类

Amazon Inspector SBOM 生成器使用 amazon:inspector:sbom_generator 命名空间并具有以下属性：

属性	描述
amazon:inspector:sbom_generator:cpu_architecture	正在清点的系统的 CPU 架构（x86_64）。
amazon:inspector:sbom_generator:ec2:instance_id	亚马逊 EC2 实例 ID。
amazon:inspector:sbom_generator:ec2:instance_type	Amazon EC2 实例类型
amazon:inspector:sbom_generator:live_patching_enabled	一个布尔值，表示是否在 Amazon A EC2 ma Linux zon 上启用了实时补丁。
amazon:inspector:sbom_generator:live_patched_cves	在 Amazon Ama EC2 zon CVEs 上通过实时补丁进行补丁的列表。Linux

属性	描述
amazon:inspector:sbom_generator:dockerfile_finding: *inspector_finding_id*	指示组件中的 Amazon Inspector 调查发现与 Dockerfile 检查有关。
amazon:inspector:sbom_generator:image_id	属于容器映像配置文件的哈希（也称为映像 ID）。
amazon:inspector:sbom_generator:image_arch	容器映像的架构。
amazon:inspector:sbom_generator:image_author	容器映像的作者。
amazon:inspector:sbom_generator:image_docker_version	用于构建容器映像的 Docker 版本。
amazon:inspector:sbom_generator:is_duplicate_package	表示主题程序包是由多个文件扫描器找到的。
amazon:inspector:sbom_generator:duplicate_purl	表示由另一个扫描器找到的重复程序包 PURL。
amazon:inspector:sbom_generator:kernel_name	正在清点的系统的内核名。
amazon:inspector:sbom_generator:kernel_version	正在清点的系统的内核版本。
amazon:inspector:sbom_generator:kernel_component	一个布尔值，表示主题程序包是否为内核组件
amazon:inspector:sbom_generator:running_kernel	一个布尔值，表示主题程序包是否为正在运行的内核
amazon:inspector:sbom_generator:layer_diff_id	未压缩的容器映像层的哈希值。

属性	描述
amazon:inspector:sbom_generator:replaced_by	替换当前 Go 模块的值。
amazon:inspector:sbom_generator:os_hostname	正在清点的系统的主机名。
amazon:inspector:sbom_generator:source_file_scanner	找到了包含程序包信息的文件的扫描器，例如：/var/lib/dpkg/status 。
amazon:inspector:sbom_generator:source_package_collector	从特定文件中提取了程序包名称和版本的收集器。
amazon:inspector:sbom_generator:source_path	从中提取了主题程序包信息的文件的路径。
amazon:inspector:sbom_generator:file_size_bytes	表示给定构件的文件大小。
amazon:inspector:sbom_generator:unresolved_version	表示尚未被程序包管理器解析的版本字符串。
amazon:inspector:sbom_generator:experimental:transitive_dependency	表示来自程序包管理器的间接依赖关系。
amazon:inspector:sbom_generator:metadata:host:hostname	已扫描系统的主机名。
amazon:inspector:sbom_generator:metadata:host:kernel_name	操作系统的内核名称（例如 Linux、Darwin、Windows_NT）。
amazon:inspector:sbom_generator:metadata:host:kernel_version	操作系统的内核版本字符串。

属性	描述
amazon:inspector:sbom_generator:metadata:host:cpu_architecture	系统的 CPU 架构（例如 x86_64、arm64）。
amazon:inspector:sbom_generator:metadata:host:bootdisk_id	启动磁盘的唯一标识符。
amazon:inspector:sbom_generator:metadata:host:boot_id	当前启动会话的唯一标识符。
amazon:inspector:sbom_generator:metadata:host:boot_time	系统启动时间，采用 ISO 8601 格式。
amazon:inspector:sbom_generator:metadata:host:system_id	永久系统标识符（在 Linux 上和 Windows MachineGuid 上使用机器 ID）。
amazon:inspector:sbom_generator:metadata:host:system_serial	系统固件中的硬件序列号。
amazon:inspector:sbom_generator:metadata:host:network_interfaces: *name*:hardware	网络接口的 MAC 地址。
amazon:inspector:sbom_generator:metadata:host:network_interfaces: *name*:ipv4	IPv4 分配给接口的地址。
amazon:inspector:sbom_generator:metadata:host:network_interfaces: *name*:ipv6	IPv6 分配给接口的地址。
amazon:inspector:sbom_generator:metadata:host:sbomgen_tag: *key*	通过--tag CLI 参数传递的自定义用户定义标签。

属性	描述
amazon:inspector:sbom_generator:metadata:imds:provider	云提供商通过 IMDS（aws、azure）检测到。
amazon:inspector:sbom_generator:metadata:imds:instance_id	亚马逊实 EC2 例 ID 或 Azure 虚拟机名称。
amazon:inspector:sbom_generator:metadata:imds:instance_type	实例类型（例如 t3.micro、standard_d2s_v3）。
amazon:inspector:sbom_generator:metadata:imds:instance_location	实例 region/location 的。
amazon:inspector:sbom_generator:metadata:imds:instance_partition	云分区（aws、aws-cn、 aws-us-gov for 或 AzurePublicCloud Azure 分区）。 AWS
amazon:inspector:sbom_generator:metadata:imds:instance_managed_id	Amazon EC2 Systems Manager 托管实例 ID（AWS 仅限）。
amazon:inspector:sbom_generator:metadata:imds:tenant_id	Azure 租户 ID（仅限 Azure）。
amazon:inspector:sbom_generator:metadata:imds:vm_id	Azure 虚拟机唯一标识符（仅限 Azure）。
amazon:inspector:sbom_generator:metadata:host:open_port: *port*:*protocol*	表示运行时资源的开放端口（即 EC2）
amazon:inspector:sbom_generator:hardened_image:vendor	强化容器镜像的供应商

将 Amazon Inspector 扫描集成到您的 CI/CD 管道中

Amazon Inspector CI/CD 集成使用 Amazon Inspector SBOM 生成器和亚马逊 Inspector Scan API 为容器映像生成漏洞报告。Amazon Inspector SBOM 生成器可为存档、容器映像、目录、本地系统以及已编译 Go 和 Rust 二进制文件创建软件物料清单（SBOM）。Amazon Inspector Scan API 可扫描 SBOM 来创建一份报告，其中包含有关检测到的漏洞的详细信息。您可以将 Amazon Inspector 容器映像扫描与您的 CI/CD 管道集成，以扫描软件漏洞并生成漏洞报告，这样您就可以在部署之前调查和修复风险。要设置 CI/CD 集成，您可以使用插件或使用 Amazon Inspector SBOM 生成器和 Amazon Inspector Scan API 创建自定义 CI/CD 集成。

主题

插件集成

Amazon Inspector 为支持的 CI/CD 解决方案提供了插件。您可以从相应的市场安装这些插件，然后使用它们将 Amazon Inspector 扫描作为构建步骤添加到管道中。插件构建步骤对您提供的映像运行 Amazon Inspector SBOM 生成器，然后对生成的 SBOM 运行 Amazon Inspector Scan API。

以下概述了如何通过插件实现 Amazon Inspector CI/CD 集成：

1. 您可以将配置为 AWS 账户 允许访问 Amazon Inspector Scan API。有关说明，请参阅设置 AWS 账户以使用 Amazon Inspector CI/CD 集成。
2. 您可以安装市场上提供的 Amazon Inspector 插件。

3. 您可以安装和配置 Amazon Inspector SBOM 生成器二进制文件。有关说明，请参阅Amazon Inspector SBOM 生成器。
4. 您可以将 Amazon Inspector 扫描作为构建步骤添加到您的 CI/CD 管道中，然后配置扫描。
5. 在您运行构建时，该插件会将容器映像作为输入，然后在映像上运行 Amazon Inspector SBOM 生成器，以生成与 CycloneDX 兼容的 SBOM。
6. 然后，该插件将生成的 SBOM 发送到 Amazon Inspector Scan API 端点，该端点会评估每个 SBOM 组件是否存在漏洞。
7. Amazon Inspector Scan API 响应将转换为 CSV、SBOM JSON 和 HTML 格式的漏洞报告。该报告包含有关 Amazon Inspector 发现的任何漏洞的详细信息。

支持的 CI/CD 解决方案

Amazon Inspector 目前支持以下 CI/CD 解决方案。有关使用插件设置 CI/CD 集成的完整说明，请选择适用于 CI/CD 解决方案的插件：

- Jenkins 插件
- TeamCity 插件
- GitHub actions

自定义集成

如果 Amazon Inspector 没有为您的 CI/CD 解决方案提供插件，您可以结合使用 Amazon Inspector SBOM 生成器和亚马逊 Inspector Scan API 来创建自己的自定义 CI/CD 集成。您还可以使用 Amazon Inspector SBOM 生成器中提供的选项，借助自定义集成来微调扫描。

以下概述了自定义 Amazon Inspector CI/CD 集成的工作原理：

1. 您可以将配置为 AWS 账户 允许访问 Amazon Inspector Scan API。有关说明，请参阅设置 AWS 账户以使用 Amazon Inspector CI/CD 集成。
2. 您可以安装和配置 Amazon Inspector SBOM 生成器二进制文件。有关说明，请参阅Amazon Inspector SBOM 生成器。
3. 您可以使用 Amazon Inspector SBOM 生成器为容器映像生成与 CycloneDX 兼容的 SBOM。
4. 您可以对生成的 SBOM 使用 Amazon Inspector Scan API，从而生成漏洞报告。

有关设置自定义集成的说明，请参阅创建与 Amazon Inspector Scan 的自定义 CI/CD 管道集成。

设置 AWS 账户以使用 Amazon Inspector CI/CD 集成

要使用 Amazon Inspector CI/CD 集成，您必须注册 AWS 账户。 AWS 账户 必须具有 IAM 角色，该角色可向你的 CI/CD 管道授予对 Amazon Inspector Scan API 的访问权限。完成以下主题中的任务以注册 AWS 账户、创建管理员用户和配置 IAM 角色以进行 CI/CD 集成。

> **Note**
>
> 如果您已经注册了 AWS 账户，则可以跳至配置 IAM 角色以进行 CI/CD 集成。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS 管理控制台在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 Signing in as the root user。
2. 为您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录 URL。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Create a permission set。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Add groups。

配置 IAM 角色以进行 CI/CD 集成

要将 Amazon Inspector 扫描集成到您的 CI/CD 管道中，您需要创建一个 IAM 策略，允许访问扫描软件物料清单的 Amazon Inspector Scan API（SBOMs）。然后，您可以将该策略附加到 IAM 角色，让您的账户可以运行 Amazon Inspector Scan API。

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择策略，然后选择创建策略。
3. 在策略编辑器中，选择 JSON，然后粘贴下列语句：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "inspector-scan:ScanSbom",
            "Resource": "*"
        }
    ]
}
```

4. 选择下一步。
5. 为策略命名（如 `InspectorCICDscan-policy`），添加可选描述，然后选择创建策略。此策略将附加到后续步骤中创建的角色。
6. 在 IAM 控制台的导航窗格中，依次选择角色和创建新角色。
7. 对于可信实体类型，选择自定义信任策略，然后粘贴以下策略：

   JSON

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

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

8. 选择下一步。
9. 在添加权限页面上，搜索并选择您之前创建的策略，然后选择下一步。
10. 为角色命名（如 InspectorCICDscan-role），添加可选描述，然后选择Create Role。

Amazon Inspector Dockerfile 检查

本节介绍如何使用 Amazon Inspector SBOM 生成器来扫描 Dockerfiles 和 Docker 容器映像，以查找是否存在会引入安全漏洞的错误配置。

主题

- 使用 Sbomgen Dockerfile 检查
- 支持的 Dockerfile 检查

使用 Sbomgen Dockerfile 检查

当发现名为 Dockerfile 或 *.Dockerfile 的文件以及扫描 Docker 映像后，会自动执行 Dockerfile 检查。

您可以使用 --skip-scanners dockerfile 参数禁用 Dockerfile 检查。您还可以将 Dockerfile 检查与任何可用的扫描器（例如操作系统或第三方软件包）结合使用。

Docker 检查命令示例

以下示例命令演示如何为 Dockerfiles 和 Docker 容器镜像以及操作系统和第三方软件包生成 SBOMs 。

```
# 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
```

示例文件组件

以下是 Dockerfile 查找文件组件的示例。

```
{
     "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"
    },
```

漏洞响应组件示例

以下是 Dockerfile 查找漏洞响应组件的示例。

```
{
      "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**
>
> 如果调用 Sbomgen 时不带 `--scan-sbom` 标志，则只能查看原始的 Dockerfile 调查发现。

支持的 Dockerfile 检查

对以下内容支持 Sbomgen Dockerfile 检查：

- Sudo 二进制程序包
- Debian APT 实用程序
- 硬编码密钥
- 根容器
- 运行时弱化命令标志
- 运行时弱化环境变量

这些 Dockerfile 检查中的每一项都有相应的严重性评级，该评级显示在以下主题的顶部。

> **Note**
>
> 以下主题中描述的建议均基于行业最佳实践。

Sudo 二进制程序包

> **Note**
>
> 此检查的严重性评级为信息。

我们建议不要安装或使用 Sudo 二进制程序包，因为它具有不可预测的 TTY 和信号转发行为。有关更多信息，请参阅 Docker Docs 网站中的 User。如果您的使用案例需要类似于 Sudo 二进制程序包的功能，我们建议使用 Gosu。

Debian APT 实用程序

> **Note**
>
> 此检查的严重性评级为高。

以下是使用 Debian APT 实用程序的最佳实践。

将 `apt-get` 命令合并到单个 `Run` 语句中以避免缓存问题

我们建议在 Docker 容器内将 `apt-get` 命令合并到单个 RUN 语句中。单独使用 `apt-get update` 会导致缓存问题且后续 `apt-get install` 指令会失败。有关更多信息，请参阅 Docker Docs 网站上的 apt-get。

> **Note**
>
> 如果 Docker 容器软件已过期，则所述的缓存行为也可能发生在 Docker 容器内。

以非交互方式使用 APT 命令行实用程序

我们建议以交互方式使用 APT 命令行实用程序。APT 命令行实用程序被设计为终端用户工具，其行为在不同版本之间会发生变化。有关更多信息，请参阅 Debian 网站中的 Script Usage and differences from other APT tools。

硬编码机密

> **Note**
>
> 此检查的严重性评级为严重。

您 Dockerfile 中的机密信息被视为硬编码机密。通过 Sbomgen Dockerfile 检查可以识别以下硬编码机密：

- AWS 访问密钥 IDs — `AKIAIOSFODNN7EXAMPLE`
- AWS 密钥 — `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
- DockerHub 个人访问令牌 — `dckr_pat_thisisa27charexample1234567`
- GitHub 个人访问令牌 — `ghp_examplev61wY7Pj1YnotrealUoY123456789`
- GitLab 个人访问令牌 — `glpat-12345example12345678`

根容器

> **Note**
>
> 此检查的严重性标记为信息。

我们建议在没有根权限的情况下运行 Docker 容器。对于没有根权限就无法运行的容器化工作负载，我们建议使用最少权限原则来构建应用程序。有关更多信息，请参阅 Docker Docs 网站中的 User。

运行时弱化环境变量

> **Note**
>
> 此检查的严重性评级为高。

一些命令行实用程序或编程语言运行时支持绕过安全默认值，从而通过不安全的方法执行。

NODE_TLS_REJECT_UNAUTHORIZED=0

当 Node.js 进程在 `NODE_TLS_REJECT_UNAUTHORIZED` 设置为 `0` 的情况下运行时，将禁用 TLS 证书验证。有关更多信息，请参阅 Node.js 网站中的 NODE_TLS_REJECT_UNAUTHORIZED=0。

GIT_SSL_NO_VERIFY=*

当 git 命令行进程在设置 `GIT_SSL_NO_VERIFY` 的情况下运行时，Git 会跳过验证 TLS 证书。有关更多信息，请参阅 Git 网站中的 Environment variables。

PIP_TRUSTED_HOST=*

当 Python pip 命令行进程在设置 `PIP_TRUSTED_HOST` 的情况下运行时，Pip 会跳过验证指定域上的 TLS 证书。有关更多信息，请参阅 Pip 网站中的 --trusted-host。

NPM_CONFIG_STRICT_SSL=false

当 Node.js npm 命令行进程在将 `NPM_CONFIG_STRICT_SSL` 设置为 false 的情况下运行时，Node Package Manager（npm）实用程序将在不验证 TLS 证书的情况下连接到 NPM 注册表。有关更多信息，请参阅 npm Docs 网站中的 strict-ssl。

运行时弱化命令标志

> **Note**
>
> 此检查的严重性评级为高。

与运行时弱化环境变量类似，一些命令行实用程序或编程语言运行时支持绕过安全默认值，从而通过不安全的方法执行。

`npm --strict-ssl=false`

当 Node.js npm 命令行进程结合 `--strict-ssl=false` 标志运行时，Node Package Manager（npm）实用程序将在不验证 TLS 证书的情况下连接到 NPM 注册表。有关更多信息，请参阅 npm Docs 网站中的 strict-ssl。

`apk --allow-untrusted`

当 Alpine Package Keeper 实用程序结合 `--allow-untrusted` 标志运行时，apk 将安装没有签名或签名不可信的软件包。有关更多信息，请参阅 Apline 网站中的以下存储库。

`apt-get --allow-unauthenticated`

当 Debian `apt-get` 软件包实用程序结合 `--allow-unauthenticated` 标志运行时，`apt-get` 不会检查软件包的有效性。有关更多信息，请参阅 Debian 网站中的 APT-Get(8)。

`pip --trusted-host`

当 Python pip 实用程序结合 `--trusted-host` 标志运行时，指定的主机名将绕过 TLS 证书验证。有关更多信息，请参阅 Pip 网站中的 --trusted-host。

`rpm --nodigest, --nosignature, --noverify, --nofiledigest`

当基于 RPM 的软件包管理器 `rpm` 结合 `--nodigest`、`--nosignature`、`--noverify` 和 `--nofiledigest` 标志运行时，RPM 软件包管理器在安装软件包时不会验证软件包标头、签名或文件。有关更多信息，请参阅 RPM 网站上的以下 RPM 手册页：

`yum-config-manager --setopt=sslverify false`

当基于 RPM 的软件包管理器 `yum-config-manager` 在将 `--setopt=sslverify` 标志设置为 false 的情况下运行时，YUM 软件包管理器将不会验证 TLS 证书。有关更多信息，请参阅 Man7 网站上的以下 YUM 手册页：

`yum --nogpgcheck`

基于 RPM 的软件包管理器 `yum` 结合 `--nogpgcheck` 标志运行时，YUM 软件包管理器会跳过检查软件包上的 GPG 签名。有关更多信息，请参阅 Man7 网站上的 yum(8)。

`curl --insecure, curl -k`

当 `curl` 结合 `--insecure` 或 `-k` 标志运行时，将禁用 TLS 证书验证。默认情况下，在进行传输之前，`curl` 建立的每个安全连接都要经过安全验证。此选项会让 `curl` 跳过验证步骤，无需检查即可继续操作。有关更多信息，请参阅 Curl 网站上的以下 Curl 手册页：

`wget --no-check-certificate`

当 `wget` 结合 `--no-check-certificate` 标志运行时，将禁用 TLS 证书验证。有关更多信息，请参阅 GNU 网站上的以下 Wget 手册页：

对容器内操作系统程序包数据库进行移除检查

> **Note**
>
> 此检查的严重性评级为信息。

移除操作系统程序包数据库会降低扫描容器映像的软件完整清单的能力。在执行容器构建步骤期间，这些数据库应保持不变。

以下程序包管理器支持对操作系统程序包数据库进行移除检查：

Alpine Package Keeper（APK）

利用已安装软件的 APK 程序包管理器的容器映像必须确保在构建期间不会移除 APK 系统文件。有关更多信息，请参阅 Arch Linux 网站上的 APK 手册页系统文件文档。

Debian Package Manager（DPKG）

利用 DPKG 程序包管理器的容器（例如基于 Debian、Ubuntu 或 Distroless 的映像）必须确保在容器构建期间不会移除 DPKG 数据库。有关更多信息，请参阅 Ubuntu 网站上的 DPKG 手册页系统文件文档。

RPM Package Manager（RPM）

利用 RPM 包管理器（yum/dnf）的容器（例如 Amazon Linux 或 Red Hat Enterprise Linux）必须确保在构建容器期间不会移除 RPM 数据库。有关更多信息，请参阅 RPM 网站上的 RPM 手册页系统文件文档。

创建与 Amazon Inspector Scan 的自定义 CI/CD 管道集成

如果 Amazon Inspector CI/CD 插件可用于您的 CI/CD 解决方案，我们建议您使用 Amazon In CI/CD spector 插件。如果您的 CI/CD 解决方案没有 Amazon Ins CI/CD pector 插件，则可以结合使用 Amazon Inspector SBOM 生成器和亚马逊 Inspector Scan API 来创建自定义 CI/CD 集成。以下步骤描述了如何创建与 Amazon Inspector Scan 的自定义 CI/CD 管道集成。

> **Tip**
>
> 如果您想通过单个命令生成并扫描 SBOM，则可以使用 Amazon Inspector SBOM 生成器（Sbomgen）跳过第 3 步和第 4 步。

步骤 1：正在配置 AWS 账户

配置一个 AWS 账户 允许访问 Amazon Inspector Scan API 的。有关更多信息，请参阅 设置 AWS 账户以使用 Amazon Inspector CI/CD 集成。

步骤 2：安装 Sbomgen 二进制文件

安装和配置 Sbomgen 二进制文件。有关更多信息，请参阅安装 Sbomgen。

步骤 3：使用 Sbomgen

使用 Sbomgen 为要扫描的容器映像创建 SBOM 文件。

您可以使用以下示例。将 *image:id* 替换为要扫描的映像的名称。将 *sbom_path.json* 替换为要保存 SBOM 输出的位置。

示例

```
./inspector-sbomgen container --image image:id -o sbom_path.json
```

步骤 4：调用 Amazon Inspector Scan API

调用 `inspector-scan` API 以扫描生成的 SBOM 并提供漏洞报告。

您可以使用以下示例。替换*sbom_path.json*为兼容 CycloneDX 的有效 SBOM 文件的位置。*ENDPOINT*替换为您当前进行身份验证 AWS 区域 的 API 端点。*REGION*替换为相应的区域。

示例

```
aws inspector-scan scan-sbom --sbom file://sbom_path.json --endpoint
ENDPOINT-URL --region REGION
```

有关 AWS 区域 和终端节点的完整列表，请参阅区域和终端节点。

（可选）第 5 步。使用单个命令生成和扫描 SBOM

> **Note**
>
> 只有在跳过第 3 步和第 4 步的情况下才需要完成此步骤。

通过单个命令使用 `--scan-bom` 标志生成和扫描 SBOM。

您可以使用以下示例。将 *image:id* 替换为要扫描的映像的名称。*profile*替换为相应的配置文件。*REGION*替换为相应的区域。*/tmp/scan.json*替换为 tmp 目录中 scan.json 文件的位置。

示例

```
./inspector-sbomgen container --image image:id --scan-sbom --aws-profile
profile --aws-region REGION -o /tmp/scan.json
```

有关 AWS 区域 和终端节点的完整列表，请参阅区域和终端节点。

API 输出格式

Amazon Inspector Scan API 能够以 CycloneDX 1.5 格式或 Amazon Inspector 调查发现 JSON 格式输出漏洞报告。可以使用 `--output-format` 标志更改默认值。

CycloneDX 1.5 格式输出示例


```
{
  "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": "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": "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 格式输出示例

```
                        {
  "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": [
          "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": "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"
          }
        ]
      }
    ]
  }
}
```


使用 Amazon Inspector Jenkins 插件

Jenkins 插件利用 Amazon Inspector SBOM 生成器二进制文件和 Amazon Inspector Scan API 在构建结束时生成详细的报告，这样您就可以在部署之前调查和修复风险。通过 Amazon Inspector Jenkins 插件，您可以将 Amazon Inspector 漏洞扫描添加到 Jenkins 管道中。可以根据检测到的漏洞数量和严重性将 Amazon Inspector 漏洞扫描配置为使管道执行通过或失败。你可以在Jenkins市场上查看该Jenkins插件的最新版本，网址为 https://plugins.jenkins.io/amazon-inspector-image-scanner/。以下步骤介绍了如何设置 Amazon Inspector Jenkins 插件。

> **Important**
>
> 在完成以下步骤之前，必须将 Jenkins 升级到 2.387.3 或更高版本才能运行该插件。

步骤 1：设置一个 AWS 账户

AWS 账户 使用允许访问 Amazon Inspector Scan API 的 IAM 角色进行配置。有关说明，请参阅设置 AWS 账户以使用 Amazon Inspector CI/CD 集成。

步骤 2：安装 Amazon Inspector Jenkins 插件

以下过程描述如何从 Jenkins 控制面板安装 Amazon Inspector Jenkins 插件。

1. 在 Jenkins 控制面板中，选择管理 Jenkins，然后选择管理插件。
2. 选择可用。
3. 从可用选项卡中搜索 Amazon Inspector Scan 插件，然后安装该插件。

（可选）第 3 步。将 docker 凭证添加到 Jenkins

> **Note**
>
> 仅当 docker 映像位于私有存储库中时，才添加 docker 凭证。否则，请跳过此步骤。

以下过程描述如何从 Jenkins 控制面板将 docker 凭证添加到 Jenkins 中。

1. 在 Jenkins 控制面板中，依次选择管理 Jenkins、凭证、系统。
2. 选择全局凭证，然后选择添加凭证。
3. 在种类中，选择用户名和密码。
4. 对于范围，选择全局(Jenkins、节点、项目、所有子项目等)。
5. 输入您的详细信息，然后选择确定。

（可选）第 4 步。添加 AWS 凭证

> **Note**
>
> 仅当您想要基于 IAM 用户进行身份验证时，才添加 AWS 证书。否则，请跳过此步骤。

以下过程介绍如何从Jenkins仪表板添加 AWS 凭据。

1. 在 Jenkins 控制面板中，依次选择管理 Jenkins、凭证、系统。
2. 选择全局凭证，然后选择添加凭证。
3. 对于种类，请选择 AWS 凭证。
4. 输入您的详细信息，包括您的访问密钥 ID 和秘密访问密钥，然后选择确定。

步骤 5。在 Jenkins 脚本中添加 CSS 支持

以下过程介绍如何在 Jenkins 脚本中添加 CSS 支持。

1. 重启 Jenkins。
2. 在控制面板中，选择管理 Jenkins、节点、内置节点，然后选择脚本控制台。
3. 在文本框中，添加行 `System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")`，然后选择运行。

步骤 6. 将 Amazon Inspector Scan 添加到您的构建中

您可以通过在项目中添加构建步骤或使用 Jenkins 声明式管道，将 Amazon Inspector 扫描添加到您的构建中。

通过在项目中添加构建步骤将 Amazon Inspector Scan 添加到构建中。

1. 在配置页面上，向下滚动到构建步骤，选择添加构建步骤。然后选择 Amazon Inspector Scan。
2. 在两种 inspector-sbomgen 安装方法之间进行选择：自动或手动。插件可利用自动选项下载最新版本。它还可确保您始终拥有最新功能、安全更新和错误修复程序。

   a. （选项 1）选择自动以下载 inspector-sbomgen 的最新版本。此选项会自动检测当前正在使用的操作系统和 CPU 架构。

   b. （选项 2）如果您要设置 Amazon Inspector SBOM 生成器二进制文件进行扫描，请选择手动。如果您选择这种方法，请确保提供之前下载的 inspector-sbomgen 版本的完整路径。

   有关更多信息，请参阅在 Amazon Inspector SBOM 生成器中安装 Amazon Inspector SBOM 生成器（Sbomgen）。
3. 完成以下操作以完成 Amazon Inspector 扫描构建步骤的配置：

a. 输入映像 ID。映像可以是本地映像、远程映像或归档映像。映像名称应遵循 Docker 命名约定。如果要分析导出的映像，请提供预期的 tar 文件的路径。请参阅下列示例映像 ID 路径：

   i. 对于本地或远程容器：`NAME[:TAG|@DIGEST]`

   ii. 对于 tar 文件：`/path/to/image.tar`

b. 选择用于发送扫描请求的 AWS 区域。

c. （可选）对于报告构件名称，请输入构建过程中生成的构件的自定义名称。这有助于对它们进行唯一标识和管理。

d. （可选）对于跳过文件，请指定要排除在扫描范围之外的一个或多个目录。对于因规模过大而无需扫描的目录，请考虑采用此选项。

e. （可选）对于 Docker 凭证，请选择您的 Docker 用户名。仅当容器映像位于私有存储库中时才执行此操作。

f. （可选）您可以提供以下支持的 AWS 身份验证方法：

   i. （可选）对于 IAM 角色，请提供角色 ARN (arn: aws: iam:: role/)。*AccountNumber RoleName*

   ii. （可选）对于 AWS 证书，请根据 IAM 用户指定要进行身份验证的 AWS 证书。

   iii. （可选）对于 AWS 配置文件名称，请提供要使用配置文件名称进行身份验证的配置文件的名称。

g. （可选）选择启用漏洞阈值。使用此选项，您可以确定如果扫描的漏洞超过某个值，您的构建是否会失败。如果所有值均等于 `0`，则无论扫描了多少漏洞，构建都会成功。对于 EPSS 分数，该值可以介于 0 到 1 之间。如果扫描的漏洞超过某个值，则构建将失败，并且所有 CVEs EPSS 分数高于该值的漏洞都会显示在控制台中。

4. 选择保存。

使用 Jenkins 声明式管道将 Amazon Inspector Scan 添加到构建中。

您可以使用 Jenkins 声明式管道自动或手动将 Amazon Inspector Scan 添加到您的构建中。

自动下载 SBOMGen 声明式管道

- 要将 Amazon Inspector Scan 添加到构建中，请使用以下示例语法。*IMAGE_PATH*替换为映像路径（例如*alpine:latest*）、*IAM_ROLE*您在步骤 1 中配置的 IAM 角色的 ARN，如果您使用的是私有存储库，则替换为您的*ID*Docker凭证 ID。您可以选择启用漏洞阈值并为每个严重性指定值。

```
pipeline {
    agent any
    stages {
        stage('amazon-inspector-image-scanner') {
            steps {
                script {
                step([
                $class:
 'com.amazon.inspector.jenkins.amazoninspectorbuildstep.AmazonInspectorBuilder',
                archivePath: 'IMAGE_PATH', // Path to your container image or tar file
                awsRegion: 'REGION', // AWS region for scan requests
                iamRole: 'IAM ROLE', // IAM role ARN for authentication
                credentialId: 'Id', // Docker credentials (empty if public repo)
                awsCredentialId: 'AWS ID', // AWS credential ID for authentication
                awsProfileName: 'Profile Name', // AWS profile name to use
                sbomgenSkipFiles: '*.log,node_modules,/tmp/*', // Files/directories to
 exclude from scanning

                // Vulnerability threshold settings (updated parameter names)
                isSeverityThresholdEnabled: false, // Enable/disable build failure on
 vulnerability count
                countCritical: 0, // Max critical vulnerabilities before build fails
                countHigh: 0, // Max high vulnerabilities before build fails
                countMedium: 5, // Max medium vulnerabilities before build fails
                countLow: 10, // Max low vulnerabilities before build fails

                // EPSS (Exploit Prediction Scoring System) settings
                isEpssThresholdEnabled: false, // Enable/disable EPSS-based failure
 threshold
                epssThreshold: 0.7, // EPSS score threshold (0.0 to 1.0)

                // NEW FEATURE: CVE Suppression - ignore specific false positives
                isSuppressedCveEnabled: false, // Enable CVE suppression feature
                suppressedCveList: '', // Comma-separated list of CVEs to ignore in
 thresholds

                // NEW FEATURE: Auto-Fail CVEs - always fail on critical security
 issues
                isAutoFailCveEnabled: false, // Enable auto-fail CVE feature
                autoFailCveList: '' // Comma-separated list of CVEs that always fail
 build
                ])
            }
```

```
        }
    }
}
```

手动下载 SBOMGen 声明式管道

- 要将 Amazon Inspector Scan 添加到构建中，请使用以下示例语法。*SBOMGEN_PATH*替换为您在步骤 3 中安装的 Amazon Inspector SBOM 生成器的路径、*IMAGE_PATH*映像的路径（例如*alpine:latest*）、*IAM_ROLE*您在步骤 1 中配置的 IAM 角色的 ARN *ID* 以及Docker您的凭证 ID（如果您使用的是私有存储库）。您可以选择启用漏洞阈值并为每个严重性指定值。

> **Note**
>
> 放Sbomgen在 Jenkins 目录中，并在插件中提供 Jenkins 目录的路径（例如*/opt/folder/arm64/inspector-sbomgen*）。

```
pipeline {
    agent any
    stages {
        stage('amazon-inspector-image-scanner') {
            steps {
                script {
                step([
                $class:
 'com.amazon.inspector.jenkins.amazoninspectorbuildstep.AmazonInspectorBuilder',
                archivePath: 'IMAGE_PATH', // Path to your container image or tar file
                awsRegion: 'REGION', // AWS region for scan requests
                iamRole: 'IAM ROLE', // IAM role ARN for authentication
                credentialId: 'Id', // Docker credentials (empty if public repo)
                awsCredentialId: 'AWS ID', // AWS credential ID for authentication
                awsProfileName: 'Profile Name', // AWS profile name to use
                sbomgenSkipFiles: '*.log,node_modules,/tmp/*', // Files/directories to
 exclude from scanning

                // Vulnerability threshold settings (updated parameter names)
                isSeverityThresholdEnabled: false, // Enable/disable build failure on
 vulnerability count
                countCritical: 0, // Max critical vulnerabilities before build fails
                countHigh: 0, // Max high vulnerabilities before build fails
```

```
                countMedium: 5, // Max medium vulnerabilities before build fails
                countLow: 10, // Max low vulnerabilities before build fails

                // EPSS (Exploit Prediction Scoring System) settings
                isEpssThresholdEnabled: false, // Enable/disable EPSS-based failure
 threshold
                epssThreshold: 0.7, // EPSS score threshold (0.0 to 1.0)

                // NEW FEATURE: CVE Suppression - ignore specific false positives
                isSuppressedCveEnabled: false, // Enable CVE suppression feature
                suppressedCveList: '', // Comma-separated list of CVEs to ignore in
 thresholds

                // NEW FEATURE: Auto-Fail CVEs - always fail on critical security
 issues
                isAutoFailCveEnabled: false, // Enable auto-fail CVE feature
                autoFailCveList: '' // Comma-separated list of CVEs that always fail
 build
                ])
            }
        }
    }
}
```

该插件包含用于管理安全漏洞的功能。

抑制的 CVE 列表

扫描偶尔可能会检测到并非实际威胁的漏洞。为了防止这些误报导致构建停止，可以将它们添加到抑制列表中。

```
isSuppressedCveEnabled: true,
suppressedCveList: 'CVE-2023-1234,CVE-2023-5678'
```

CVEs 在检查您的构建是否应该失败时，这会忽略具体内容。只有在处理了误报之后，才应将它们添加到抑制列表中。将这些漏洞添加到禁止列表后，它们 CVEs 仍会出现在您的安全报告中，但它们不会导致构建失败。

自动失败 CVE 列表

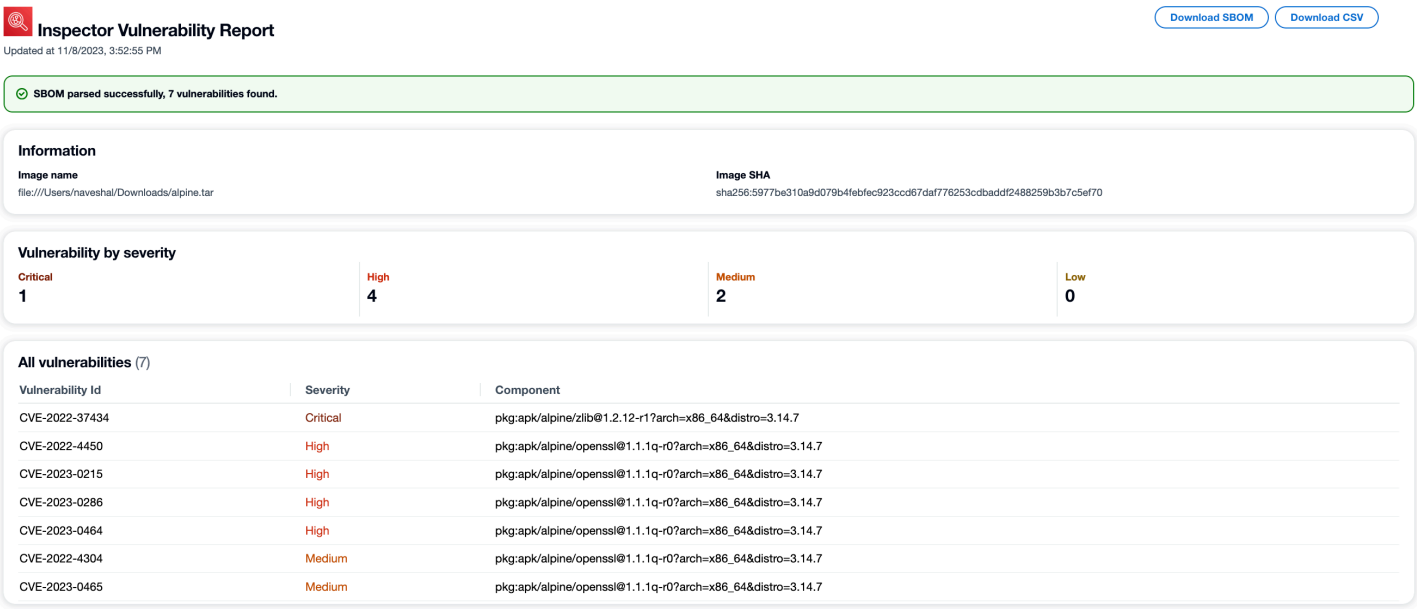
对于关键安全漏洞，您可以创建一个总会导致构建失败的列表。

```
isAutoFailCveEnabled: true,
autoFailCveList: 'CVE-2024-9999'
```

无论启用了哪些设置，此类漏洞总会导致构建失败。应仅为那些永远不应部署的高优先级安全问题创建此列表。该列表会覆盖所有其他阈值设置，以显著提高安全性。

第 7 步。查看 Amazon Inspector 漏洞报告

1. 完成项目的新构建。
2. 构建完成后，从结果中选择一种输出格式。如果选择 HTML 格式，您可以选择下载 JSON SBOM 或 CSV 版本的报告。下面显示了一个 HTML 报告的示例：



> **Note**
>
> 您可以使用较旧的脚本，因为该插件支持旧参数名称。但是，您会在控制台中遇到警告，建议将这些参数更新为较新的参数。例如，如果您使用 `isThresholdEnabled`，将遇到一条警告，建议您将该参数更新为 `isSeverityThresholdEnabled`。

问题排查

以下是您在使用适用于 Jenkins 的 Amazon Inspector Scan 插件时可能遇到的常见错误。

无法加载凭证或 sts 异常错误

错误：

```
InstanceProfileCredentialsProvider(): Failed to load credentials or sts
exception.
```

解决办法

aws_secret_access_key获取aws_access_key_id并使用您的 AWS 帐户。在 ~/.aws/credentials 中设置 aws_access_key_id 和 aws_secret_access_key。

无法从 tarball、本地或远程来源加载映像

错误：

```
2024/10/16 02:25:17 [ImageDownloadFailed]: failed to load image from
tarball, local, or remote sources.
```

> **Note**
>
> 发生以下情况时可能会出现此错误：如果 Jenkins 插件无法读取容器映像、在 Docker 引擎中找不到容器映像，以及在远程容器注册表中找不到容器映像。

解决方法：

请验证以下内容：

- Jenkins 插件用户对您希望扫描的映像具有读取权限。
- 您要扫描的映像存在于 Docker 引擎中。
- 您的远程映像 URL 正确。
- 您已向远程注册表进行身份验证（如果适用）。

Inspector-sbomgen 路径错误

错误：

```
Exception:com.amazon.inspector.jenkins.amazoninspectorbuildstep.exception.Sbomge
There was an issue running inspector-sbomgen, is /opt/inspector/inspector-
sbomgen the correct path?
```

解决方案：

要解决此问题，请完成以下步骤。

1. 将正确的操作系统架构 Inspector-sbomgen 置于 Jenkins 目录中。有关更多信息，请参阅 Amazon Inspector SBOM 生成器。
2. 使用以下命令为该二进制文件授予可执行权限：`chmod +x inspector-sbomgen`。
3. 在插件中提供正确的 Jenkins 机器路径，例如 `/opt/folder/arm64/inspector-sbomgen`。
4. 保存配置，然后执行 Jenkins 作业。

使用 Amazon Inspector TeamCity 插件

Amazon Inspector TeamCity 插件利用 Amazon Inspector SBOM 生成器二进制文件和 Amazon Inspector Scan API 在构建结束时生成详细的报告，这样您就可以在部署之前调查和修复风险。通过 Amazon Inspector TeamCity 插件，您可以将 Amazon Inspector 漏洞扫描添加到 TeamCity 管道中。可以根据检测到的漏洞数量和严重性将 Amazon Inspector 漏洞扫描配置为使管道执行通过或失败。你可以在TeamCity市场上查看 Amazon Inspector TeamCity 插件的最新版本，网址为 https://plugins.jetbrains.com/plugin/23236-。amazon-inspector-scanner有关如何将 Amazon Inspector Scan 集成到您的 CI/CD 管道中的信息，请参阅将 Amazon Inspector 扫描集成到您的 CI/CD 管道中。有关 Amazon Inspector 支持的操作系统和编程语言列表，请参阅支持的操作系统和编程语言。以下步骤介绍了如何设置 Amazon Inspector TeamCity 插件。

1. 设置一个 AWS 账户.
   - AWS 账户 使用允许访问 Amazon Inspector Scan API 的 IAM 角色进行配置。有关说明，请参阅设置 AWS 账户以使用 Amazon Inspector CI/CD 集成。
2. 安装 Amazon Inspector TeamCity 插件。
   a. 在控制面板中，前往管理 > 插件。
   b. 搜索 Amazon Inspector 扫描。
   c. 安装 插件。
3. 安装 Amazon Inspector SBOM 生成器。
   - 在 Teamcity 服务器目录中安装 Amazon Inspector SBOM 生成器二进制文件。有关说明，请参阅安装 Sbomgen。
4. 将 Amazon Inspector 扫描构建步骤添加到项目中。

a. 在配置页面上，向下滚动到构建步骤，选择添加构建步骤，然后选择 Amazon Inspector Scan。

b. 通过填写以下详细信息来配置 Amazon Inspector 扫描构建步骤：

- 添加步骤名称。
- 在两种 Amazon Inspector SBOM 生成器安装方法之间进行选择：自动或手动。
  - 自动方法会根据系统和 CPU 架构下载最新版本的 Amazon Inspector SBOM 生成器。
  - 手动方法会要求您提供之前下载的 Amazon Inspector SBOM 生成器版本的完整路径。

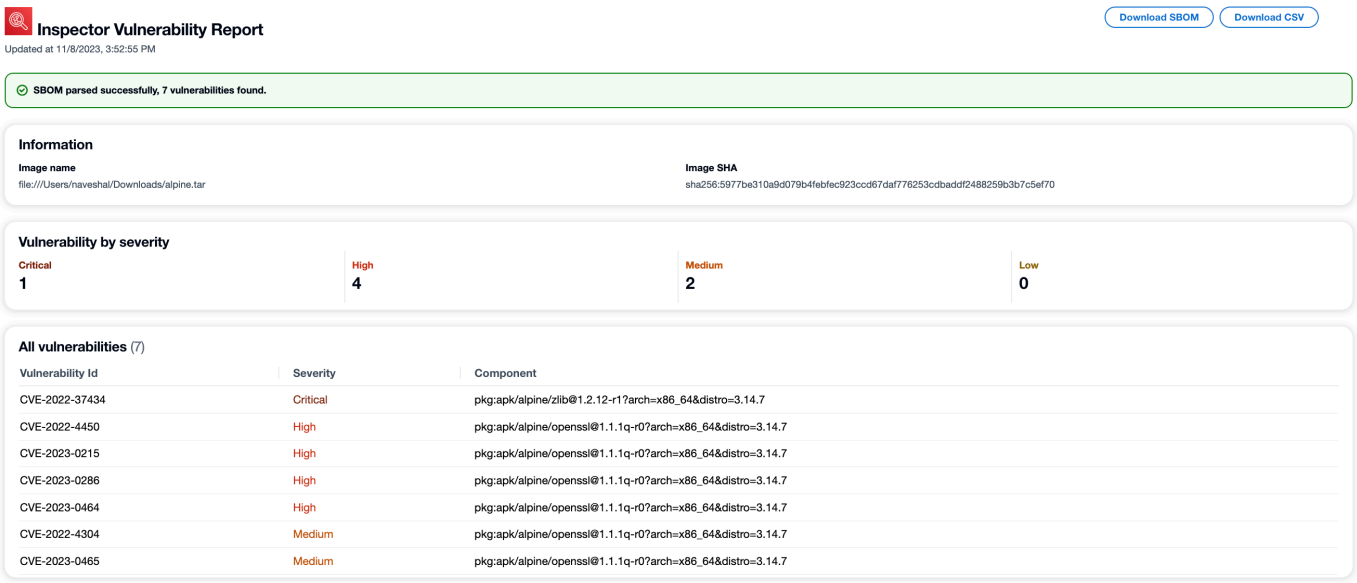
  有关更多信息，请参阅在 Amazon Inspector SBOM 生成器中安装 Amazon Inspector SBOM 生成器（Sbomgen）。
- 输入映像 ID。映像可以是本地映像、远程映像或归档映像。映像名称应遵循 Docker 命名约定。如果要分析导出的映像，请提供预期的 tar 文件的路径。请参阅下列示例映像 ID 路径：
  - 对于本地或远程容器：`NAME[:TAG|@DIGEST]`
  - 对于 tar 文件：`/path/to/image.tar`
- 对于 IAM 角色，输入您在步骤 1 中配置的角色的 ARN。
- 选择用于发送扫描请求的 AWS 区域。
- （可选）对于 Docker 身份验证，请输入您的 Docker 用户名和 Docker 密码。仅当容器映像位于私有存储库中时才执行此操作。
- （可选）对于AWS 身份验证，请输入您的 AWS 访问密钥 ID 和 AWS 密钥。只有在您想要根据 AWS 凭据进行身份验证时才执行此操作。
- （可选）指定每种严重性的漏洞阈值。如果扫描期间的漏洞数超过了您指定的数量，则映像构建将失败。如果值全部为 `0`，则无论发现多少漏洞，构建都将成功。

c. 选择保存。

5. 查看 Amazon Inspector 漏洞报告。

a. 完成项目的新构建。

b. 构建完成后，从结果中选择一种输出格式。如果选择 HTML，您可以选择下载 JSON SBOM 或 CSV 版本的报告。以下是一个 HTML 报告的示例：


Inspector Vulnerability Report

Updated at 11/8/2023, 3:52:55 PM

Download SBOM Download CSV

SBOM parsed successfully, 7 vulnerabilities found.

Information

Image name
file:///Users/naveshal/Downloads/alpine.tar

Image SHA
sha256:5977be310a9d079b4febfec923ccd67daf776253cdbaddf2488259b3b7c5ef70

Vulnerability by severity

Critical	High	Medium	Low
1	4	2	0

All vulnerabilities (7)

Vulnerability Id	Severity	Component
CVE-2022-37434	Critical	pkg:apk/alpine/zlib@1.2.12-r1?arch=x86_64&distro=3.14.7
CVE-2022-4450	High	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7
CVE-2023-0215	High	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7
CVE-2023-0286	High	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7
CVE-2023-0464	High	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7
CVE-2022-4304	Medium	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7
CVE-2023-0465	Medium	pkg:apk/alpine/openssl@1.1.1q-r0?arch=x86_64&distro=3.14.7

结合 Amazon Inspector 使用 GitHub 操作

您可以将 Amazon Inspector 与 GitHub actions 配合使用，来将 Amazon Inspector 漏洞扫描添加到 GitHub 工作流程中。这会利用 Amazon Inspector SBOM 生成器和 Amazon Inspector Scan API 在构建结束时生成详细的报告，这样您就可以在部署之前调查和修复风险。可以根据检测到的漏洞数量和严重性将 Amazon Inspector 漏洞扫描配置为使工作流程通过或失败。您可以在 GitHub 网站上查看最新版本的 Amazon Inspector 操作。有关如何将 Amazon Inspector Scan 集成到您的 CI/CD 管道中的信息，请参阅将 Amazon Inspector 扫描集成到您的 CI/CD 管道中。有关 Amazon Inspector 支持的操作系统和编程语言列表，请参阅支持的操作系统和编程语言。

将 Amazon Inspector 与 GitLab 组件结合使用

你可以使用带有 C GitLab I/CD 组件的 Amazon Inspector，将亚马逊 Inspector 漏洞扫描添加到你的 GitLab项目中。这会利用 Amazon Inspector SBOM 生成器和 Amazon Inspector Scan API 在构建结束时生成详细的报告，这样您就可以在部署之前调查和修复风险。可以根据检测到的漏洞数量和严重性将 Amazon Inspector 漏洞扫描配置为使工作流程通过或失败。您可以在 GitLab 网站上查看最新版本的 Amazon Inspector 组件。有关如何将 Amazon Inspector Scan 集成到您的 CI/CD 管道中的信息，请参阅将 Amazon Inspector 扫描集成到您的 CI/CD 管道中。有关 Amazon Inspector 支持的操作系统和编程语言列表，请参阅支持的操作系统和编程语言。

结合 Amazon Inspector 使用 CodeCatalyst 操作

您可以将 Amazon Inspector 与亚马逊配合使用 CodeCatalyst，将亚马逊 Inspector 漏洞扫描添加到您的 CodeCatalyst 工作流程中。这会利用 Amazon Inspector SBOM 生成器和 Amazon Inspector Scan API 在构建结束时生成详细的报告，这样您就可以在部署之前调查和修复风险。可以根据检测到的漏洞数量和严重性将 Amazon Inspector 漏洞扫描配置为使工作流程通过或失败。有关如何将 Amazon Inspector Scan 集成到您的 CI/CD 管道中的信息，请参阅将 Amazon Inspector 扫描集成到您的 CI/CD 管道中。有关 Amazon Inspector 支持的操作系统和编程语言列表，请参阅支持的操作系统和编程语言。

将 Amazon Inspector 扫描操作与 CodePipeline

您可以通过在工作流程中添加漏洞扫描 AWS CodePipeline 来使用 Amazon Inspector。此集成利用 Amazon Inspector SBOM 生成器和 Amazon Inspector Scan API 在构建结束时生成详细报告。此集成可帮助您在部署之前调查和修复风险。该`InspectorScan`操作是一种托管计算操作 CodePipeline ，可自动检测和修复开源代码中的安全漏洞。您可以对第三方存储库（例如 GitHub 或 Bitbucket Cloud）中的应用程序源代码或容器应用程序的图像使用此操作。有关更多信息，请参阅《AWS CodePipeline 用户指南》中的InspectorScan 调用操作参考。

评估 Amazon Inspector 对您 AWS 环境的覆盖范围

您可以从 Amazon Inspector 控制台的账户管理屏幕中评估 Amazon Inspector 对您 AWS 环境的覆盖范围，该屏幕显示有关您的账户和资源的 Amazon Inspector 扫描状态的详细信息和统计数据。

> **Note**
>
> 如果您是组织的委派管理员，则可以查看组织中所有账户的详细信息和统计数据。

以下过程介绍了如何评测 Amazon Inspector 环境的覆盖率。

评估 Amazon Inspector 对您 AWS 环境的覆盖范围

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择账户管理。
3. 要查看覆盖率，请选择以下选项卡之一：
   - 选择账户可以查看账户级别的覆盖率。
   - 选择实例可查看 Amazon Elastic Compute Cloud（Amazon EC2）的覆盖率。
   - 选择容器存储库，可查看 Amazon Elastic Container Registry（Amazon ECR）的覆盖率。
   - 选择容器映像可查看 Amazon ECR 容器映像的覆盖率。
   - 选择 Lambda 函数可以查看 Lambda 函数的覆盖率。

以下主题介绍了每个选项卡提供的信息。

主题

评测账户级别的覆盖率

如果您的账户不是组织的一员，或者不是组织的 Amazon Inspector 委托管理员账户，则账户选项卡会提供有关您的账户和账户资源扫描状态的信息。在此选项卡上，您可以激活或停用对您账户中所有或仅特定类型资源的扫描。有关更多信息，请参阅 Amazon Inspector 中的自动扫描类型。

如果您的账户是组织的 Amazon Inspector 委托管理员账户，则账户选项卡会提供组织中账户的自动激活设置并列出组织中的所有账户。对于每个账户，该列表都会显示账户是否已激活 Amazon Inspector，如果已激活，还会显示为该账户激活的资源扫描类型。作为委托管理员，您可以使用此选项卡更改组织的自动激活设置。您还可以为个人成员账户激活或停用特定类型的资源扫描。有关更多信息，请参阅 为成员账户激活 Amazon Inspector 扫描。

评测 Amazon EC2 实例的覆盖率

实例选项卡显示环境中 AWS 的 Amazon EC2 实例。列表按以下选项卡分组：

- 全部 – 显示环境中的所有实例。状态列显示实例的当前扫描状态。
- 扫描 – 显示 Amazon Inspector 在环境中主动监控和扫描的所有实例。
- 未扫描 – 显示 Amazon Inspector 未在环境中主动监控和扫描的所有实例。原因列说明了为什么 Amazon Inspector 没有监控和扫描实例。

  EC2 实例可能由于多种原因出现在未扫描选项卡上。Amazon Inspector 使用 AWS Systems Manager (SSM) 和 SSM 代理来自动监控和扫描您的 EC2 实例中是否存在漏洞。如果实例没有运行 SSM 代理，没有支持 Systems Manager 的 AWS Identity and Access Management (IAM) 角色，或者没有运行支持的操作系统或架构，则 Amazon Inspector 无法监控和扫描该实例。有关更多信息，请参阅 Amazon EC2 实例扫描。

在每个选项卡上，账户列都指定 AWS 账户 了拥有实例的。

EC2 实例标签 – 此列显示与实例关联的标签，可用于确定您的实例是否已按标签排除在扫描之外。

操作系统 – 此列显示操作系统类型，可以是 `WINDOWS`、`MAC`、`LINUX` 或 `UNKNOWN`。

使用已监控 – 此列显示 Amazon Inspector 对此实例使用的是基于代理还是无代理的扫描方法。

上次扫描时间 – 此列显示 Amazon Inspector 上次检查资源是否存在漏洞的时间。Amazon Inspector 执行扫描的频率取决于它用来扫描实例的扫描方法。

要查看有关 EC2 实例的更多详细信息，请选择 EC2 实例列中的链接。然后，Amazon Inspector 会显示有关该实例的详细信息以及该实例的当前调查发现。要查看调查发现的详细信息，请选择标题列中的链接。如需了解这些详细信息，请参阅查看 Amazon Inspector 调查发现的详细信息。

扫描 Amazon EC2 实例的状态值

对于 Amazon Elastic Compute Cloud (Amazon EC2) 实例，可能的状态值包括：

- 主动监控 – Amazon Inspector 正在持续监控和扫描实例。
- 超出无代理实例存储限制 - 当连接到实例的所有卷的总大小超过 1200 GB，或者一个实例所连接的卷超过 8 个时，Amazon Inspector 将使用此状态。
- 超出无代理实例收集时间限制 - Amazon Inspector 在尝试对实例运行无代理扫描时超时。
- EC2 实例已停止 – 由于实例处于停止状态，Amazon Inspector 暂停了对实例的扫描。所有现有调查发现都将持续到实例终止。如果实例重新启动，Amazon Inspector 将自动恢复对实例的扫描。
- 内部错误 – Amazon Inspector 尝试扫描实例时发生内部错误。Amazon Inspector 将自动处理错误并尽快恢复扫描。
- 无清单 – Amazon Inspector 找不到用于扫描实例的软件应用程序清单。实例的 Amazon Inspector 关联可能已被删除或者无法运行。

  要修复此问题，请使用 AWS Systems Manager 来确保`InspectorInventoryCollection-do-not-delete`关联存在且其关联状态为成功。此外，使用 AWS Systems Manager Fleet Manager 验证实例的软件应用程序清单。
- 待禁用 – Amazon Inspector 已停止扫描该实例。正在禁用该实例，等待清理任务完成。
- 等待初始扫描 – Amazon Inspector 已将实例纳入队列，等待初始扫描。
- 资源已终止 – 实例已终止。Amazon Inspector 当前正在清理该实例的现有调查发现和覆盖率数据。
- 清单过期 – Amazon Inspector 无法收集过去 7 天内为该实例捕获的更新后的软件应用程序清单。

  要修复此问题，请使用 AWS Systems Manager 来确保该实例所必需的 Amazon Inspector 关联存在且正在运行。此外，使用 AWS Systems Manager Fleet Manager 验证实例的软件应用程序清单。
- 非托管型 EC2 实例 – Amazon Inspector 未监控或扫描实例。实例不由 AWS Systems Manager托管。

  要修复此问题，你可以使用 A AWS Systems Manager utomation AWSSupport-TroubleshootManagedInstance runbook提供的。在您配置 AWS Systems Manager 为管理实例后，Amazon Inspector 将自动开始持续监控和扫描该实例。

- 不支持的操作系统 – Amazon Inspector 未监控或扫描实例。实例使用了 Amazon Inspector 不支持的操作系统或架构。有关 Amazon Inspector 支持的操作系统的列表，请参阅Amazon EC2 实例的状态值。
- 主动监视且存在部分错误 – 此状态表示 EC2 扫描处于活动状态，但存在与 基于 Linux 的 Amazon EC2 实例的 Amazon Inspector 深度检查 相关的错误。可能的深度检查错误包括：
  - 超出深度检查程序包收集限制 – 该实例已超过 Amazon Inspector 深度检查的 5000 个程序包限制。要恢复对此实例的深度检查，您可以尝试调整与该账户关联的自定义路径。
  - 超出深度检查每日 SSM 清单限制 – SSM Agent 无法向 Amazon Inspector 发送清单，因为对于该实例，已达到每天每个实例收集的清单数据的 SSM 配额。有关更多信息，请参阅 Amazon EC2 Systems Manager 端点和配额。
  - 超过深度检查收集时间限制 – Amazon Inspector 未能提取程序包清单，因为程序包收集时间超过了 15 分钟的最大阈值。
  - 深度检查没有清单 – Amazon Inspector SSM 插件尚未能够收集此实例的程序包清单。这通常是待处理扫描的结果，但是，如果此状态在 6 小时后仍然存在，请使用 Amazon EC2 Systems Manager 确保该实例存在所需的 Amazon Inspector 关联并且关联正在运行。

有关为 EC2 实例配置扫描设置的详细信息，请参阅Amazon EC2 实例扫描。

评测 Amazon ECR 存储库的覆盖率

存储库选项卡显示 AWS 环境中的 Amazon ECR 存储库。列表按以下选项卡分组：

- 全部 – 显示环境中的所有存储库。状态列显示存储库的当前扫描状态。
- 已激活 – 显示根据 Amazon Inspector 配置要在环境中监控和扫描的所有存储库。状态列显示存储库的当前扫描状态。
- 已激活 – 显示 Amazon Inspector 未在环境中监控和扫描的所有存储库。原因列说明了为什么 Amazon Inspector 没有监控和扫描存储库。

在每个选项卡上，“帐户” 列指定 AWS 账户 拥有存储库的。

要查看有关存储库的其他详细信息，请选择存储库的名称。然后，Amazon Inspector 会显示存储库中的容器映像的列表以及每个映像的详细信息。详细信息包括映像标签、映像摘要和扫描状态。其中还包括关键调查发现统计数据，例如映像的关键调查发现数量。要深入了解和查看调查发现统计数据的支持数据，请选择映像的映像标签。

> **Note**
>
> 未连续扫描的 Amazon ECR 映像不包含在覆盖小部件中。

扫描 Amazon ECR 存储库的状态值

对于 Amazon Elastic Container Registry（Amazon ECR）存储库，可能的状态值包括：

- 已激活(持续) – 对于存储库，Amazon Inspector 会持续监控存储库中的映像。存储库的增强扫描设置为持续扫描。Amazon Inspector 最初会在推送新映像时对其进行扫描，如果发布了与该映像相关的新 CVE，则会重新扫描映像。Amazon Inspector 将会在您配置的 Amazon ECR 重新扫描持续时间内，继续监控此存储库中的映像。
- 已激活(推送时) – Amazon Inspector 会在推送新映像时自动扫描存储库中的各个容器映像。会为存储库激活增强扫描，并设置为推送时扫描。
- 访问被拒绝 – Amazon Inspector 无法访问存储库或存储库中的任何容器映像。

  要修复此问题，请确保仓库的 AWS Identity and Access Management (IAM) 策略允许 Amazon Inspector 访问存储库。
- 已停用（手动）– Amazon Inspector 未监控或扫描存储库中的任何容器映像。存储库的 Amazon ECR 扫描设置为基本手动扫描。

  要开始使用 Amazon Inspector 扫描存储库中的映像，请将存储库的扫描设置更改为增强扫描，然后选择是持续扫描映像还是仅在推送新映像时扫描映像。
- 已激活(推送时) – Amazon Inspector 会在推送新映像时自动扫描存储库中的各个容器映像。存储库的增强扫描设置为推送时扫描。
- 内部错误 – Amazon Inspector 尝试扫描存储库时发生内部错误。Amazon Inspector 将自动处理错误并尽快恢复扫描。

有关为存储库配置扫描设置的详细信息，请参阅Amazon ECR 容器映像扫描。

评测 Amazon ECR 容器映像的覆盖率

映像选项卡显示 AWS 环境中的 Amazon ECR 容器映像。列表按以下选项卡分组：

- 全部 – 显示环境中的所有容器映像。状态列显示映像的当前扫描状态。

- 正在扫描 – 显示根据 Amazon Inspector 配置要在环境中监控和扫描的所有容器映像。状态列显示映像的当前扫描状态。
- 未扫描 – 显示 Amazon Inspector 未在环境中监控和扫描的所有容器映像。原因列说明了为什么 Amazon Inspector 没有监控和扫描映像。

  容器映像可能会由于多种原因出现在未激活选项卡上。映像可能存储在未激活 Amazon Inspector 扫描的存储库中，或者 Amazon ECR 筛选规则阻止扫描该存储库。或者未在您针对 ECR 重新扫描持续时间配置的天数内推送或拉取映像。有关更多信息，请参阅配置 Amazon ECR 重新扫描持续时间。

在每个选项卡上，存储库名称列指定存储容器映像的存储库的名称。“帐户” 列指定 AWS 账户 拥有存储库的。上次扫描列显示 Amazon Inspector 上次检查资源是否存在脆弱性的时间。这可能包括在更新调查发现元数据时、更新资源的应用程序清单时，或者针对新 CVE 重新扫描时进行检查。有关更多信息，请参阅 Amazon ECR 扫描的扫描行为。

要查看有关容器映像的其他详细信息，请选择 ECR 容器映像列中的链接。然后，Amazon Inspector 会显示有关该映像的详细信息以及该映像的当前调查发现。要查看调查发现的详细信息，请选择标题列中的链接。如需了解这些详细信息，请参阅查看 Amazon Inspector 调查发现的详细信息。

扫描 Amazon ECR 容器映像的状态值

对于 Amazon Elastic Container Registry 容器映像，可能的状态值包括：

- 主动监控(持续) - Amazon Inspector 会持续监控映像，并且每当发布新的相关 CVE 时，都会对其进行新一轮扫描。每当推送或拉取映像时，都会刷新映像的 Amazon ECR 重新扫描持续时间。存储映像的存储库启用了增强扫描，存储库的增强扫描设置为持续扫描。
- 已激活(推送时) - 每次推送新映像时，Amazon Inspector 都会自动扫描该映像。存储映像的存储库启用了增强扫描，存储库的增强扫描设置为推送时扫描。
- 内部错误 – Amazon Inspector 尝试扫描容器映像时发生内部错误。Amazon Inspector 将自动处理错误并尽快恢复扫描。
- 等待初始扫描 – Amazon Inspector 已将映像纳入队列，等待初始扫描。
- 扫描资格已过期(持续) - Amazon Inspector 会暂停对映像的扫描。在您为自动重新扫描存储库中的映像指定的持续时间内，映像尚未更新。您可以推送或拉取映像以恢复扫描。
- 扫描资格已过期(推送时) - Amazon Inspector 会暂停对映像的扫描。在您为自动重新扫描存储库中的映像指定的持续时间内，映像尚未更新。您可以推送映像以恢复扫描。

- 手动扫描频率（手动）– Amazon Inspector 不会扫描 Amazon ECR 容器映像。存储映像的存储库的 Amazon ECR 扫描设置为基本手动扫描。要开始使用 Amazon Inspector 自动扫描映像，请将存储库设置为增强扫描，然后选择持续扫描映像或者仅在推送新映像时扫描。
- 不支持的操作系统 – Amazon Inspector 未监控或扫描映像。该映像基于 Amazon Inspector 不支持的操作系统，或者它使用了 Amazon Inspector 不支持的媒体类型。

  有关 Amazon Inspector 支持的操作系统的列表，请参阅支持的操作系统：使用 Amazon Inspector 执行 Amazon ECR 扫描。有关 Amazon Inspector 支持的媒体类型的列表，请参阅支持的媒体类型。

有关为存储库和映像配置扫描设置的详细信息，请参阅Amazon ECR 容器映像扫描。

评估 AWS Lambda 职能覆盖范围

Lambda 选项卡显示您的环境中的 Lambda 函数。 AWS 本页有两个表，一个显示 Lambda 标准扫描的函数覆盖率详细信息，另一个显示 Lambda 代码扫描的函数覆盖率详细信息。您可以根据以下选项卡对函数进行分组：

- 全部 – 显示环境中的所有 Lambda 函数。状态列显示 Lambda 函数的当前扫描状态。
- 扫描 – 显示根据 Amazon Inspector 配置要扫描的 Lambda 函数。状态列显示每个 Lambda 函数的当前扫描状态。
- 未扫描 – 显示根据 Amazon Inspector 配置未扫描的 Lambda 函数。原因列说明了为什么 Amazon Inspector 没有监控和扫描函数。

  Lambda 函数可能由于多种原因出现在未扫描选项卡上。Lambda 函数可能属于尚未添加到 Amazon Inspector 的账户，或者筛选规则阻止扫描此函数。有关更多信息，请参阅 Lambda 函数扫描。

在每个选项卡上，函数名称列指定 Lambda 函数的名称。“帐户” 列指定拥有 AWS 账户 该函数的。运行时系统指定函数的运行时系统。状态列显示每个 Lambda 函数的当前扫描状态。资源标签显示已应用于函数的标签。上次扫描列显示 Amazon Inspector 上次检查资源是否存在脆弱性的时间。这可能包括在更新调查发现元数据时、更新资源的应用程序清单时，或者针对新 CVE 重新扫描时进行检查。有关更多信息，请参阅 Lambda 函数扫描的扫描行为。

正在扫描 AWS Lambda 函数的状态值

对于 Lambda 函数，可能的状态值包括：

- 主动监控 – Amazon Inspector 正在持续监控和扫描 Lambda 函数。持续扫描包括在将新功能推送到存储库时对其进行初始扫描，以及在函数更新或发布新的常见漏洞和风险敞口 (CVEs) 时自动重新扫描这些函数。
- 按标签排除 – Amazon Inspector 未扫描此函数，因为按标签它已被排除在扫描范围之外。
- 扫描资格已过期 – Amazon Inspector 未监控此函数，因为自上次调用或更新该函数已过去 90 天或更长时间。
- 内部错误 – Amazon Inspector 尝试扫描函数时发生内部错误。Amazon Inspector 将自动处理错误并尽快恢复扫描。
- 等待初始扫描 – Amazon Inspector 已将函数纳入队列，等待初始扫描。
- 不支持 – Lambda 函数的运行时系统不受支持。

使用 Amazon Inspector 管理多个账户 AWS Organizations

可以使用 Amazon Inspector 管理组织中的多个账户。Amazon Inspector 支持两种多账户管理方法：

- AWS Organizations 策略的委托管理员-通过跨区域跨组织账户自动启用 Amazon Inspector，为委托管理员提供集中管理。组织策略强制启用哪些扫描类型，并且优先于非策略管理的委托管理员和成员帐户启用。
- 非 AWS Organizations 策略的委托管理员-指定用于在不使用组织政策的情况下管理组织的 Amazon Inspector 的账户。授权的管理员可以为成员账户启用 Amazon Inspector 并配置扫描设置。

这些方法可以一起使用。组织策略制定后，它们会控制资源类型的启用（启用了哪些扫描类型），而委派的管理员则保留对扫描配置设置（例如扫描模式和深度检查路径）的控制权。以下主题描述了这些管理方法、如何指定委派管理员以及如何管理成员账户。

主题

了解 Amazon Inspector 中的委派管理员账户和成员账户

在多账户环境中使用 Amazon Inspector 时，委派管理员账户可以访问特定元数据。元数据包括 Amazon EC2、Amazon ECR 和 Lambda 的标准扫描以及 Lambda 代码扫描。它还包括成员账户的安全调查发现结果。本节提供了有关委派管理员账户可以执行哪些操作以及成员账户可以执行哪些操作的信息。

组织政策治理模型

当使用 AWS Organizations 策略来启用 Amazon Inspector 时，将强制执行一种管理模型，该模型决定允许哪些操作：

策略管理的资源

> 由组织策略明确启用或禁用的资源不能由委派的管理员或成员账户进行修改。启用或禁用策略管理的扫描类型的 API 请求将失败，并显示一个明显的错误，表明该资源由组织策略管理。

Non-policy-managed resources

组织策略中未指定的资源可以由委托的管理员和成员账户使用 Amazon Inspector 控制台或 API 进行正常管理。

扫描配置管理

无论资源类型是否受策略管理，委派的管理员都可以随时配置扫描设置，例如 EC2 扫描模式、深度检查路径和 ECR 重新扫描持续时间。组织策略仅控制扫描是否启用，而不控制扫描的运行方式。

有关创建和管理 Amazon Inspector 组织政策的更多信息，请参阅 Amazon Inspector 政策 AWS Organizations 文档。

委派管理员操作

通常，当委派管理员将设置应用于其账户时，这些设置会应用于组织中的所有其他账户。委派管理员还可以查看和检索自有账户和任何关联成员的信息。Amazon Inspector 委派管理员账户可以执行以下操作：

- 只有 AWS Organizations 管理账号才能指定和移除委派的管理员。
- 指定委派管理员时，您必须与要管理的成员账户属于同一个组织。
- 查看和管理关联账户的 Amazon Inspector 状态，包括激活和停用 Amazon Inspector。
- 为组织内的所有成员账户激活或停用扫描类型。
- 查看整个组织的汇总调查发现数据，以及组织内所有成员账户的调查发现详情。
- 创建和管理应用于组织内所有账户的调查发现的抑制规则。
- 为组织的所有成员激活 Amazon ECR 增强扫描。
- 查看整个组织的资源覆盖率。
- 为组织内所有成员账户定义自动重新扫描 ECR 容器映像的持续时间。委托管理员的扫描持续时间设置会覆盖成员账户先前的所有设置。组织内的所有账户共享委派管理员的 Amazon ECR 自动重新扫描持续时间。不能为各个账户设置不同的重新扫描持续时间。
- 为 Amazon EC2 的 Amazon Inspector 深度检查指定五个自定义路径，这些路径将在组织内的所有账户中使用。除此之外，委托管理员还可为个人账户设置五个自定义路径。有关配置深度检查自定义路径的更多信息，请参阅Amazon Inspector 深度检查的自定义路径。
- 为成员账户激活和停用 Amazon Inspector 深度检查。
- SBOMs为组织中的任何成员账户@@ 导出。
- 为组织中的所有成员账户设置 Amazon EC2 扫描模式。有关更多信息，请参阅 管理扫描模式。

- 创建和管理组织中所有账户的 CIS 扫描配置，但成员账户创建的任何扫描配置除外。

> **Note**
>
> 如果成员账户离开组织，则委派管理员将无法再看到该账户计划的扫描配置。

- 查看组织中所有账户的 CIS 扫描结果。
- 使用组织策略时，请为策略管理的资源配置扫描设置，但不能自行启用或禁用策略管理的扫描类型。

成员账户操作

成员账户可以在 Amazon Inspector 中查看和检索有关其账户的信息，而其账户的设置则由委派管理员管理。组织内的成员账户可以在 Amazon Inspector 中执行以下操作：

- 为自己的账户激活 Amazon Inspector。
- 查看自己账户的资源覆盖率。
- 查看自己账户的调查发现详细信息。
- 查看自己账户的 ECR 容器映像自动重新扫描持续时间设置。
- 为 EC2 的 Amazon Inspector 深度检查指定五个自定义路径，这些路径将用于他们的个人账户。除了委派管理员为组织指定的自定义路径外，还会扫描这些路径。有关配置深度检查路径的更多信息，请参阅Amazon Inspector 深度检查的自定义路径。
- 查看您的委派管理员为 Amazon Inspector 深度检查设置的自定义路径。
- 导 SBOMs出与其账户关联的所有资源。
- 查看其账户的扫描模式。
- 为其账户创建和管理 CIS 扫描配置。
- 查看其账户中资源的任何 CIS 扫描结果，包括由委派管理员计划的扫描。
- 启用不受组织策略管理的扫描类型。成员帐户无法启用或禁用策略管理的扫描类型。

> **Note**
>
> 激活后，只有委托管理员账户才能停用 Amazon Inspector。

指定 Amazon Inspector 的委派管理员账户

委派管理员是一个为组织管理服务的账户。本主题介绍了如何指定 Amazon Inspector 的委派管理员。

注意事项

在指定委派管理员之前，请注意以下事项：

委派管理员最多可以管理 1 万个成员。

如果您的成员账户超过 10,000 个，您将通过 Amazon Person CloudWatch al Health Dashboard 收到通知，并通过电子邮件发送至委托管理员账户。

> **Note**
>
> 当通过针对拥有超过 10,000 个账户（最多 50,000 个）的组织的 AWS Organizations 策略启用 Amazon Inspector 时，该政策将适用于所有账户。但是，只有 10,000 个账户将与 Amazon Inspector 组织关联。也就是说，授权的管理员只能在 Amazon Inspector 控制台中查看这 10,000 个账户的调查结果和账户状态。

委派管理员是区域性的。

Amazon Inspector 是一项区域服务。在计划使用 Amazon Inspector 的每个 AWS 区域 地方，您都必须重复该过程中的步骤。

一个组织只能有一名委托管理员。

如果将一个账户指定为其中一个账户的委托管理员 AWS 区域，则该账户必须是所有其他账户中的委托管理员 AWS 区域。

更改委托管理员不会停用成员账户的 Amazon Inspector。

如果您移除委派管理员，则成员账户将变为独立账户，扫描设置不受影响。

您的 AWS 组织必须激活所有功能。

这是的默认设置 AWS Organizations。如果未激活所有特征，请参阅激活组织中的所有特征。

组织策略优先于委派的管理员设置。

如果您的组织使用 AWS Organizations 策略来启用 Amazon Inspector，则策略设置将决定启用哪些扫描类型。我们建议在创建组织策略之前指定委派的管理员，以确保一致的管理。有关更多信息，请参阅 组织政策治理模型。

指定委托管理员所需的权限

您必须拥有激活 Amazon Inspector 和指定 Amazon Inspector 委托管理员的权限。将以下语句添加到 IAM 策略的末尾以授予这些权限。有关更多信息，请参阅管理 IAM 策略。

```
{
    "Sid": "PermissionsForInspectorAdmin",
    "Effect": "Allow",
    "Action": [
        "inspector2:EnableDelegatedAdminAccount",
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization"
    ],
    "Resource": "*"
}
```

为您的 AWS 组织指定委派管理员

以下过程介绍了如何为您的组织指定委派管理员。在完成该过程之前，请确保您与想要委派管理员管理的成员账户均在同一个组织中。

> **Note**
>
> 您必须使用 AWS Organizations 管理账户才能完成此过程。只有 AWS Organizations 管理账号才能指定委派管理员。可能需要权限才能指定委派管理员。有关更多信息，请参阅 指定委托管理员所需的权限。

当您首次激活 Amazon Inspector 时，Amazon Inspector 会为账户创建服务相关角色 `AWSServiceRoleForAmazonInspector`。有关 Amazon Inspector 如何使用服务相关角色的信息，请参阅对 Amazon Inspector 使用服务相关角色。

Console

指定 Amazon Inspector 委托管理员

1. 登录 AWS Organizations 管理账户，然后在 https://console.aws.amazon.com/inspector/v2/home 上打开 Amazon Inspector 控制台。
2. 使用 AWS 区域 选择器指定要在 AWS 区域 哪里指定委派管理员。
3. 在导航窗格中，选择常规设置。
4. 在 “委托管理员” 下，输入 AWS 账户 要指定为委派管理员的 12 位 ID。
5. 选择委派，然后再次选择委派。

当您指定委派管理员时，默认情况下会为该账户激活所有扫描类型。如果您想为 AWS Organizations 管理账户激活 Amazon Inspector，请完成以下步骤。

为 AWS Organizations 管理账户激活 Amazon Inspector

1. 登录委托管理员账户，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 在导航窗格中，选择账户管理。
3. 在 “帐户” 下，选择 AWS Organizations 管理帐户，然后选择 “激活”。
4. 选择要为 AWS Organizations 管理账户激活的扫描类型，然后选择提交。

API

使用 API 指定委派管理员

- 使用 Organizations 管理账户 AWS 账户 的凭据运行 EnableDelegatedAdminAccountAPI 操作。您也可以通过运行 AWS Command Line Interface 以下 CLI 命令来执行此操作：`aws inspector2 enable-delegated-admin-account --delegated-admin-account-id 11111111111`。

> **Note**
>
> 确保指定要设置为 Amazon Inspector 委派管理员的账户的账户 ID。

为成员账户激活 Amazon Inspector 扫描

您可以通过多种方法为组织中的成员账户激活 Amazon Inspector。您选择的方法取决于您的治理要求和组织结构。

AWS Organizations 策略（建议用于集中式治理）

使用 AWS Organizations 策略在整个组织中自动启用具有集中控制功能的 Amazon Inspector。这种方法可确保一致的扫描覆盖范围，并自动应用于新帐户。有关详细说明，请参阅有关创建 Amazon Inspector 策略的 AWS Organizations 文档。

委托管理员激活

作为委托管理员，您可以通过亚马逊检查器控制台或 API 为特定成员账户或所有成员账户手动激活 Amazon Inspector。当不使用组织策略时，这种方法提供了灵活性。

会员账户自动激活

在不受组织政策限制的情况下，成员账户可以为自己的账户激活 Amazon Inspector。激活后，该账户将与委派的管理员关联。

为成员账户激活扫描

以下过程介绍如何使用委派管理员和成员帐户方法激活对成员帐户的扫描。有关 Amazon Inspector 扫描类型的信息，请参阅Amazon Inspector 中的自动扫描类型。

要为所有成员账户自动激活扫描

1. 使用委派的管理员账户证书登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 上打开 Amazon Inspector 控制台。
2. 使用区域选择器选择要激活所有成员账户扫描功能 AWS 区域 的位置。
3. 在导航窗格中，选择账户管理。“帐户” 选项卡显示与 AWS Organizations 管理账户关联的所有成员账户。
4. 在组织下，选中账号旁边的复选框。然后选择激活，选择要应用于成员账户的扫描选项。您可以选择以下扫描类型：

   - Amazon EC2 扫描
   - Amazon ECR 扫描
   - Lambda 标准扫描
   - Lambda 代码扫描

- 选择首选扫描类型后，选择保存。

> **Note**
>
> 如果您有多页账户，则必须在每个页面上重复此步骤。要更改每个页面上显示的账户数量，请选择齿轮图标。

5. 打开为新成员账户自动激活 Inspector 设置，然后选择要为添加到组织的新成员账户应用的扫描选项。您可以选择以下扫描类型：

   - Amazon EC2 扫描
   - Amazon ECR 扫描
   - Lambda 标准扫描
   - Lambda 代码扫描

   - 选择首选扫描类型后，选择激活。

> **Note**
>
> 为新成员账户自动激活 Inspector 设置可为组织的所有未来成员激活 Amazon Inspector。如果成员账户数量超过 5000 时，此设置将自动关闭。如果成员账户总数减少到少于 5000，则该设置将自动重新激活。

6. （推荐）在要激活对成员帐户 AWS 区域 的扫描功能的每个步骤中重复上述每个步骤。

为特定成员账户激活扫描

1. 使用委派的管理员账户证书登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 上打开 Amazon Inspector 控制台。
2. 使用区域选择器选择要激活所有成员账户扫描功能 AWS 区域 的位置。
3. 在导航窗格中，选择账户管理。“帐户” 选项卡显示与 AWS Organizations 管理账户关联的所有成员账户。
4. 在组织下，选中要为其激活扫描的每个成员账号旁边的复选框。然后选择激活，选择要应用于成员账户的扫描选项。您可以选择以下扫描类型：

- Amazon EC2 扫描
- Amazon ECR 扫描
- Lambda 标准扫描
- Lambda 代码扫描

- 选择首选扫描类型后，选择保存。

> **Note**
>
> 如果您有多页账户，则必须在每个页面上重复此步骤。要更改每个页面上显示的账户数量，请选择齿轮图标。

5. （推荐）在要激活对特定成员 AWS 区域 的扫描功能的每个步骤中重复上述每个步骤。

以成员账户身份激活扫描

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用区域选择器选择要激活所有成员账户扫描功能 AWS 区域 的位置。
3. 在导航窗格中，选择账户管理。“帐户” 选项卡显示与 AWS Organizations 管理账户关联的所有成员账户。
4. 在组织下，选中您的账号旁边的复选框。然后选择激活，选择要应用的扫描选项。您可以选择以下扫描类型：

   - Amazon EC2 扫描
   - Amazon ECR 扫描
   - Lambda 标准扫描
   - Lambda 代码扫描

   - 选择首选扫描类型后，选择保存。
5. （推荐）在要为成员账户激活扫描的每个区域重复这些步骤。

> **Note**
>
> 如果您的 AWS Organizations 管理账户有 Amazon Inspector 的委托管理员账户，则可以将您的账户激活为成员账户以查看扫描详情。

> **重要提示**
>
> 如果组织策略正在管理您的账户的 Amazon Inspector 启用，则委托的管理员和成员账户无法使用 Amazon Inspector 修改策略管理的扫描类型。 enablement/disablement APIsAPI 请求将失败，并显示一条错误消息，表明该资源由组织策略管理。您仍然可以启用策略未管理的其他扫描类型。

在 Amazon Inspector 中取消成员账户的关联

作为委派管理员，您可能需要取消成员账户与账户的关联。在取消关联成员账户时，账户中的 Amazon Inspector 仍处于激活状态，该账户将变为独立账户。您也无权再管理该账户的 Amazon Inspector。但是，您可以随时将之前取消关联的成员账户与您的账户再次关联起来。本节介绍了如何以委派管理员身份取消关联成员账户。

> **Note**
>
> 要解除与策略管理的账户的关联，则不应为该账户附加任何针对该扫描类型的Amazon Inspector组织政策。

Console

使用控制台取消成员账户的关联

1. 使用委派的管理员账户证书登录，然后在 https://console.aws.amazon.com/inspector/v 2/home 上打开 Amazon Inspector 控制台
2. 使用区域选择器选择要取消关联成员账户 AWS 区域 的位置。
3. 在导航窗格中，选择账户管理。
4. 在组织下，选中您要取消关联的每个账号旁边的复选框。
5. 选择操作菜单，然后选择取消关联账户。

API

使用 API 取消成员账户的关联

运行 DisassociateMember API 操作。在请求中，提供 IDs 您要取消关联的账户。

在 Amazon Inspector 中移除委派管理员

您可能需要移除 Amazon Inspector 委派管理员账户。您可以通过 AWS Organizations 管理账户执行此操作。在移除 Amazon Inspector 委派管理员账户时，该账户及其所有成员账户中的 Amazon Inspector 仍处于激活状态。委派管理员账户及其所有成员账户将成为独立账户，并保留其原始扫描设置。

> **Note**
>
> 如果 AWS Organizations 策略正在管理 Amazon Inspector 的启用，则移除委派的管理员不会影响策略的执行。根据组织策略设置，账户将保持启用状态，但在指定新的授权管理员之前，成员账户的发现结果将不再显示在中央委派管理员控制台中。

本节介绍了如何移除委派管理员账户。

移除 Amazon Inspector 委派管理员

以下过程介绍了如何移除 Amazon Inspector 委派管理员以及如何将成员账户与委派管理员账户关联起来。

有关如何指定 Amazon Inspector 委派管理员的信息，请参阅为 Amazon Inspector 指定委派管理员账户。

> **Note**
>
> 在指定 Amazon Inspector 委派管理员后，Amazon Inspector 委派管理员必须手动关联成员账户。

移除委托管理员

1. AWS 管理控制台 使用 AWS Organizations 管理账户登录。
2. 在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。

3. 使用区域选择器选择要移除委派管理员 AWS 区域 的位置。
4. 在导航窗格中，选择常规设置。
5. 在委派管理员下，选择移除，然后确认操作。

将成员与新的委托管理员关联

1. 使用委派的管理员账户证书登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 上打开 Amazon Inspector 控制台。
2. 使用区域选择器选择您想要关联成员 AWS 区域 的位置。
3. 在导航窗格中，选择账户管理。
4. 在组织下，选中账号旁边的复选框。
5. 选择操作，然后选择添加成员。

标记 Amazon Inspector 资源

标签是添加到 AWS 资源的标签。标签可帮助您根据特定标准对 AWS 资源进行分类。标签由键值对组成。标签键是一个通用标签。标签值是标签键的描述。使用 Amazon Inspector，您可以标记抑制规则和 CIS 扫描配置。您最多可以向每个 Amazon Inspector 资源添加 50 个标签。

标记基础知识

一个标签由一个键值对组成。标签键是一个通用标签。标签值是标签键的描述。本主题介绍了标记 Amazon Inspector 资源的基础知识。在标记 Amazon Inspector 资源时，请注意以下几点：

- 您可以标记抑制规则和 CIS 扫描配置。
- 您最多可以向每个 Amazon Inspector 资源添加 50 个标签。
- 标签键必须是唯一的。
- 一个标签键只能有一个标签值。
- 标签键和标签值最多可以包含 128 个 UTF-8 字符。这些字符可以是字母、数字、空格或以下符号：`_ . : / = + - @`。
- 不能在任何标签中使用 `aws` 前缀，也不能修改带有此前缀的标签。带有 `aws` 前缀的标签保留供 AWS 使用。
- 分配给 Amazon Inspector 资源的标签仅在您的 AWS 账户以及您在其中创建它们的 AWS 区域中可用。
- 当您删除一个资源时，与其关联的所有标签也会被删除。

有关标签的更多信息，请参阅《标记 AWS 资源和标签编辑器用户指南》中的最佳实践和策略。

> **Note**
>
> 标签不能用于存储机密或敏感信息。切勿使用标签来存储此类数据。可从其他 AWS 服务访问标签。

添加标签

您可以为 Amazon Inspector 资源添加标签。这些资源包括抑制规则和 CIS 扫描配置。标签可帮助您根据特定标准对 AWS 资源进行分类。本主题介绍如何向 Amazon Inspector 资源添加标签。

为 Amazon Inspector 资源添加标签

您可以标记抑制规则和 CIS 扫描配置。以下步骤介绍了如何在控制台中以及使用 Amazon Inspector API 添加标签。

在控制台中添加标签

您可以在控制台中为 Amazon Inspector 资源添加标签。

向抑制规则添加标签

您可以在创建期间向抑制规则添加标签。有关更多信息，请参阅创建抑制规则。

您也可以编辑抑制规则以包含标签。有关更多信息，请参阅编辑抑制规则。

将标签添加到 CIS 扫描配置

在创建过程中，可以向 CIS 扫描配置添加标签。有关更多信息，请参阅创建 CIS 扫描配置。

您还可以编辑 CIS 扫描配置，以包含标签。有关更多信息，请参阅编辑 CIS 扫描配置。

使用 Amazon Inspector API 添加标签

您可以使用 Amazon Inspector API 向 Amazon Inspector 资源添加标签。

为 Amazon Inspector 资源添加标签

使用 `TagResource` API 向 Amazon Inspector 资源添加标签。您必须在命令中包含资源的 ARN 以及标签的键值对。以下示例命令使用空资源 ARN 作为抑制筛选条件。键是 `CostAllocation`，值是 `dev`。有关 Amazon Inspector 的资源类型的信息，请参阅《服务授权参考》中的 Amazon Inspector2 的操作、资源和条件键。

```
aws inspector2 tag-resource \
--resource-arn "arn:${Partition}:inspector2:${Region}:${Account}:owner/${OwnerId}/
filter/${FilterId}" \
--tags CostAllocation=dev \
--region us-west-2
```

在创建期间向抑制规则添加标签

在创建期间，使用 `CreateFilter` API 向抑制规则添加标签。

```
aws inspector2 create-filter \
```

```
--name "ExampleSuppressionRuleECR" \
--action SUPPRESS \
--filter-criteria 'resourceType=[{comparison="EQUALS", value="AWS_ECR_IMAGE"}]' \
--tags Owner=ApplicationSecurity \
--region us-west-2
```

将标签添加到 CIS 扫描配置

使用 CreateCisScanConfiguration API 向 CIS 扫描配置添加标签。

```
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
```

删除标签

可以从 Amazon Inspector 资源中删除标签。这些资源包括抑制规则和 CIS 扫描配置。标签可帮助您根据特定标准对 AWS 资源进行分类。本主题介绍了如何从 Amazon Inspector 资源中删除标签。

从 Amazon Inspector 资源中删除标签

您可以从抑制规则和 CIS 扫描配置中删除标签。以下步骤介绍了如何在控制台中以及使用 Amazon Inspector API 删除标签。

在控制台中删除标签

您可以在控制台中从 Amazon Inspector 资源中删除标签。

从抑制规则中删除标签

通过将抑制规则编辑为不再包含某个标签，可以从抑制规则中删除该标签。有关更多信息，请参阅编辑抑制规则。

从 CIS 扫描配置中删除标签

通过将 CIS 扫描配置编辑为不再包含某个标记，可以从 CIS 扫描配置中删除该标签。有关更多信息，请参阅编辑 CIS 扫描配置。

使用 Amazon Inspector API 删除标签

您可以使用 Amazon Inspector API 从 Amazon Inspector 资源中删除标签。

从 Amazon Inspector 资源中删除标签

使用 `UntagResource` API 从 Amazon Inspector 资源中删除标签。

以下片段展示了有关如何使用 `UntagResource` 从 Amazon Inspector 资源中删除标签的示例。您必须在命令中包含资源的 ARN 和标签的键。以下示例使用空资源 ARN 作为抑制筛选条件。键是 `CostAllocation`。有关 Amazon Inspector 的资源类型的信息，请参阅《服务授权参考》中的 Amazon Inspector2 的操作、资源和条件键。

```
aws inspector2 untag-resource \
--resource-arn "arn:${Partition}:inspector2:${Region}:${Account}:owner/${OwnerId}/cis-configuration/${CISScanConfigurationId}" \
--tag-keys CostAllocation \
--region us-west-2
```

监控 Amazon Inspector 的使用量和成本

可以使用 Amazon Inspector 控制台和 API 来预测环境中 Amazon Inspector 的每月成本。如果您是多账户环境的 Amazon Inspector 管理员，则可以查看环境的总成本以及所有成员账户的成本指标。本节介绍了如何访问使用情况统计数据以及如何计算使用成本。

使用“使用量”控制台

您可以通过控制台评测 Amazon Inspector 的使用量和预计成本。

访问“使用量”统计数据

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用页面右上角的 AWS 区域 选择器，选择要监控成本的区域。
3. 在导航窗格中，选择使用量。

在按账户选项卡中，您将看到账户使用量下列出的 30 天期间的预计总费用。在预计成本列下的表格中，选择一个值，以查看该账户按扫描类型划分的使用量明细。在此详细信息窗格中，您还可以看到该账户的哪些扫描类型激活了免费试用。

如果您是某组织的委托管理员，您将在表格中看到组织内每个账户对应的行。如果您组织中的某个账户已取消关联，则控制台会将其预计费用显示为 -。

在按扫描类型选项卡中，您可以看到当前 30 天内按扫描类型划分的使用量明细。这些信息用于计算按账户选项卡中的预计成本。

如果您是某组织的委托管理员，您可以看到组织内每个账户的使用量。

在此选项卡中，您可以展开以下任一窗格以获取使用量统计数据：

亚马逊 EC2 扫描

Amazon Inspector 使用情况控制台会针对基于代理的扫描和无代理扫描跟踪以下指标：

- 实例（平均）— Amazon Inspector 使用服务时间来计算 EC2 实例扫描的平均资源数量。平均值等于总覆盖时间除以 720 小时（30 天内的小时数）。
- 覆盖时长 — 对于亚马逊 EC2 扫描，这是过去 30 天内 Amazon Inspector 为账户中的每个 EC2 实例提供有效保险的总时数。例如 EC2 ，服务时间是指从 Amazon Inspector 发现实例到实例被

终止或停止，或者按标签将其排除在扫描之外的时间。（当您重启已停止的实例或删除排除标签时，Amazon Inspector 将恢复覆盖范围，并且该实例的覆盖时间将继续累积）。

CIS 实例扫描 - 对账户中的实例执行的 CIS 扫描总数。

Amazon ECR 扫描

初始扫描 — 过去 30 天内首次扫描账户中映像的总次数。

重新扫描 — 过去 30 天内重新扫描账户中映像的总次数。重新扫描是指对 Amazon Inspector 之前扫描过的 ECR 映像进行的所有扫描。如果您已将 ECR 存储库配置为持续扫描，则当 Amazon Inspector 向其数据库添加新的常见脆弱性和风险 (CVE) 时，会自动进行重新扫描。

Lambda 扫描

Amazon Inspector 使用情况控制台会针对 Lambda 标准扫描和 Lambda 代码扫描跟踪以下指标：

- Lambda 函数的数量(平均值) - Amazon Inspector 使用覆盖时间来计算 Lambda 函数扫描的平均函数数量。平均值等于总覆盖时间除以 720 小时（30 天内的小时数）。
- 覆盖时间 — 对于 Lambda 函数扫描，这是过去 30 天内 Amazon Inspector 为账户中的每个 Lambda 函数提供有效覆盖的总小时数。对于 AWS Lambda 函数，覆盖时间是指从 Amazon Inspector 发现函数到函数被删除或从扫描中排除的时间。如果被排除的函数再次被纳入，则该函数的覆盖时间将继续累积。

了解 Amazon Inspector 如何计算使用成本

Amazon Inspector 提供的费用是估算值，而不是实际成本，因此它们可能与您的 AWS Billing 主机中的费用不同。

请注意以下有关 Amazon Inspector 如何在使用量页面上计算成本的信息：

- 使用成本仅反映当前区域的情况。每种扫描类型的价格因 AWS 地区而异，要查看每个地区的确切价格，请参阅 Amazon Inspector 的定价
- 所有使用量预测均按美元四舍五入。
- 预计费用不包含折扣。
- 预计成本代表每种扫描类型 30 天使用期内的总成本。如果账户的使用天数少于 30 天，Amazon Inspector 会预测 30 天后的费用，并假设当前覆盖的所有资源仍将在 30 天的剩余时间内继续保持覆盖。
- 每种扫描类型的成本根据以下方式计算：
  - EC2 扫描：费用反映了过去 30 天内 Amazon Inspector 覆盖的平均 EC2 实例数。

- ECR 容器扫描：成本反映过去 30 天内初始映像扫描次数 + 映像重新扫描次数的总和。
- Lambda 标准扫描：成本反映过去 30 天内 Amazon Inspector 覆盖的 Lambda 函数的平均数量。
- Lambda 代码扫描：成本反映过去 30 天内 Amazon Inspector 覆盖的 Lambda 函数的平均数量。

关于 Amazon Inspector 免费试用

在 Amazon Inspector 中，每种扫描类型都可以免费试用。激活扫描类型后，您将自动注册该扫描类型的 15 天免费试用。免费试用开始后，即使您停用扫描类型，它也会在 15 天后自动过期。

> **Note**
>
> 免费试用不适用于 CIS 扫描。

Amazon Inspector 安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Inspector 的合规计划，请参阅合规计划范围内的 AWSAWS 服务按合分的范围内服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Inspector 时应用责任共担模式。以下主题说明如何配置 Amazon Inspector 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon Inspector 资源。

主题

Amazon Inspector 中的数据保护

AWS 分担责任模型分担责任模型适用于 Amazon Inspector 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私

常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅AWS CloudTrail 用户指南中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您 AWS 服务 使用控制台、API 或与 Amazon Inspector 或其他人合作时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

主题

静态加密

默认情况下，Amazon Inspector 使用 AWS 加密解决方案存储静态数据。Amazon Inspector 对以下内容等数据进行加密：

- 使用收集的资源清单 AWS Systems Manager。
- 从 Amazon Elastic Container Registry 映像解析的资源清单
- 使用 AWS 自有的加密密钥生成安全调查结果 AWS Key Management Service

您无法管理、使用或查看 AWS 拥有的密钥。但是无需执行任何操作或更改任何计划即可保护用于加密数据的密钥。有关更多信息，请参阅 AWS 拥有的密钥。

如果您禁用 Amazon Inspector，它将永久删除自己为您存储或维护的所有资源，例如收集的清单和安全调查发现。

对调查发现中的代码进行静态加密

为了扫描 Amazon Inspector Lambda 代码，Amazon Inspector 可与 Amazon Q 搭配使用，来扫描您的代码中是否存在漏洞。当检测到漏洞时，Amazon Q 会提取包含相应漏洞的代码片段，并将该代码存储起来，直到 Amazon Inspector 请求访问为止。默认情况下，Amazon Q 使用 AWS 自有密钥对提取的代码进行加密。但是，您可以将 Amazon Inspector 配置为使用您自己的客户管理的 AWS KMS 密钥进行加密。

以下工作流说明了 Amazon Inspector 如何使用您配置的密钥来加密代码：

1. 您可以使用 Amazon Inspector UpdateEncryptionKeyAPI 向亚马逊 Inspector 提供 AWS KMS 密钥。
2. Amazon Inspector 会将有关您的 AWS KMS 密钥的信息转发给 Amazon Q，Amazon Q 会存储这些信息以备将来使用。
3. Amazon Q 使用您通过密钥策略在 Amazon Inspector 中配置的 KMS 密钥。
4. Amazon Q 会根据您的密钥创建加密数据 AWS KMS 密钥并将其存储。此数据密钥用于加密由 Amazon Q 存储的代码数据。
5. 当 Amazon Inspector 请求从代码扫描中获取数据时，Amazon Q 会使用 KMS 密钥来解密数据密钥。当禁用 Lambda 代码扫描时，Amazon Q 会删除关联的数据密钥。

使用客户托管密钥进行代码加密的权限

为了进行加密，您必须使用一个策略创建 KMS 密钥，Amazon Inspector 和 Amazon Q 可利用该策略中包含的语句执行以下操作。

- `kms:Decrypt`
- `kms:DescribeKey`
- `kms:Encrypt`
- `kms:GenerateDataKey`
- `kms:GenerateDataKeyWithoutPlainText`

策略声明

创建 KMS 密钥时，您可以使用以下策略语句。

> **Note**
>
> *account-id*替换为您的 12 位数 AWS 账户 身份证。*Region*替换为您启用 Amazon Inspector 和 Lambda 代码扫描 AWS 区域 的位置。将 *role-ARN* 替换为您的 IAM 角色的 Amazon 资源名称。

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "q.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKeyWithoutPlaintext",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:EncryptionContext:aws:qdeveloper:lambda-codescan-scope": "account-id"
    },
    "StringEquals": {
      "aws:SourceAccount": "account-id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:qdeveloper:Region:account-id:scans/*"
    }
  }
},
{
  "Effect": "Allow",
  "Principal": {
    "Service": "q.amazonaws.com"
  },
  "Action": "kms:DescribeKey",
```

```
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "account-id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:qdeveloper:Region:account-id:scans/*"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKeyWithoutPlaintext",
    "kms:GenerateDataKey"
  ],
  "Principal": {
    "AWS": "role-ARN"
  },
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "inspector2.Region.amazonaws.com"
    },
    "StringLike": {
      "kms:EncryptionContext:aws:qdeveloper:lambda-codescan-scope": "account-id"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "kms:DescribeKey"
  ],
  "Principal": {
    "AWS": "role-ARN"
  },
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "inspector2.Region.amazonaws.com"
    }
```

```
  }
}
```

该策略语句采用 JSON 格式。包含该语句后，请检查策略以确保语法有效。如果该语句是策略中的最后一条语句，请在前一语句的右大括号后加上一个逗号。如果该语句是策略中的第一个语句或介于两个现有语句之间，请在该语句的右括号后加上一个逗号。

> **Note**
>
> Amazon Inspector 不再支持对从程序包中提取的代码片段进行加密的授权。如果您使用的是基于授权的策略，则您仍然可以访问扫描结果。但是，如果您在任何时候更新或重置 KMS 密钥或禁用 Lambda 代码扫描，则需要使用本节中介绍的 KMS 密钥策略。

如果您设置、更新或重置账户的加密密钥，则必须使用 Amazon Inspector 管理员策略，例如 AWS 托管策略`AmazonInspector2FullAccess`。

使用客户托管密钥配置加密

要使用客户托管密钥为您的账户配置加密，您必须是具有 使用客户托管密钥进行代码加密的权限 中列出的权限的 Amazon Inspector 管理员。此外，您还需要一个与您的发现位于同一 AWS 区域的 AWS KMS 密钥，或者一个多区域密钥。您可以使用账户中现有的对称密钥，也可以使用 AWS 管理控制台创建对称客户托管密钥，或者。 AWS KMS APIs有关更多信息，请参阅 AWS KMS 用户指南中的创建对称加密 AWS KMS 密钥。

> **Note**
>
> 自 2025 年 6 月 13 日起，在代码片段 CloudTrail 期间登录的 AWS KMS 请求中的服务主体将从 “codeguru-reviewer” 更改 encryption/decryption 为 “q”。

使用 Amazon Inspector API 配置加密

要设置加密密钥，请在以亚马逊 Inspector 管理员身份登录时UpdateEncryptionKey运行 Amazon Inspector API。在 API 请求中，使用`kmsKeyId`字段指定要使用的 AWS KMS 密钥的 ARN。对于`scanType`，请输入`CODE`，对于`resourceType`，请输入`AWS_LAMBDA_FUNCTION`。

你可以使用 UpdateEncryptionKeyAPI 来查看 Amazon Inspector 正在使用哪个 AWS KMS 密钥进行加密。

> **Note**
>
> 如果您在未设置客户托管密钥`GetEncryptionKey`的情况下尝试使用，则操作会返回`ResourceNotFoundException`错误，这意味着正在使用 AWS 自有密钥进行加密。

如果删除密钥或更改其策略以拒绝访问 Amazon Inspector 或 Amazon Q，您将无法访问代码漏洞扫描结果，并且您的账户的 Lambda 代码扫描将失败。

您可以使用恢复使用 AWS 自有密钥`ResetEncryptionKey`对作为 Amazon Inspector 调查结果一部分提取的代码进行加密。

传输中加密

AWS 对 AWS 内部系统和其他 AWS 服务之间传输的所有数据进行加密。 AWS Systems Manager 通过受传输层安全 (TLS) 保护的通道从客户拥有的 EC2 实例中收集遥测数据以 AWS 进行评估。发送到 Security Hub CSPM 的 Amazon ECR 和 Lamb AWS da 函数扫描结果使用受 TLS 保护的通道进行加密。有关更多信息，请参阅 Systems Manager 中的数据保护来了解 SSM 如何加密传输中数据。

适用于 Amazon Inspector 的 Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Amazon Inspector 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon Inspector 如何与 IAM 配合使用
- Amazon Inspector 基于身份的策略示例
- AWS Amazon Inspector 的托管政策
- 对 Amazon Inspector 使用服务相关角色

- Amazon Inspector 身份和访问问题排查

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅Amazon Inspector 身份和访问问题排查）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅Amazon Inspector 如何与 IAM 配合使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅Amazon Inspector 基于身份的策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Inspector 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon Inspector 的访问权限之前，您应该了解哪些 IAM 特征可用于 Amazon Inspector。

可与 Amazon Inspector 结合使用的 IAM 功能

IAM 特征	Amazon Inspector 支持
基于身份的策略	是
基于资源的策略	否

IAM 特征	Amazon Inspector 支持
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	是
服务角色	否
服务关联角色	是

要全面了解 Amazon Inspector 和其他 AWS 服务 功能如何使用大多数 IAM 功能 AWS 服务 ，请在 IAM 用户指南中查看与 IAM 配合使用的方法。

Amazon Inspector 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Amazon Inspector 基于身份的策略示例

要查看 Amazon Inspector 基于身份的策略的示例，请参阅Amazon Inspector 基于身份的策略示例。

Amazon Inspector 基于资源的策略

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amazon Inspector 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

有关 Amazon Inspector 操作的列表，请参阅《服务授权参考》中的 Amazon Inspector 定义的操作。

Amazon Inspector 中的策略操作在操作前使用以下前缀：

```
inspector2
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "inspector2:action1",
      "inspector2:action2"
         ]
```

要查看 Amazon Inspector 基于身份的策略的示例，请参阅Amazon Inspector 基于身份的策略示例。

Amazon Inspector 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 Amazon Inspector 资源类型及其列表 ARNs，请参阅《服务授权参考》中的 Amazon Inspector 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon Inspector 定义的操作。

要查看 Amazon Inspector 基于身份的策略的示例，请参阅Amazon Inspector 基于身份的策略示例。

Amazon Inspector 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Condition` 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 Amazon Inspector 条件键的列表，请参阅《服务授权参考》中的 Amazon Inspector 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Inspector 定义的操作。

要查看 Amazon Inspector 基于身份的策略的示例，请参阅Amazon Inspector 基于身份的策略示例。

ACLs 在亚马逊 Inspector 中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

ABAC 与 Amazon Inspector

支持 ABAC（策略中的标签）：部分支持

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`aws:RequestTag/*key-name* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 Amazon Inspector

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

Amazon Inspector 的跨服务主体权限

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon Inspector 的服务角色

支持服务角色：否

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon Inspector 的功能。仅当 Amazon Inspector 提供相关指导时才编辑服务角色。

Amazon Inspector 的服务相关角色

支持服务关联角色：是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅使用 IAM 的AWS 服务 服务。在表中查找服务相关角色列中包含 Yes 额表。选择是链接以查看该服务的服务相关角色文档。

Amazon Inspector 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon Inspector 资源的权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 Amazon Inspector 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon Inspector 的操作、资源和条件键。 ARNs

主题

- 策略最佳实践
- 使用 Amazon Inspector 控制台
- 允许用户查看他们自己的权限
- 允许以只读方式访问所有 Amazon Inspector 资源
- 允许完全访问所有 Amazon Inspector 资源

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon Inspector 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。

- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon Inspector 控制台

要访问 Amazon Inspector 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 AWS 账户中的 Amazon Inspector 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 Amazon Inspector 控制台，还需要将亚马逊检查器*ConsoleAccess*或*ReadOnly* AWS 托管策略附加到这些实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
}
```

允许以只读方式访问所有 Amazon Inspector 资源

以下示例展示了一个策略，该策略允许以只读方式访问所有 Amazon Inspector 资源。

JSON

```
{
    "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": "*"
        }
    ]
}
```

允许完全访问所有 Amazon Inspector 资源

以下示例展示了一个策略，该策略允许完全访问所有 Amazon Inspector 资源。

JSON

```
{
    "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 的托管政策

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略：AmazonInspector2FullAccess_v2

您可以将 `AmazonInspector2FullAccess_v2` 策略附加到 IAM 身份。

此策略会授予对 Amazon Inspector 的完全访问权限以及对其他相关服务的访问权限。

权限详细信息

该策略包含以下权限。

- `inspector2`— 允许完全访问亚马逊 Inspector APIs。
- `codeguru-security` – 便于管理员检索账户的安全扫描结果和配置设置。
- `iam` – 便于 Amazon Inspector 创建服务相关角色 `AWSServiceRoleForAmazonInspector2` 和 `AWSServiceRoleForAmazonInspector2Agentless`。Amazon Inspector 需要 `AWSServiceRoleForAmazonInspector2` 来执行诸如检索有关 Amazon EC2 实例、Amazon ECR 存储库和 Amazon ECR 容器映像的信息等操作。还需要解密使用密钥加密的 Amazon EBS 快照。 AWS KMS 有关更多信息，请参阅 对 Amazon Inspector 使用服务相关角色。
- `organizations`— 仅`AllowServicePrincipalBasedAccessToOrganizationApis`允许服务委托人为组织创建服务相关角色 AWS 账户、注册为 AWS 账户 组织的委托管理员以及列出组织中的授权管理员。 `AllowOrganizationalBasedAccessToOrganizationApis`允许保单持有人检索有关组织单位的信息 ARNs，特别是资源级别的信息。`AllowAccountsBasedAccessToOrganizationApis`允许保单持有人检索有关某人的信息，特别是资源级 ARNs信息。 AWS 账户`AllowAccessToOrganizationApis`允许保单持有人查看与组织和组织 AWS 服务 集成的信息。该政策允许列出 Inspector 组织政策，并按照 Inspector 策略类型进行筛选，查看由管理账户建立的委托资源策略，以及查看应用于账户的有效 Inspector 策略。

> **Note**
>
> Amazon Inspector 不再使用它 CodeGuru 来执行 Lambda 扫描。 AWS 将于 2025 年 11 月 20 CodeGuru 日停止提供支持。有关更多信息，请参阅终止对 CodeGuru 安全的支持。Amazon Inspector 现在使用 Amazon Q 执行 Lambda 扫描，不需要本节中描述的权限。

要查看此策略的权限，请参阅《AWS 托管策略参考指南》中的 AmazonInspectorFullAccess2 _v2。

AWS 托管策略：AWSInspector2OrganizationsAccess

您可以将 `AWSInspector2OrganizationsAccess` 策略附加到 IAM 身份。

该策略授予管理权限，以启用和管理 Amazon Inspector 中的组织 AWS Organizations。该策略的权限允许组织管理账户为 Amazon Inspector 指定委托管理员账户。它们还允许委派管理员账户将组织账户启用为成员账户。

此策略仅为提供权限 AWS Organizations。组织管理账户和委派管理员账户也需要关联操作的权限。这些权限可以使用 AmazonInspector2FullAccess_v2 管理策略来授予。

权限详细信息

该策略包含以下权限。

- organizations:ListAccounts：允许主体检索属于某个组织的账户列表。
- organizations:DescribeOrganization：允许主体检索有关组织的信息。
- organizations:ListRoots：允许主体列出组织的根。
- organizations:ListDelegatedAdministrators：允许主体列出组织的委派管理员。
- organizations:ListAWSServiceAccessForOrganization— 允许委托人列出组织使用 AWS 服务 的。
- organizations:ListOrganizationalUnitsForParent：允许主体列出父组织单元（OU）的子 OU。
- organizations:ListAccountsForParent：允许主体列出父 OU 的子账户。
- organizations:ListParents— 列出作为指定子组织单位或账户的直系父级的根单位或组织单位 (OUs)。
- organizations:DescribeAccount – 让委托人可以检索有关企业中某个账户的信息。
- organizations:DescribeOrganizationalUnit：允许主体检索有关组织中某个 OU 的信息。
- organizations:ListPolicies – 检索指定类型的组织中所有策略的列表。
- organizations:ListPoliciesForTarget – 列出直接附加到指定的目标根、组织单元（OU）或账户的策略。
- organizations:ListTargetsForPolicy— 列出指定策略所关联的所有根目录、组织单位 (OUs) 和帐户。
- organizations:DescribeResourcePolicy— 检索有关资源策略的信息。
- organizations:EnableAWSServiceAccess – 允许主体启用与 Organions 的集成。
- organizations:RegisterDelegatedAdministrator – 允许主体指定委派管理员账户。
- organizations:DeregisterDelegatedAdministrator – 允许主体移除委派管理员账户。
- organizations:DescribePolicy – 检索有关策略的信息。
- organizations:DescribeEffectivePolicy – 返回指定的策略类型和账户的有效策略的内容。

- `organizations:CreatePolicy`— 创建指定类型的策略，您可以将其附加到根目录、组织单位 (OU) 或个人 AWS 账户。
- `organizations:UpdatePolicy` – 使用新的名称、描述或内容更新现有策略。
- `organizations:DeletePolicy` – 从您的组织中删除指定的策略。
- `organizations:AttachPolicy` – 将策略附加到根、组织单元（OU）或单个账户。
- `organizations:DetachPolicy` – 将策略从目标根、组织单元（OU）或账户分离。
- `organizations:EnablePolicyType` – 在根中启用一种策略类型。
- `organizations:DisablePolicyType` – 在根中禁用一种组织策略类型。
- `organizations:TagResource` – 将一个或多个标签添加到指定的资源。
- `organizations:UntagResource` – 从指定的资源中移除具有指定密钥的任何标签。
- `organizations:ListTagsForResource` – 列出附加到指定资源的标签。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 AWSInspector2OrganizationsAccess。

AWS 托管策略：AmazonInspector2FullAccess

您可以将 `AmazonInspector2FullAccess` 策略附加到 IAM 身份。

此策略授予允许完全访问 Amazon Inspector 的权限。

> **Important**
>
> 为了增强安全性并限制对 Inspector 2 服务主体的权限，我们建议您使用 AmazonInspector2FullAccess _v2。

权限详细信息

该策略包含以下权限。

- `inspector2` – 允许完全访问 Amazon Inspector 功能。
- `iam` – 支持 Amazon Inspector 创建服务相关角色 `AWSServiceRoleForAmazonInspector2` 和 `AWSServiceRoleForAmazonInspector2Agentless`。Amazon Inspector 如

果要执行检索 Amazon EC2 实例、Amazon ECR 存储库和容器映像的相关信息等操作，则需要 AWSServiceRoleForAmazonInspector2。此外，Amazon Inspector 如果要分析您的 VPC 网络并描述与您的组织关联的账户，也需要它。如果 Amazon Inspector 要执行检索 Amazon EC2 实例和 Amazon EBS 快照的相关信息等操作，则需要 AWSServiceRoleForAmazonInspector2Agentless。还需要解密使用密钥加密的 Amazon EBS 快照。 AWS KMS 有关更多信息，请参阅 对 Amazon Inspector 使用服务相关角色。

- organizations — 允许管理员将 Amazon Inspector 用于 AWS Organizations中的组织。当您在中激活 Amazon Inspector 的可信访问权限时 AWS Organizations，委托管理员账户的成员可以管理其组织中的设置并查看调查结果。
- codeguru-security— 允许管理员使用 Amazon Inspector 检索信息代码片段并更改 CodeGuru 安全部门存储的代码的加密设置。有关更多信息，请参阅 对调查发现中的代码进行静态加密。

要查看此策略的权限，请参阅《AWS 托管策略参考指南》FullAccess中的 AmazonInspector2。

AWS 托管策略：AmazonInspector2ReadOnlyAccess

您可以将 AmazonInspector2ReadOnlyAccess 策略附加到 IAM 身份。

此策略授予允许对 Amazon Inspector 进行只读访问的权限。

权限详细信息

该策略包含以下权限。

- inspector2 – 允许以只读方式访问 Amazon Inspector 功能。
- organizations— 允许查看有关组织的 Amazon Inspector 覆盖范围 AWS Organizations 的详细信息。此外，还允许通过按 Inspector 策略类型ListPolicies进行筛选、通过查看委托资源策略以及通过DescribeResourcePolicy查看应用于账户的有效 Inspector 策略来查看 Inspector 组织政策DescribeEffectivePolicy。这使用户能够了解通过组织策略建立的集中式检查员支持，而无需对其进行修改。
- codeguru-security— 允许从 “ CodeGuru 安全” 中检索代码片段。还允许查看存储在 “ CodeGuru 安全” 中的代码的加密设置。

要查看此策略的权限，请参阅《AWS 托管策略参考指南》ReadOnlyAccess中的 AmazonInspector2。

AWS 托管策略：AmazonInspector2ManagedCisPolicy

可以将 AmazonInspector2ManagedCisPolicy 策略附加到您的 IAM 实体。此策略应附加到向 Amazon EC2 实例授予权限以对实例运行 CIS 扫描的角色。您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

权限详细信息

该策略包含以下权限。

- inspector2 - 支持访问用于运行 CIS 扫描的操作。

要查看此策略的权限，请参阅《AWS 托管策略参考指南》ManagedCisPolicy中的 AmazonInspector2。

AWS 托管策略：AmazonInspector2ServiceRolePolicy

无法将 AmazonInspector2ServiceRolePolicy 策略附加到 IAM 实体。将此策略附加到允许 Amazon Inspector 代表您执行操作的服务相关角色。有关更多信息，请参阅 对 Amazon Inspector 使用服务相关角色。

AWS 托管策略：AmazonInspector2AgentlessServiceRolePolicy

无法将 AmazonInspector2AgentlessServiceRolePolicy 策略附加到 IAM 实体。将此策略附加到允许 Amazon Inspector 代表您执行操作的服务相关角色。有关更多信息，请参阅 对 Amazon Inspector 使用服务相关角色。

AWS 托管策略：AmazonInspector2ManagedTelemetryPolicy

可以将 AmazonInspector2ManagedTelemetryPolicy 策略附加到您的 IAM 实体。该政策授予 Amazon Inspector 遥测操作的权限，允许该服务收集和传输包裹库存数据以进行漏洞扫描。

权限详细信息

该策略包含以下权限。

- inspector2-telemetry— 允许访问包裹库存数据传输的操作。

要查看有关策略的更多详细信息，包括最新版本的 JSON 策略文档，请参阅《AWS 托管策略参考指南》ManagedTelemetryPolicy中的 AmazonInspector2。

Amazon Inspector 更新 AWS 了托管政策

查看自该服务开始跟踪这些更改以来对 Amazon Inspector AWS 托管政策的更新的详细信息。要获得有关此页面更改的自动提醒，请订阅 Amazon Inspector 文档历史记录页面上的 RSS 源。

更改	描述	日期
AWSInspector2OrganizationsAccess - 新策略	Amazon Inspector 添加了一项新的托管策略，该策略授予通过 AWS Organizations 策略启用和管理亚马逊 Inspector 所需的权限。	2026 年 3 月 3 日
AmazonInspector2 ManagedTelemetryPolicy — 新政策	Amazon Inspector 添加了一项新的托管策略，该策略授予对 Amazon Inspector 遥测操作的权限，允许该服务收集和传输包裹库存数据以进行漏洞扫描。	2026 年 2 月 5 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 添加了一项新权限，允许亚马逊 Inspector 描述用于网络可访问性分析的防火墙元数据。此外，Amazon Inspector 还添加了额外的资源范围界定，以允许 Amazon Inspector 创建、更新和启动与 SSM 文档的 SSM 关联。`AWS-ConfigureAWSPackage`	2026 年 2 月 3 日
AmazonInspector2 FullAccess _v2 和 AmazonInspector2	Amazon Inspector 增加了新的权限，允许保单持有人查看 Inspector 组织策略和委托配置	2025 年 11 月 14 日

更改	描述	日期
ReadOnlyAccess — 对现有策略的更新	。这支持通过 AWS Organizations 策略对 Inspector 的启用进行集中管理和可见性。	
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许亚马逊 Inspector AWS Organizations 策略强制启用和禁用 Amazon Inspector 。	2025 年 11 月 10 日
AmazonInspector2 FullAccess_v2 — 新政策	Amazon Inspector 已增加了一个新的托管式策略，提供对 Amazon Inspector 的完全访问权限以及对其他相关服务的访问权限。	2025 年 7 月 3 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 已增加了一个新权限，允许 Amazon Inspector 描述 IP 地址和互联网网关。	2025 年 4 月 29 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 已增加了一个新权限，允许对 Amazon ECS 和 Amazon EKS 操作进行只读访问。	2025 年 3 月 25 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，让 Amazon Inspector 可以在 AWS Lambda中返回函数标签。	2024 年 7 月 31 日

更改	描述	日期
AmazonInspector2 FullAccess — 对现有政策的更新	Amazon Inspector 增加了相应的权限，让 Amazon Inspector 可以创建服务相关角色 `AWSServiceRoleForAmazonInspector2Agentless` ，从而让用户可以在启用 Amazon Inspector 时执行基于代理的扫描和无代理扫描。	2024 年 4 月 24 日
AmazonInspector2 ManagedCisPolicy — 新政策	Amazon Inspector 增加了一个新的托管式策略，您可以将其用作实例配置文件的一部分，以便对实例进行 CIS 扫描。	2024 年 1 月 23 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，让 Amazon Inspector 可以在目标实例上启动 CIS 扫描。	2024 年 1 月 23 日
AmazonInspector2 AgentlessServiceRolePolicy — 新政策	Amazon Inspector 添加了一项新的服务相关角色策略，以允许对 EC2 实例进行无代理扫描。	2023 年 11 月 27 日
AmazonInspector2 ReadOnlyAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许只读用户检索程序包漏洞调查发现的漏洞情报详细信息。	2023 年 9 月 22 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许 Amazon Inspector 扫描属于 Elastic Load Balancing 目标组的 Amazon EC2 实例的网络配置。	2023 年 8 月 31 日

更改	描述	日期
AmazonInspector2 ReadOnlyAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许只读用户为其资源导出软件材料清单 (SBOM)。	2023 年 6 月 29 日
AmazonInspector2 ReadOnlyAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许只读用户检索其账户的 Lambda 代码扫描结果的加密设置详情。	2023 年 6 月 13 日
AmazonInspector2 FullAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许用户配置客户托管的 KMS 密钥，来加密 Lambda 代码扫描结果中的代码。	2023 年 6 月 13 日
AmazonInspector2 ReadOnlyAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许只读用户检索其账户的 Lambda 代码扫描状态和结果的详细信息。	2023 年 5 月 2 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 添加了新的权限，允许亚马逊检查员在您激活 Lambda 扫描时在您的账户中创建 AWS CloudTrail 与服务相关的渠道。这样，Amazon Inspector 就可以监控您账户中的 CloudTrail 事件。	2023 年 4 月 30 日
AmazonInspector2 FullAccess — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许用户检索 Lambda 代码扫描漏洞调查发现的详细信息。	2023 年 4 月 21 日

更改	描述	日期
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，让 Amazon Inspector 可以向 Amazon EC2 Systems Manager 发送有关客户为 Amazon EC2 深度检查定义的自定义路径的信息。	2023 年 4 月 17 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 添加了新的权限，允许亚马逊检查员在您激活 Lambda 扫描时在您的账户中创建 AWS CloudTrail 与服务相关的渠道。这样，Amazon Inspector 就可以监控您账户中的 CloudTrail 事件。	2023 年 4 月 30 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了新的权限，允许亚马逊 Inspector 请求扫描 AWS Lambda 函数中的开发者代码，并从亚马逊 CodeGuru 安全部门接收扫描数据。此外，Amazon Inspector 还增加了审查 IAM policy 的权限。Amazon Inspector 使用这些信息扫描 Lambda 函数中是否存在代码脆弱性。	2023 年 2 月 28 日

更改	描述	日期
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 添加了一条新语句，允许 Amazon Inspector 检索 CloudWatch 有关上次调用 AWS Lambda 函数的时间的信息。Amazon Inspector 使用这些信息将扫描重点放在您环境中过去 90 天内处于活动状态的 Lambda 函数上。	2023 年 2 月 20 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 添加了一个新声明，允许亚马逊 Inspector 检索有关 AWS Lambda 函数的信息，包括与每个函数关联的每个层版本。Amazon Inspector 使用这些信息扫描 Lambda 函数中是否存在安全漏洞。	2022 年 11 月 28 日
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 增加了一项新操作，允许 Amazon Inspector 描述 SSM 关联的执行情况。此外，Amazon Inspector 还增加了额外的资源范围，允许 Amazon Inspector 使用 `AmazonInspector2` 拥有的 SSM 文档创建、更新、删除和启动 SSM 关联。	2022 年 8 月 31 日
AmazonInspector2 对现有政策的ServiceRolePolicy更新	Amazon Inspector 更新了政策的资源范围，允许亚马逊 Inspector 收集其他 AWS 分区中的软件库存。	2022 年 8 月 12 日

更改	描述	日期
AmazonInspector2 ServiceRolePolicy — 对现有政策的更新	Amazon Inspector 重组了操作的资源范围，允许 Amazon Inspector 创建、删除和更新 SSM 关联。	2022 年 8 月 10 日
AmazonInspector2 ReadOnlyAccess — 新政策	Amazon Inspector 增加了一项新策略，允许以只读方式访问 Amazon Inspector 功能。	2022 年 1 月 21 日
AmazonInspector2 FullAccess — 新政策	Amazon Inspector 增加了一项新策略，允许完全访问 Amazon Inspector 功能。	2021 年 11 月 29 日
AmazonInspector2 ServiceRolePolicy — 新政策	Amazon Inspector 增加了一项新策略，允许 Amazon Inspector 代表您在其他服务中执行操作。	2021 年 11 月 29 日
Amazon Inspector 开始跟踪更改	Amazon Inspector 开始跟踪其 AWS 托管政策的变更。	2021 年 11 月 29 日

对 Amazon Inspector 使用服务相关角色

Amazon Inspector 使用名`AWSServiceRoleForAmazonInspector2`为的 AWS Identity and Access Management (IAM) 服务相关角色。此服务相关角色是与 Amazon Inspector 直接相关的 IAM 角色。它由 Amazon Inspector 预定义，它包括亚马逊检查员 AWS 服务 代表您致电他人所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon Inspector，因为您不必手动添加必要的权限。Amazon Inspector 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon Inspector 可以代入该角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其他 IAM 实体。

必须配置权限，允许 IAM 实体（如组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。只有在首先删除服务相关角色的相关资源后，才能删除该角色。这将保护您的 Amazon Inspector 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并在服务相关角色列中查找标有 “是” 的服务。选择带有链接的是可以查看该服务的服务相关角色文档。

Amazon Inspector 的服务相关角色权限

Amazon Inspector 使用名为 AWSServiceRoleForAmazonInspector2 的托管式策略。该服务相关角色信任 inspector2.amazonaws.com 服务担任该角色。

该角色的权限策略名为 AmazonInspector2ServiceRolePolicy，允许 Amazon Inspector 执行以下任务：

- 使用 Amazon Elastic Compute Cloud (Amazon EC2) 操作检索有关实例和网络路径的信息。
- 使用 AWS Systems Manager 操作从您的 Amazon EC2 实例中检索库存，并从自定义路径中检索有关第三方包裹的信息。
- 使用 AWS Systems Manager SendCommand操作调用目标实例的 CIS 扫描。
- 使用 Amazon Elastic Container Registry 操作检索有关您的容器映像的信息。
- 使用 AWS Lambda 操作来检索有关您的 Lambda 函数的信息。
- 使用 AWS Organizations 操作来描述关联的账户。
- 使用 CloudWatch 操作来检索有关上次调用 Lambda 函数的时间的信息。
- 使用“选择 IAM”操作检索可能在您的 Lambda 代码中造成安全漏洞的 IAM policy 的相关信息。
- 使用 Amazon Q 操作对您的 Lambda 函数中的代码执行扫描。Amazon Inspector 使用以下 Amazon Q 操作：
  - codeguru-security: CreateScan — 授予创建 Amazon Q; 扫描的权限。
  - codeguru-security：GetScan — 授予检索 Amazon Q 扫描元数据的权限。
  - codeguru-security：ListFindings — 授予检索 Amazon Q 生成的调查结果的权限
  - codeguru-security：DeleteScansByCategory — 授予 Amazon Q 删除由 Amazon Inspector 启动的扫描的权限。
  - codeguru-security：BatchGetFindings — 授予检索 Amazon Q 生成的一批特定调查结果的权限
- 使用“选择 Elastic Load Balancing”操作对属于 Elastic Load Balancing 目标组的 EC2 实例执行网络扫描。
- 使用 Amazon ECS 和 Amazon EKS 操作允许进行只读访问，以查看集群和任务并描述任务。
- 使用 AWS Organizations 操作列出跨组织的 Amazon Inspector 的委托管理员。
- 使用 Amazon Inspector 操作可跨组织启用和禁用 Amazon Inspector。
- 使用 Amazon Inspector 操作可跨组织指定委派管理员账户并关联成员账户。

> **Note**
>
> Amazon Inspector 不再使用它 CodeGuru 来执行 Lambda 扫描。 AWS 将于 2025 年 11 月 20 CodeGuru 日停止提供支持。有关更多信息，请参阅终止对 CodeGuru 安全的支持。Amazon Inspector 现在使用 Amazon Q 执行 Lambda 扫描，不需要本节中描述的权限。

要查看此策略的权限，请参阅《AWS 托管策略参考指南》ServiceRolePolicy中的 AmazonInspector2。

为 Amazon Inspector 创建服务相关角色

您无需手动创建服务关联角色。当你在 AWS 管理控制台、或 AWS API 中激活 Amazon Inspector 时，Amazon Inspector 会为你创建与服务相关的角色。 AWS CLI

为 Amazon Inspector 编辑服务相关角色

Amazon Inspector 不允许编辑 `AWSServiceRoleForAmazonInspector2` 服务相关角色。在创建服务相关角色后，您无法更改角色的名称，因为可能有多个实体会引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除适用于 Amazon Inspector 的服务相关角色

如果您不再使用 Amazon Inspector，我们建议您删除 `AWSServiceRoleForAmazonInspector2` 服务相关角色。在删除角色之前，您必须在每个激活该角色 AWS 区域 的地方停用 Amazon Inspector。停用 Amazon Inspector 时，它不会为您删除该角色。因此，如果您再次激活 Amazon Inspector，它可以使用现有角色。这样，您就可以避免出现未监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

如果删除此服务相关角色然后需要再次创建它，则可以使用相同的流程在您的账户中重新创建此角色。激活 Amazon Inspector 时，Amazon Inspector 会为您重新创建服务相关角色。

> **Note**
>
> 如果在您试图删除资源时，Amazon Inspector 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟，然后再次尝试操作。

您可以使用 IAM 控制台 AWS CLI、或 AWS API 删除`AWSServiceRoleForAmazonInspector2`服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务关联角色。

Amazon Inspector 无代理扫描的服务相关角色权限

Amazon Inspector 无代理扫描使用名为 `AWSServiceRoleForAmazonInspector2Agentless` 的服务相关角色。这个 SLR 允许 Amazon Inspector 在您的账户中创建 Amazon EBS 卷快照，然后访问该快照中的数据。该服务相关角色信任 `agentless.inspector2.amazonaws.com` 服务担任该角色。

> **Important**
>
> 此服务相关角色中的语句会阻止 Amazon Inspector 对您使用 `InspectorEc2Exclusion` 标签从扫描中排除的任何 EC2 实例执行无代理扫描。此外，当用于加密卷的 KMS 密钥带有 `InspectorEc2Exclusion` 标签时，这些语句会阻止 Amazon Inspector 访问相应卷中的加密数据。有关更多信息，请参阅 从 Amazon Inspector 扫描中排除实例。

该角色的权限策略名为 `AmazonInspector2AgentlessServiceRolePolicy`，允许 Amazon Inspector 执行以下任务：

- 使用 Amazon Elastic Compute Cloud（Amazon EC2）操作检索有关 EC2 实例、卷和快照的信息。
  - 使用 Amazon EC2 标记操作，用 `InspectorScan` 标签键为扫描的快照添加标签。
  - 使用 Amazon EC2 快照操作创建快照，用 `InspectorScan` 标签键为其添加标签，然后删除 Amazon EBS 卷带有 `InspectorScan` 标签键的快照。
- 使用 Amazon EBS 操作，从带有 `InspectorScan` 标签键的快照中检索信息。
- 使用选择 AWS KMS 解密操作来解密使用客户托管密钥加密的 AWS KMS 快照。当用于加密快照的 KMS 密钥带有 `InspectorEc2Exclusion` 标签时，Amazon Inspector 不会解密相应快照。

该角色使用以下权限策略进行配置：

JSON

```
{
 "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/*"
  }
 ]
}
```

创建用于无代理扫描的服务相关角色

您无需手动创建服务关联角色。当你在 AWS 管理控制台、或 AWS API 中激活 Amazon Inspector 时，Amazon Inspector 会为你创建与服务相关的角色。 AWS CLI

编辑用于无代理扫描的服务相关角色

Amazon Inspector 不允许编辑 `AWSServiceRoleForAmazonInspector2Agentless` 服务相关角色。在创建服务相关角色后，您无法更改角色的名称，因为可能有多个实体会引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除用于无代理扫描的服务相关角色

如果不再需要使用某个需要服务相关角色的特征或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。

> **Important**
>
> 要删除 `AWSServiceRoleForAmazonInspector2Agentless` 角色，您必须在所有支持无代理扫描的区域中将扫描模式设置为基于代理。

使用 IAM 手动删除服务关联角色

使用 IAM 控制台 AWS CLI、或 AWS API 删除 AWSService RoleForAmazonInspector 2Agentless 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

Amazon Inspector 身份和访问问题排查

您可以使用以下信息，帮助诊断和修复在使用 Amazon Inspector 和 IAM 时可能遇到的常见问题。

主题

我无权在 Amazon Inspector 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `inspector2:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 inspector2:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `inspector2:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，指明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon Inspector。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon Inspector 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我的 Amazon Inspector 资源 AWS 账户

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Inspector 是否支持这些功能，请参阅Amazon Inspector 如何与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

监控 Amazon Inspector

监控是维护 Amazon Inspector 和其他 AWS 解决方案的可用性、可靠性和性能的重要组成部分。 AWS 提供了用于监控 Amazon Inspector、报告出现的问题并采取措施修复这些问题的工具：

- Amazon EventBridge 是一项使用事件将应用程序组件连接在一起的 AWS 服务，使您可以更轻松地构建可扩展的事件驱动型应用程序。 EventBridge 提供来自您的应用程序、 Software-as-a-Service (SaaS) 应用程序以及 AWS 服务和路由的实时数据流，因此您可以监控服务中发生的事件并构建事件驱动的架构。
- AWS CloudTrail是一项捕获由您或代表您进行的 API 调用和相关事件的 AWS 服务 AWS 账户。CloudTrail 将日志文件传送到您指定的 Amazon S3 存储桶，这样您就可以识别哪些用户和账户拨打了电话 AWS、发出呼叫的源 IP 地址以及调用的发生时间。

使用记录亚马逊 Inspector API 调用 AWS CloudTrail

Amazon Inspector 与 AWS CloudTrail一项服务集成，该服务提供了 IAM 用户或角色或 Amazon Inspect AWS 服务 or 中的角色所采取的操作的记录。 CloudTrail 将 Amazon Inspector 的所有 API

调用捕获为事件。捕获调用中包括通过 Amazon Inspector 控制台的调用和对 Amazon Inspector API 操作的调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括针对 Amazon Inspector 的事件。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台的 Event history（事件历史记录）中查看最新事件。使用收集的信息 CloudTrail，您可以确定：

- 向 Amazon Inspector 发出的请求。
- 已从中发出请求的 IP 地址。
- 谁发出了请求。
- 发出请求的时间。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

Amazon Inspector 中的信息 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 Amazon Inspector 中发生活动时，该活动会与其他 CloudTrail AWS 服务 事件一起记录在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的事件 AWS 账户，包括 Amazon Inspector 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务 ，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅以下主题：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个账户的 CloudTrail 日志文件
- 接收来自多个区域的 CloudTrail 日志文件

Amazon Inspector 的所有操作都由记录 CloudTrail。Amazon Inspector 可以执行的所有操作都记录在 Amazon Inspector API 参考中。例如，对 `CreateFindingsReport`、`ListCoverage` 和 `UpdateOrganizationConfiguration` 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。

- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Amazon Inspector 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件表示来自任何源的单个请求。事件包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

Amazon Inspector 扫描中的信息 CloudTrail

Amazon Inspector Scan 已与集成 CloudTrail。所有 Amazon Inspector Scan API 操作都记录为管理事件。有关亚马逊 Inspector 登录的亚马逊 Inspector Scan API 操作的列表 CloudTrail，请参阅《亚马逊检查器 API 参考》中的 Amazon Inspector Scan。

以下示例显示了演示该ScanSbom操作的 CloudTrail 日志条目：


```
{
    "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 的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

Amazon Inspector 故障恢复能力

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离、隔离的可用区，这些可用区连接到低延迟、高吞吐量和高度冗余的网络。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

Amazon Inspector 基础设施安全性

作为一项托管服务，Amazon Inspector 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon Inspector。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

Amazon Inspector 中的事件响应

AWS非常重视安全性。正如 “云安全” 下的AWS 分担责任模型中所述 AWS ，负责保护在云中运行所有服务的基础架构。 AWS AWS 还负责与 Amazon Inspector 服务相关的任何事件响应。

作为 AWS 客户，您有责任维护 AWS 云端的安全。这意味着您可以控制您选择实施的安全性，其中包括您访问的所有 AWS 工具和功能。这也意味着，您还需要负责您在责任共担模式中的事件响应部分。

通过为在 AWS 云端运行的应用程序建立满足所有目标的安全基准，您可以检测出可以响应的偏差。由于事件响应是一个复杂的主题，因此请查看以下资源，来更好地了解事件响应的影响以及您的选择对企业目标有怎样的影响：AWS Security Incident Response Guide、AWS Security Best Practices 以及 AWS Cloud Adoption Framework: Security Perspective。

使用接口端点（ AWS PrivateLink ）访问 Amazon Inspector

您可以使用 AWS PrivateLink 在您的 VPC 和 Amazon Inspector 之间创建私有连接。您可以像在您的 VPC 中一样访问 Amazon Inspector，无需使用互联网网关、NAT 设备、VPN Direct Connect 连接或连接。VPC 中的实例不需要公有 IP 地址即可访问 Amazon Inspector。

您可以通过创建由 AWS PrivateLink提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管式网络接口，用作发往 Amazon Inspector 的流量的入口点。

有关更多信息，请参阅AWS PrivateLink 指南 AWS PrivateLink中的AWS 服务 通过访问。

Amazon Inspector 注意事项

在为 Amazon Inspector 设置接口端点之前，请先查看《AWS PrivateLink 指南》中的注意事项。

Amazon Inspector 支持通过接口端点调用其所有 API 操作。

Amazon Inspector 不支持 VPC 端点策略。默认情况下，允许通过接口端点对 Amazon Inspector 进行完全访问。或者，也可以将安全组与端点网络接口关联，以控制通过接口端点流向 Amazon Inspector 的流量。

为 Amazon Inspector 创建接口端点

您可以使用亚马逊 VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 Amazon Inspector 创建接口终端节点。有关更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

在为 Amazon Inspector 创建接口端点时，请使用以下任一服务名称：

```
com.amazonaws.region.inspector2
```

```
com.amazonaws.region.inspector-scan
```

*region*替换为适用的 AWS 区域 代码 AWS 区域。

如果为接口端点启用私有 DNS，则可以使用其默认的区域 DNS 名称向 Amazon Inspector 发出 API 请求，例如美国东部（弗吉尼亚州北部）为 `service-name.us-east-1.amazonaws.com` 或 `service-name.us-east-1.api.aws.com`。

Amazon Inspector 集成

Amazon Inspector 与其他 AWS 服务集成。这些服务可以从 Amazon Inspector 摄取数据，以便您可以通过不同的方式查看调查发现。要了解更多信息，请查看以下集成选项。

将 Amazon Inspector 与 AWS Organizations

AWS Organizations帮助您集中管理和治理您的 AWS 环境。您可以使用 AWS Organizations 策略在组织中的多个账户中自动启用和管理 Amazon Inspector。

Amazon Inspector 组织政策允许您：

- 在整个组织中集中启用 Amazon Inspector 扫描类型（EC2、ECR、Lambda、代码存储库）
- 对加入该组织的新账户自动启用 Amazon Inspector
- 在各组织单位之间实施一致的扫描覆盖范围
- 防止成员帐户禁用必需的扫描

组织策略控制资源类型的启用，而委派的管理员则保留对扫描配置设置的控制权。有关组织策略如何与委派管理员和成员账户权限交互的信息，请参阅使用 Amazon Inspector 管理多个账户 AWS Organizations。有关创建亚马逊 Inspector 策略的详细说明，请参阅亚马逊 Inspector 政策 AWS Organizations 文档。

Amazon Inspector 与 Amazon ECR 集成

Amazon Elastic Container Registry (Amazon ECR) AWS是一个支持私有注册表的托管容器镜像注册表。Amazon ECR 私有注册表在可用性和可扩展性都非常高的架构中托管容器映像。可以使用 Amazon Inspector 扫描驻留在 Amazon ECR 存储库中的容器映像，查找易受攻击的操作系统程序包和编程语言程序包。有关更多信息，请参阅 Amazon Inspector 与 Amazon Elastic Container Registry (Amazon ECR) 集成。

Amazon Inspector 与 AWS Security Hub CSPM

AWS Security Hub CSPM提供您的安全状态的全面视图， AWS 并帮助您根据安全行业标准和最佳实践检查您的环境 Security Hub CSPM 从 AWS 帐户、服务和支持产品收集安全数据。您可以使用 Security Hub CSPM 来摄取 Amazon Inspector 的调查结果数据，并为所有集成 AWS 服务和 AWS 合

作伙伴网络产品中的调查结果创建一个中心位置。有关更多信息，请参阅 Amazon Inspector 与 AWS Security Hub CSPM。

Amazon Inspector 与 Amazon Elastic Container Registry (Amazon ECR) 集成

Amazon Elastic 容器注册表是一个完全托管的容器注册表，支持 Docker 和 OCI 镜像和工件 AWS 。如果您使用 Amazon ECR，则可以为容器注册表激活增强扫描。如果激活增强扫描，Amazon Inspector 会自动检测容器映像，并对其进行扫描，以查找易受攻击的操作系统程序包和编程语言程序包。此集成使您可以在 Amazon ECR 控制台中查看容器映像的 Amazon Inspector 调查发现并管理扫描的频率和范围。有关更多信息，请参阅使用 Amazon Inspector 扫描 Amazon ECR 容器映像。

激活集成

您可以通过使用 Amazon Inspector 控制台或 API 激活 Amazon Inspector 扫描来激活此集成，也可以通过 Amazon ECR 控制台或 API 配置您的存储库以使用 Amazon Inspector 的增强扫描，从而激活此集成。

有关通过 Amazon Inspector 激活集成的更多信息，请参阅Amazon Inspector 中的自动扫描类型。

有关在 Amazon ECR 中激活和配置增强扫描的信息，请参阅 Amazon ECR 用户指南中的增强扫描。

使用与多账户环境的集成

如果您是多账户环境的成员，则可以通过 Amazon ECR 激活增强型扫描。但是，一旦激活，则只能由您的 Amazon Inspector 委托管理员停用。如果停用，则恢复为基本扫描。有关更多信息，请参阅 停用 Amazon Inspector。

Amazon Inspector 与 AWS Security Hub CSPM

Security Hub CSPM 提供了您的安全状态的全面视图。 AWS这将帮助您根据安全行业标准和最佳实践来检查您的环境。Security Hub CSPM 从 AWS 账户、服务和支持产品收集安全数据。您可以使用此信息来分析安全趋势并确定安全问题。当你激活 Amazon Inspector 与 Security Hub CSPM 的集成时，Amazon Inspector 可以将调查结果发送到 Security Hub CSPM，而 Security Hub CSPM 可以将这些发现作为安全态势的一部分进行分析。

Security Hub CSPM 将安全问题作为发现结果进行跟踪。有些发现可能是由于在其他 AWS 服务或第三方产品中检测到的安全问题所致。Security Hub CSPM 使用一组规则来检测安全问题并生成调查结果，并提供工具，以便您可以管理发现。Amazon Inspector 的调查结果结束后，Security Hub CSPM

将存档亚马逊检查员的调查结果。您还可以查看调查发现历史记录和调查发现详细信息，以及跟踪针对调查发现的调查状态。

Security Hub CSPM 以AWS 安全调查结果格式 (ASFF) 处理调查结果。此格式包括唯一标识符、严重性级别、受影响资源、修复指导、工作流状态以及上下文信息等详细内容。

> **Note**
>
> Amazon Inspector 代码安全性生成的安全调查发现不适用于此集成。但是，您可以在 Amazon Inspector 控制台中和通过 Amazon Inspector API 访问这些特定调查发现。

主题

在中查看亚马逊 Inspector 的调查结果 AWS Security Hub CSPM

你可以在 Security Hub CSPM 中查看 Amazon Inspector Classic 和亚马逊 Inspector 的调查结果。

> **Note**
>
> 要仅筛选 Amazon Inspector 调查发现，请将 `"aws/inspector/ProductVersion": "2"` 添加到筛选栏中。此筛选条件排除 Security Hub CSPM 控制面板中的 Amazon Inspector Classic 调查结果。

Amazon Inspector 调查发现示例

```
{
  "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"
}
```

激活和配置 Amazon Inspector 与 Security Hub CSPM 的集成

您可以通过启用 Security Hub CSPM AWS Security Hub CSPM 来激活 Amazon Inspector 与的集成。启用 Security Hub CSPM 后，Amazon Inspector 与 AWS Security Hub CSPM 的集成将自动激活，Amazon Inspector 开始使用安全调查格式 (ASFF) 将其所有发现结果发送到 Sec AWS ur ity Hub CSPM。

使用组织策略从 Security Hub CSPM 激活 Amazon Inspector

您可以直接从 Security Hub CSPM 控制台使用组织策略管理整个 AWS 组织的 Amazon Inspector 激活。这种集中式方法允许您通过组织级别的策略管理同时启用 Amazon Inspector 扫描多个账户。

有关使用组织策略管理通过 Security Hub CSPM 激活 Amazon Inspector 的详细说明，请参阅用户指南中的管理 Security Hub CSPM 的委托管理员账户。AWS Security Hub CSPM

禁用来自集成的调查发现流

要阻止 Amazon Inspector 向 Security Hub CSPM 发送调查结果，你可以使用 Security Hub CSPM 控制台或 API 然后... AWS CLI

在 Security Hub CSPM 中查看亚马逊 Inspector 的安全控制措施

Security Hub CSPM 会分析受支持产品 AWS 和第三方产品的发现，并根据规则进行自动和持续的安全检查，以生成自己的调查结果。这些规则以安全控件表示，可帮助您确定是否满足标准中的要求。

Amazon Inspector 使用安全控件来检查是否启用或应该启用 Amazon Inspector 的特征。这些功能如下所示：

- Amazon EC2 扫描
- Amazon ECR 扫描
- Lambda 标准扫描
- Lambda 代码扫描

有关更多信息，请参阅《AWS Security Hub CSPM 用户指南》中的 Amazon Inspector 控件。

Amazon Inspector 支持的操作系统和编程语言

Amazon Inspector 可以扫描安装在以下设备上的软件应用程序：

- Amazon Elastic Compute Cloud（Amazon EC2）实例

> **Note**
>
> 对于 Amazon EC2 实例，Amazon Inspector 可以在支持基于代理扫描的操作系统中扫描程序包漏洞。Amazon Inspector 还可以扫描支持混合扫描的操作系统和编程语言中的程序包漏洞。Amazon Inspector 不会扫描工具链漏洞。用于构建应用程序的编程语言编译器的版本引入了这些漏洞。

- 存储在 Amazon Elastic Container Registry（Amazon ECR）存储库中的容器映像

> **Note**
>
> 对于 ECR 容器映像，Amazon Inspector 可扫描操作系统和编程语言程序包漏洞。Amazon Inspector 还支持 Chainguard 和 Minimus 提供的强化图像。Amazon Inspector 不会扫描中的工具链漏洞 Rust ——用于构建应用程序的编程语言编译器版本引入了这些漏洞。

- AWS Lambda 函数

> **Note**
>
> 对于 Lambda 函数，Amazon Inspector 可以扫描编程语言程序包漏洞和代码漏洞。Amazon Inspector 不会扫描工具链漏洞。用于构建应用程序的编程语言编译器的版本引入了这些漏洞。

当 Amazon Inspector 扫描资源时，Amazon Inspector 会获取 50 多个数据源，以生成常见漏洞和漏洞的调查结果（CVEs）。这些来源的示例包括供应商安全公告、数据源和威胁情报源，以及国家漏洞数据库（NVD）和 MITRE。Amazon Inspector 每天至少更新一次来自源的漏洞数据。

要让 Amazon Inspector 扫描资源，相应资源必须运行支持的操作系统或使用支持的编程语言。本节中的主题列出了 Amazon Inspector 针对不同资源和扫描类型支持的操作系统、编程语言以及运行时系统。此外，它们还列出了已停用的操作系统。

> **Note**
>
> 在供应商停止对操作系统的支持后，Amazon Inspector 只能为相应操作系统提供有限的支持。

主题

支持的操作系统

本节列出了 Amazon Inspector 支持的操作系统。

支持的操作系统：Amazon EC2 扫描

下表列出了 Amazon Inspector 支持用于扫描 Amazon EC2 实例的操作系统。它为每个操作系统指定供应商安全公告，以及哪些操作系统支持基于代理的扫描和无代理扫描。

使用基于代理的扫描方法时，可以将 SSM Agent 配置为对所有符合条件的实例执行连续扫描。Amazon Inspector 建议配置版本高于 3.2.2086.0 的 SSM Agent。有关更多信息，请参阅《Amazon EC2 Systems Manager 用户指南》中的使用 SSM Agent。

仅默认程序包管理器存储库（rpm 和 dpkg）支持 Linux 操作系统检测，不包括第三方应用程序、扩展支持存储库（RHEL EUS、E4S、AUS 和 TUS）以及可选存储库（应用程序流）。Amazon Inspector 会扫描正在运行的内核，以查看是否存在漏洞。对于某些操作系统（例如 Ubuntu），需要重启才能在活动调查发现中显示升级信息。

操作系统	版本	供应商安全公告	无代理扫描支持	基于代理的扫描支持
AlmaLinux	8	勘误表 CVE	支持	是
AlmaLinux	9	勘误表 CVE	支持	是
AlmaLinux	10	勘误表 CVE	否	是

操作系统	版本	供应商安全公告	无代理扫描支持	基于代理的扫描支持
亚马逊 Linux (AL2)	AL2	ALAS 勘误表 CVE	支持	是
亚马逊 Linux 2023 (AL2023)	AL2023	ALAS 勘误表 CVE	支持	是
Bottlerocket	1.7.0 及更高版本	勘误表 CVE	否	是
Debian 服务器 (Bullseye)	11	DSA CVE	支持	是
Debian 服务器 (Bookworm)	12	DSA CVE	支持	是
Debian 服务器（Trixie）	13	DSA CVE	支持	是
Fedora	42	Errata CVE	是	是
OpenSUSE Leap	15.6	勘误表 CVE	支持	是
Oracle Linux (Oracle)	8	勘误表 CVE	支持	是
Oracle Linux (Oracle)	9	勘误表 CVE	支持	是
Oracle Linux (Oracle)	10	勘误表 CVE	否	是
Red Hat Enterprise Linux (RHEL)	8	RHEL VEX CVE	支持	是
Red Hat Enterprise Linux (RHEL)	9	RHEL VEX CVE	支持	是

操作系统	版本	供应商安全公告	无代理扫描支持	基于代理的扫描支持
Red Hat Enterprise Linux (RHEL)	10	RHEL VEX CVE	否	是
Rocky Linux	8	勘误表 CVE	支持	是
Rocky Linux	9	勘误表 CVE	支持	是
Rocky Linux	10	勘误表 CVE	否	是
SUSE Linux Enterprise Server (SLES)	15.7	SUSE CVE	支持	是
Ubuntu (Xenial)	16.04	USN、Ubuntu Pro（esm-infra 和 esm-apps）	支持	是
Ubuntu (Bionic)	18.04	USN、Ubuntu Pro（esm-infra 和 esm-apps）	支持	是
Ubuntu (Focal)	20.04	USN、Ubuntu Pro（esm-infra 和 esm-apps）	支持	是
Ubuntu (Jammy)	22.04	USN、Ubuntu Pro（esm-infra 和 esm-apps）	支持	是
Ubuntu (Noble Numbat)	24.04	USN, Ubuntu Pro (esm-infra & esm-apps)	是	是
Windows Server	2016	MSKB	否	是

操作系统	版本	供应商安全公告	无代理扫描支持	基于代理的扫描支持
Windows Server	2019	MSKB	否	是
Windows Server	2022	MSKB	否	是
Windows Server	2025	MSKB	否	是
macOS（Mojave）	10.14	APPLE-SA	否	是
macOS（Catalina）	10.15	APPLE-SA	否	是
macOS（Big Sur）	11	APPLE-SA	否	是
macOS（Monterey）	12	APPLE-SA	否	是
macOS（Ventura）	13	APPLE-SA	否	是
macOS（Sonoma）	14	APPLE-SA	否	是
macOS（Sequoia）	15	APPLE-SA	否	是

支持的操作系统：使用 Amazon Inspector 执行 Amazon ECR 扫描

下表列出了 Amazon Inspector 支持用于扫描 Amazon ECR 存储库中容器映像的操作系统。它还为每个操作系统指定了供应商安全公告。

操作系统	版本	供应商安全公告
AlmaLinux	8	Errata CVE

操作系统	版本	供应商安全公告
AlmaLinux	9	Errata CVE
AlmaLinux	10	Errata CVE
Alpine Linux (Alpine)	3.20	Errata CVE
Alpine Linux (Alpine)	3.21	Errata CVE
Alpine Linux (Alpine)	3.22	Errata CVE
Alpine Linux (Alpine)	3.23	Errata CVE
Amazon Linux (AL2)	AL2	CVE
Amazon Linux 2023 (AL2023)	AL2023	CVE
BusyBox	–	MITRE CVE
Chainguard	–	Errata CVE
Debian Server (Bullseye)	11	DSA CVE
Debian Server (Bookworm)	12	DSA CVE
Debian Server (Trixie)	13	DSA CVE
Echo	2	Errata CVE
Fedora	42	Errata CVE
Minimus	–	Errata CVE
OpenSUSE Leap	15.6	Errata CVE
Oracle Linux (Oracle)	8	Errata CVE
Oracle Linux (Oracle)	9	Errata CVE
Oracle Linux (Oracle)	10	Errata CVE

操作系统	版本	供应商安全公告
Photon OS	4	Errata CVE
Photon OS	5	Errata CVE
Red Hat Enterprise Linux (RHEL)	8	RHEL VEX CVE
Red Hat Enterprise Linux (RHEL)	9	RHEL VEX CVE
Red Hat Enterprise Linux (RHEL)	10	RHEL VEX CVE
Rocky Linux	8	Errata CVE
Rocky Linux	9	Errata CVE
Rocky Linux	10	Errata CVE
SUSE Linux Enterprise Server (SLES)	15.7	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)
Ubuntu (Jammy)	22.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Ubuntu (Noble Numbat)	24.04	USN, Ubuntu Pro (esm-infra & esm-apps)
Wolfi	–	Errata CVE

支持的操作系统：CIS 扫描

下表列出了 Amazon Inspector 支持用于 CIS 扫描的操作系统。它还为每个操作系统指定了 CIS 基准版本。

> **Note**
>
> CIS 标准适用于 x86_64 操作系统。在基于 ARM 的资源上，某些检查可能无法评估或返回无效的修补说明。

操作系统	版本	CIS 基准版本
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
Rocky Linux	8	2.0.0
Rocky Linux	9	1.0.0
SUSE Linux Enterprise Server	15	2.0.1
Ubuntu (Bionic)	18.04	2.2.0
Ubuntu (Focal)	20.04	3.0.0
Ubuntu (Jammy)	22.04	2.0.0
Ubuntu（Noble Numbat）	24.04	1.0.0
Windows Server	2016	3.0.0
Windows Server	2019	4.0.0

操作系统	版本	CIS 基准版本
Windows Server	2022	4.0.0
Windows Server	2025	1.0.0

支持的操作系统：Amazon Inspector Scan API

下表列出了 Amazon Inspector 扫描 API 支持的操作系统。有关更多信息，请参阅 ScanSbomAmazon Inspector V2 API 参考中的。

操作系统	版本
AlmaLinux 8	8
AlmaLinux	9
AlmaLinux	10
Alpine Linux	3.20
Alpine Linux	3.21
Alpine Linux	3.22
Alpine Linux	3.23
Amazon Linux	2
Amazon Linux	2023
Bottlerocket	–
BusyBox	1.36.0+
Chainguard	–
Debian	11
Debian	12

操作系统	版本
Debian	13
Debian Sid	–
Echo	2
Fedora	42
Fedora	43
macOS	11+
MinimOS	–
OpenSUSE	15.6
Oracle Linux	8
Oracle Linux	9
Oracle Linux	10
Photon OS	4
Photon OS	5
Red Hat Enterprise Linux	8
Red Hat Enterprise Linux	9
Red Hat Enterprise Linux	10
Rocky Linux	8
Rocky Linux	9
Rocky Linux	10
SUSE Server	15.7

操作系统	版本
Ubuntu	16.04
Ubuntu	18.04
Ubuntu	20.04
Ubuntu	22.04
Ubuntu	24.04
Ubuntu	25.10
Wolfi Linux	–

停产的操作系统

下表列出了已停产的操作系统及其停产时间。

尽管 Amazon Inspector 不为已停产的操作系统提供全面支持，但 Amazon Inspector 将继续扫描运行这些实例的亚马逊 EC2 实例和亚马逊 ECR 容器映像。作为安全最佳实践，我们建议改用支持的版本。Amazon Inspector 针对已停产的操作系统生成的调查结果应仅用于提供信息。

根据供应商政策，已停产的操作系统将不再收到补丁更新。对于已停用的操作系统，可能不会发布新的安全公告。对于标准支持期结束的操作系统，供应商可以从他们的信息源中删除现有的安全公告和检测。因此，Amazon Inspector 可以停止生成已知的调查结果 CVEs。

操作系统	版本	已停产
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
Alpine Linux (Alpine)	3.6	2019 年 5 月 1 日

操作系统	版本	已停产
Alpine Linux (Alpine)	3.7	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	2023 年 5 月 1 日
Alpine Linux (Alpine)	3.15	2023 年 11 月 1 日
Alpine Linux (Alpine)	3.16	2024 年 5 月 23 日
Alpine Linux (Alpine)	3.17	2024 年 11 月 22 日
Alpine Linux (Alpine)	3.18	2025 年 5 月 9 日
Alpine Linux (Alpine)	3.19	2025年11月1日
亚马逊 Linux (AL1)	2012 年	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 服务器（Jessie）	8	2020 年 6 月 30 日
Debian Server (Stretch)	9	2022 年 6 月 30 日
Debian 服务器 (Buster)	10	2024 年 6 月 30 日
Fedora	33	2021 年 11 月 30 日

操作系统	版本	已停产
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
Fedora	41	2025 年 11 月 19 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	2024 年 12 月 31 日
Oracle Linux (Oracle)	6	2021 年 3 月 1 日
Oracle Linux (Oracle)	7	2024 年 12 月 31 日
Photon 操作系统	2	2021 年 12 月 2 日
Photon 操作系统	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

操作系统	版本	已停产
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
SUSE Linux Enterprise Server (SLES)	15.6	2025 年 12 月 31 日
Ubuntu (Trusty)	12.04	2017 年 4 月 28 日

操作系统	版本	已停产
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	2023 年 7 月 20 日
Ubuntu (Lunar Lobster)	23.04	2024 年 1 月 25 日
Ubuntu (Mantic Minotaur)	23.10	2024 年 7 月 11 日
Ubuntu (Oracular Oriole)	24.10	2025 年 7 月 10 日
Ubuntu (Plucky Puffin)	25.04	2026 年 1 月 15 日
Windows Server	2012 年	2023 年 10 月 10 日
Windows Server	2012 R2	2023 年 10 月 10 日

支持的编程语言

本节列出了 Amazon Inspector 支持的编程语言。

支持的编程语言：Amazon EC2 无代理扫描

在对符合条件的 Amazon EC2 实例执行无代理扫描时，Amazon Inspector 目前支持以下编程语言。有关更多信息，请参阅无代理扫描。

> **Note**
>
> Amazon Inspector 不会在 Go 和 Rust 中扫描工具链漏洞。用于构建应用程序的编程语言编译器的版本引入了这些漏洞。

- C#

- Go
- Java
- JavaScript
- PHP
- Python
- Ruby
- Rust

支持的编程语言：Amazon EC2 深度检查

在对 Amazon EC2 Linux 实例执行深度检查扫描时，Amazon Inspector 目前支持以下编程语言。有关更多信息，请参阅基于 Linux 的 Amazon EC2 实例的 Amazon Inspector 深度检查。

- Java（.ear、.jar、.par 和 .war 存档格式）
- JavaScript
- Python

Amazon Inspector 使用 Systems Manager Distributor 部署对 Amazon EC2 实例进行深度检查的插件。

> **Note**
>
> Bottlerocket 操作系统不支持深度检查。

要执行深度检查扫描，Systems Manager Distributor 和 Amazon Inspector 必须支持您的 Amazon EC2 实例的操作系统。有关 Systems Manager Distributor 支持的操作系统的信息，请参阅《Systems Manager 用户指南》中的支持的软件包平台和架构。

支持的编程语言：Amazon ECR 扫描

Amazon Inspector 在扫描 Amazon ECR 存储库中的容器映像时目前支持以下编程语言：

> **Note**
>
> Amazon Inspector 不会在 Rust 中扫描工具链漏洞。用于构建应用程序的编程语言编译器的版本引入了这些漏洞。对于使用Chainguard库的Python应用程序，Amazon Inspector 会识别向后移植的安全补丁并将其排除在调查结果之外。

- C#
- Go
- Go 工具链
- Java
- Java JDK
- JavaScript
- PHP
- Python（包括Chainguard图书馆）
- Ruby
- Rust

支持的运行时

本节列出了 Amazon Inspector 支持的运行时系统。

支持的运行时系统：Amazon Inspector Lambda 标准扫描

对于在扫描 Lambda 函数以查找第三方软件包中是否存在漏洞时可以使用的编程语言，Amazon Inspector Lambda 标准扫描目前支持以下运行时系统：

> **Note**
>
> Amazon Inspector 不会在 Rust 中扫描工具链漏洞。用于构建应用程序的编程语言编译器的版本引入了这些漏洞。

- Go
  - go1.x

- Java
  - java8
  - java8.al2
  - java11
  - java17
  - java21
- .NET
  - .NET 6
  - .NET 8
  - .NET 10
- Node.js
  - nodejs12.x
  - nodejs14.x
  - nodejs16.x
  - nodejs18.x
  - nodejs20.x
  - nodejs22.x
  - nodejs24.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

支持的运行时系统：Amazon Inspector Lambda 代码扫描

对于在扫描 Lambda 函数以查找代码中是否存在漏洞时可以使用的编程语言，Amazon Inspector Lambda 代码扫描目前支持以下运行时系统：

- 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

停用 Amazon Inspector

可使用 Amazon Inspector 控制台或使用 Amazon Inspector API 停用 Amazon Inspector。如果您停用某个账户的所有扫描类型，则该账户的 Amazon Inspector 将自动停用。

如果您停用某个账户的 Amazon Inspector，该账户的所有扫描类型都将被停用。此外，该账户的所有 Amazon Inspector 扫描设置都将被删除，包括筛选条件、禁止规则以及调查发现。

当你停用 Amazon Inspector 亚马逊 EC2 扫描时，Amazon Inspector 会删除以下 SSM 关联：

- `InspectorDistributor-do-not-delete`
- `InspectorInventoryCollection-do-not-delete`
- `InvokeInspectorSsmPlugin-do-not-delete`。此外，通过此关联安装的 Amazon Inspector SSM 插件将从您的所有 Windows 主机上移除。有关更多信息，请参阅 扫描 Windows EC2 实例。

> **Note**
>
> 停用 Amazon Inspector 后，您就不会再产生服务费。不过，您随时可以重新激活 Amazon Inspector。

有关如何停用不同资源的扫描类型的信息，请参阅停用扫描类型。

先决条件

根据账户类型，请考虑以下事项：

- 如果您的账户是一个独立的 Amazon Inspector 账户，则可以随时停用 Amazon Inspector。
- 如果您是多账户环境中的成员账户，则无法停用 Amazon Inspector。您必须联系贵组织的委派管理员来停用 Amazon Inspector。
- 如果您是组织的委派管理员，则必须先解除所有成员账户的关联，然后才能停用 Amazon Inspector。
- 如果您的账户启用的 Amazon Inspector 由 AWS Organizations 策略管理，则无法通过亚马逊 Inspector 控制台或 API 停用策略管理的扫描类型。要停用 Amazon Inspector 扫描类型，您必须修改组织政策，使其通过 AWS Organizations 控制台或 API 明确禁用它们。您可以通过 Amazon Inspector 控制台或 API 停用不受组织策略管理的扫描类型。

> **Note**
>
> 当您以委派管理员身份停用 Amazon Inspector 时，将会停用贵组织的自动激活特征。

停用由组织政策管理的 Amazon Inspector

如果通过 AWS Organizations 策略在您的账户中启用了 Amazon Inspector，则必须使用 AWS Organizations 控制台或 API 禁用 Inspector。成员账户和委托管理员无法通过 Amazon Inspector 控制台或 API 禁用策略管理的扫描类型。

要为政策管理的账户停用 Amazon Inspector，请执行以下操作：

停用策略管理的 Amazon Inspector 启用

1. 登录 AWS Organizations 管理账户或策略管理员账户。
2. 修改组织政策，在要禁用 Inspector 的区域中将扫描类型明确设置为禁用。您必须更新策略内容，为要停用的扫描类型指定禁用的区域。
3. AWS Organizations 将自动应用政策变更，Amazon Inspector 将在受影响账户中禁用指定的扫描类型。

有关修改或分离组织政策的详细说明，请参阅 Amazon Inspector 政策 AWS Organizations 文档。

> **Note**
>
> 当您将组织政策与账户分离时，这些账户会保留其当前的 Amazon Inspector 设置（根据上次应用的策略启用或禁用）。这些账户不再受该政策的管理，然后可以独立或通过授权的管理员来管理他们的 Amazon Inspector 设置。

停用 Amazon Inspector

> **Note**
>
> 在停用 Amazon Inspector 之前，请考虑先导出调查发现。

Console

要停用 Amazon Inspector

1. 使用您的凭证登录，然后在 https://console.aws.amazon.com/inspector/v2/ home 中打开 Amazon Inspector 控制台。
2. 使用页面右上角的 AWS 区域 选择器，选择要停用 Amazon Inspector 的区域。
3. 在导航窗格中，选择常规设置。
4. 选择停用 Inspector。
5. 出现确认提示时，在文本框中输入停用，然后选择停用 Inspector。
6. （推荐）在您要停用 Amazon Inspector 的每个区域中重复这些步骤。

API

运行“禁用 API”操作。在请求中，提供 IDs 您要停EC2，ECR，LAMBDA用的帐户，以及resourceTypes要停用所有扫描的帐户，这将停用该帐户。

Amazon Inspector 配额

本节列出了每个 AWS 区域的 Amazon Inspector 配额。

资源	默认值	评论
成员账户	10000	与 Amazon Inspector 委派管理员账户关联的成员账户的最大数量。此限制基于 AWS Organizations 的配额。
抑制规则	500	每个区域每个 AWS 账户可保存的最大抑制规则数量。您无法请求提高配额。
Amazon EC2 网络调查发现	10000	每个 AWS 账户的最大 Amazon EC2 网络调查发现数量。您无法请求提高配额。
CIS 扫描配置	500	CIS 扫描配置的最大数量。您无法请求提高配额。

有关与 Amazon Inspector Classic 相关的配额列表，请参阅《AWS 一般参考》中的 Amazon Inspector Classic 服务配额。有关与 AWS Organizations 相关的配额列表，请参阅《AWS 一般参考》中的 AWS Organizations 服务配额。

区域和端点

本主题包含的表格显示了 Amazon Inspector 和 Amazon Inspector Scan 的端点。它还包括显示哪些 AWS 区域 支持 Amazon Inspector 功能的表格。要查看 Amazon Inspector 的可用 AWS 区域 位置，请参阅中的亚马逊 Inspector 终端节点和配额Amazon Web Services 一般参考。

Amazon Inspector 的服务端点

下表显示了 Amazon Inspector 的服务端点。Amazon Inspector 端点的命名规范是 `inspector2.Region.amazonaws.com`。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	inspector2.us-east-2.amazonaws.com	HTTPS
		inspector2-fips.us-east-2.amazonaws.com	HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	inspector2.us-east-1.amazonaws.com	HTTPS
		inspector2-fips.us-east-1.amazonaws.com	HTTPS
美国西部（北加利福尼亚）	us-west-1	inspector2.us-west-1.amazonaws.com	HTTPS
		inspector2-fips.us-west-1.amazonaws.com	HTTPS
美国西部（俄勒冈州）	us-west-2	inspector2.us-west-2.amazonaws.com	HTTPS
		inspector2-fips.us-west-2.amazonaws.com	HTTPS
非洲（开普敦）	af-south-1	inspector2.af-south-1.amazonaws.com	HTTPS
亚太地区（香港）	ap-east-1	inspector2.ap-east-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（海得拉巴）	ap-south-2	inspector2.ap-south-2.amazonaws.com	HTTPS
亚太地区（雅加达）	ap-southeast-3	inspector2.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（马来西亚）	ap-southeast-5	inspector2.ap-southeast-5.amazonaws.com	HTTPS
亚太地区（墨尔本）	ap-southeast-4	inspector2.ap-southeast-4.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	inspector2.ap-south-1.amazonaws.com	HTTPS
亚太地区（大阪）	ap-northeast-3	inspector2.ap-northeast-3.amazonaws.com	HTTPS
亚太地区（首尔）	ap-northeast-2	inspector2.ap-northeast-2.amazonaws.com	HTTPS
亚太地区（新加坡）	ap-southeast-1	inspector2.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	inspector2.ap-southeast-2.amazonaws.com	HTTPS
亚太地区（泰国）	ap-southeast-7	inspector2.ap-southeast-7.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（东京）	ap-northeast-1	inspector2.ap-northeast-1.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	inspector2.ca-central-1.amazonaws.com	HTTPS
加拿大西部（卡尔加里）	ca-west-1	inspector2.ca-west-1.amazonaws.com	HTTPS
欧洲地区（法兰克福）	eu-central-1	inspector2.eu-central-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	inspector2.eu-west-1.amazonaws.com	HTTPS
欧洲地区（伦敦）	eu-west-2	inspector2.eu-west-2.amazonaws.com	HTTPS
欧洲地区（米兰）	eu-south-1	inspector2.eu-south-1.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	inspector2.eu-west-3.amazonaws.com	HTTPS
欧洲（西班牙）	eu-south-2	inspector2.eu-south-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	inspector2.eu-north-1.amazonaws.com	HTTPS
欧洲（苏黎世）	eu-central-2	inspector2.eu-central-2.amazonaws.com	HTTPS

区域名称	区域	端点	协议
以色列（特拉维夫）	il-central-1	inspector2.il-central-1.amazonaws.com	HTTPS
墨西哥（中部）	mx-central-1	inspector2.mx-central-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	inspector2.me-south-1.amazonaws.com	HTTPS
中东（阿联酋）	me-central-1	inspector2.me-central-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	inspector2.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	inspector2.us-gov-east-1.amazonaws.com	HTTPS
		inspector2-fips.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	inspector2.us-gov-west-1.amazonaws.com	HTTPS
		inspector2-fips.us-gov-west-1.amazonaws.com	HTTPS

Amazon Inspector Scan API 的端点

下表显示了在调用 Amazon Inspector Scan API 时可以使用的区域端点。使用 API 时，您必须提供终端节点及其与您当前已通过身份验证的 AWS 区域对应的区域。

Amazon Inspector 扫描端点的命名约定是 `inspector-scan.`*`region`*`.amazonaws.com`。例如，如果您在 `us-west-2` 中进行了身份验证，则将使用端点 `inspector-scan.us-west-2.amazonaws.com` 来调用 `inspector-scan` API。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	inspector-scan.us-east-2.amazonaws.com inspector-scan-fips.us-east-2.amazonaws.com	HTTPS HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	inspector-scan.us-east-1.amazonaws.com inspector-scan-fips.us-east-1.amazonaws.com	HTTPS HTTPS
美国西部（北加利福尼亚）	us-west-1	inspector-scan.us-west-1.amazonaws.com inspector-scan-fips.us-west-1.amazonaws.com	HTTPS HTTPS
美国西部（俄勒冈州）	us-west-2	inspector-scan.us-west-2.amazonaws.com inspector-scan-fips.us-west-2.amazonaws.com	HTTPS HTTPS
非洲（开普敦）	af-south-1	inspector-scan.af-south-1.amazonaws.com	HTTPS
亚太地区（香港）	ap-east-1	inspector-scan.ap-east-1.amazonaws.com	HTTPS
亚太地区（海得拉巴）	ap-south-2	inspector-scan.ap-south-2.amazonaws.com	HTTPS
亚太地区（雅加达）	ap-southeast-3	inspector-scan.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（马来西亚）	ap-southeast-5	inspector-scan.ap-southeast-5.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（墨尔本）	ap-southeast-4	inspector-scan.ap-southeast-4.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	inspector-scan.ap-south-1.amazonaws.com	HTTPS
亚太地区（大阪）	ap-northeast-3	inspector-scan.ap-northeast-3.amazonaws.com	HTTPS
亚太地区（首尔）	ap-northeast-2	inspector-scan.ap-northeast-2.amazonaws.com	HTTPS
亚太地区（新加坡）	ap-southeast-1	inspector-scan.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	inspector-scan.ap-southeast-2.amazonaws.com	HTTPS
亚太地区（泰国）	ap-southeast-7	inspector-scan.ap-southeast-7.amazonaws.com	HTTPS
亚太地区（东京）	ap-northeast-1	inspector-scan.ap-northeast-1.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	inspector-scan.ca-central-1.amazonaws.com	HTTPS
加拿大西部（卡尔加里）	ca-west-1	inspector-scan.ca-west-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
欧洲地区（法兰克福）	eu-central-1	inspector-scan.eu-central-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	inspector-scan.eu-west-1.amazonaws.com	HTTPS
欧洲地区（伦敦）	eu-west-2	inspector-scan.eu-west-2.amazonaws.com	HTTPS
欧洲地区（米兰）	eu-south-1	inspector-scan.eu-south-1.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	inspector-scan.eu-west-3.amazonaws.com	HTTPS
欧洲（西班牙）	eu-south-2	inspector-scan.eu-south-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	inspector-scan.eu-north-1.amazonaws.com	HTTPS
欧洲（苏黎世）	eu-central-2	inspector-scan.eu-central-2.amazonaws.com	HTTPS
以色列（特拉维夫）	il-central-1	inspector-scan.il-central-1.amazonaws.com	HTTPS
墨西哥（中部）	mx-central-1	inspector-scan.mx-central-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	inspector-scan.me-south-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
中东（阿联酋）	me-central-1	inspector-scan.me-central-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	inspector-scan.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	inspector-scan.us-gov-east-1.amazonaws.com inspector-scan-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	inspector-scan.us-gov-west-1.amazonaws.com inspector-scan-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS

特定于区域的特征可用性

本节介绍按 AWS 区域划分的 Amazon Inspector 可用特征。

适用于 Amazon EC2 的无代理 EC2 扫描功能可用的区域

下表显示了目前可用于 Amazon EC2 的无代理扫描 AWS 区域 的地方。

区域名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国东部（俄亥俄州）	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1

区域名称	区域代码
亚太地区（香港）	ap-east-1
亚太地区（东京）	ap-northeast-1
亚太地区 (首尔)	ap-northeast-2
亚太地区（大阪）	ap-northeast-3
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（墨尔本）	ap-southeast-4
亚太地区（马来西亚）	ap-southeast-5
亚太地区（泰国）	ap-southeast-7
加拿大（中部）	ca-central-1
加拿大西部（卡尔加里）	ca-west-1
欧洲地区（斯德哥尔摩）	eu-north-1
欧洲地区（法兰克福）	eu-central-1
欧洲（苏黎世）	eu-central-2
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3

区域名称	区域代码
欧洲地区（米兰）	eu-south-1
欧洲（西班牙）	eu-south-2
以色列（特拉维夫）	il-central-1
中东（阿联酋）：	me-central-1
中东（巴林）	me-south-1
墨西哥（中部）	mx-central-1
南美洲（圣保罗）	sa-east-1
AWS GovCloud （美国东部）	us-gov-east-1
AWS GovCloud （美国西部）	us-gov-west-1

Lambda 代码扫描区域

下表显示了当前可 AWS 区域 用 Lambda 代码扫描的地方。

区域名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国东部（俄亥俄州）	us-east-2
亚太地区（悉尼）	ap-southeast-2
亚太地区（东京）	ap-northeast-1
欧洲地区（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2

区域名称	区域代码
欧洲地区（斯德哥尔摩）	eu-north-1
亚太地区（新加坡）	ap-southeast-1

> **Important**
>
> 如果您尝试在无法进行 Lambda 代码扫描的情况下使用 Amazon Inspector Enable API 启用 Lamb AWS 区域 da 代码扫描，则会收到以下拒绝访问错误：
>
> ```
> An error occurred (AccessDeniedException) when calling the Enable operation:
>  Lambda code scanning is not supported in unsupported-AWS ##
> ```

Amazon Inspector 代码安全性区域

下表显示了 Amazon Inspector 代码安全目前 AWS 区域 在哪些地方可用。

区域名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国东部（俄亥俄州）	us-east-2
亚太地区（悉尼）	ap-southeast-2
亚太地区（东京）	ap-northeast-1
欧洲地区（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（斯德哥尔摩）	eu-north-1

区域名称	区域代码
亚太地区（新加坡）	ap-southeast-1

AWS GovCloud (US) 区域

有关最新信息，请参阅 AWS GovCloud (US) 用户指南中的 Amazon Inspector。

文档历史记录

下表列出了从 2021 年 11 月开始的每个版本的《Amazon Inspector 用户指南》中的重要更改。要接收有关文档更新的通知，您可以订阅 RSS 源。

Amazon Inspector 产品更新

变更	说明	日期
Amazon Inspector SBOM 生成器的更新	Amazon Inspector 意识到一种情况，即亚马逊 Inspector SBOM 生成器可能会生成 CVE-2026-25679、CVE-2026-27142 和 CVE-2026-27139 的漏洞发现。经证实，Amazon Inspector SBOM 生成器没有受到这些漏洞的影响。此漏洞可以通过将 Amazon Inspector SBOM 生成器版本升级到 1.11.2 或更高版本来解决。	2026年3月11日
Amazon Inspector SBOM 生成器的更新	Amazon Inspector 知道存在一种情况，即亚马逊 Inspector SBOM 生成器可能会生成漏洞调查结果。CVE-2025-15558 经证实，Amazon Inspector SBOM 生成器未受到 CVE-2025-15558 的影响。此漏洞可以通过将 Amazon Inspector SBOM 生成器版本升级到 1.11.1 或更高版本来解决。	2026年3月5日
Amazon Inspector SBOM 生成器的更新	Amazon Inspector 知道存在一种情况，即亚马逊 Inspector	2026 年 3 月 2 日

	SBOM 生成器可能会生成漏洞调查结果。CVE-2025-68121 经证实，Amazon Inspector SBOM 生成器未受到 CVE-2025-68121 的影响。此漏洞可以通过将 Amazon Inspector SBOM 生成器版本升级到 1.11.0 或更高版本来解决。	
新托管式策略	Amazon Inspector 发布了一项新的托管策略AmazonInspector2ManagedTelemetryPolicy ，该策略授予亚马逊 Inspector 遥测操作的权限，允许该服务收集和传输包裹库存数据以进行漏洞扫描。有关信息，请参阅 Amazon Inspector 对 AWS 托管策略的更新。	2026 年 2 月 5 日
更新了政策	Amazon Inspector 为名为 AmazonInspector2ServiceRolePolicy 的服务相关角色添加了新权限。Amazon Inspector 添加了一项新权限，允许亚马逊 Inspector 描述用于网络可访问性分析的防火墙元数据。此外，Amazon Inspector 还添加了额外的资源范围界定，以允许 Amazon Inspector 创建、更新和启动与 SSM 文档的 SSM 关联。AWS-ConfigureAWSPackage 有关更多信息，请参阅 Amazon Inspector 的服务相关角色权限。	2026 年 2 月 3 日

Amazon Inspector SSM 插件和亚马逊 Inspector SBOM 生成器的更新	Amazon Inspector 意识到 Amazon Inspector SSM 插件和 Amazon Inspector SBOM 生成器可能会生成漏洞调查结果的情况。CVE-2025-61728, CVE-2025-61730, and CVE-2025-61726 这些漏洞可以通过将 Amazon Inspector SSM 插件版本升级到 1.0.2327.0 或 Amazon Inspector SBOM Generator 1.10.1 或更高版本来解决。	2026 年 1 月 29 日
Amazon Inspector SSM 插件和亚马逊 Inspector SBOM 生成器的更新	Amazon Inspector 意识到 Amazon Inspector SSM 插件和 Amazon Inspector SBOM 生成器可能会生成漏洞调查结果的情况。CVE-2025-61729 已确认这些应用程序不受此 CVE 的影响。我们目前正在努力改进以解决此检测问题。同时，客户可以放心地忽略或抑制此漏洞。	2025年12月3日
Amazon Inspector SBOM 生成器更新	Amazon Inspector 意识到 Amazon Inspector SBOM 生成器可能会为CVE-2025-47914 和CVE-2025-58181 生成漏洞发现的情况。经证实，Amazon Inspector SBOM 生成器没有受到这些CVEs因素的影响。我们目前正在努力改进以解决这些检测问题。同时，客户可以放心地忽略或抑制这些漏洞。	2025 年 11 月 20 日

新特征	Amazon Inspector 现在支持跨组织账户集中启用和管理的 AWS Organizations 策略。组织政策允许您在整个组织中自动启用 Amazon Inspector 扫描类型，并防止未经授权的修改。有关更多信息，请参阅入门教程和管理多个账户。	2025 年 11 月 19 日
Amazon Inspector SBOM 生成器更新	Amazon Inspector 意识到一种情况，即亚马逊检查器 SBOM 生成器可能会生成漏洞调查结果。CVE-2025-47913 经证实，Amazon Inspector SBOM 生成器不受此 CVE 的影响，并已部署更新来解决此检测问题。	2025 年 11 月 14 日
更新策略	Amazon Inspector 为托管策略添加了新的权限 AmazonInspector2FullAccess_v2 ，并且AmazonInspector2ReadOnlyAccess 。这些权限允许查看通过策略建立的 Amazon Inspector 组织 AWS Organizations 策略和委托配置。有关更多信息，请参阅 Amazon Inspector 的 AWS 托管式策略。	2025 年 11 月 14 日
Amazon Inspector SBOM 生成器更新	Amazon Inspector 更新了 Amazon Inspector SBOM 生成器版本。有关更多信息，请参阅先前版本的 Amazon Inspector SBOM 生成器。	2025 年 11 月 11 日

更新了政策	Amazon Inspector 为名为 AmazonInspector2ServiceRolePolicy 的服务相关角色添加了新权限。这些权限允许 Amazon Inspector AWS Organizations 策略强制启用和禁用 Amazon Inspector 。有关更多信息，请参阅 Amazon Inspector 的服务相关角色权限。	2025 年 11 月 10 日
Amazon Inspector SBOM 生成器更新	Amazon Inspector 注意到这样一种情况，即 Amazon Inspector SBOM 生成器可能会为 CVE-2025-58188 和 CVE-2025-61725 生成漏洞调查发现。经证实，Amazon Inspector SBOM 生成器不受这些影响 CVEs，Amazon Inspector 更新了 Amazon Inspector SBOM 生成器版本。有关更多信息，请参阅先前版本的 Amazon Inspector SBOM 生成器。	2025 年 11 月 4 日
插件更新	Amazon Inspector 获悉存在这样一种情况，即 Amazon Inspector SSM 插件可能会针对 CVE-2025-58188 和 CVE-2025-61725 生成漏洞调查发现。已确认Amazon Inspector SSM插件不受这些因素的影响 CVEs，并已部署更新来解决此检测问题。	2025 年 11 月 3 日

插件更新	Amazon Inspector 获悉存在这样一种情况，即 Amazon Inspector SSM 插件可能会针对 CVE-2025-47907 生成漏洞调查发现。已确认Amazon Inspector SSM插件不受这些因素的影响 CVEs，并已部署更新来解决此检测问题。	2025 年 8 月 8 日
新策略	Amazon Inspector 增加了一个新的托管式策略，提供对 Amazon Inspector 的完全访问权限以及对其他相关服务的访问权限。有关更多信息，请参阅 Amazon Inspector 的AWS 托管式策略。	2025 年 7 月 3 日
更新了功能	Amazon Inspector 现在已在新的 AWS 区域上市。有关更多信息，请参阅区域和端点。	2025 年 7 月 1 日
更新了功能	Amazon Inspector 更新了已关闭调查发现的保留期。只要关联资源被删除、终止或不再符合扫描资格，Amazon Inspector 就会在 3 天后移除调查发现。有关更多信息，请参阅了解 Amazon Inspector 调查发现。	2025 年 6 月 25 日

更新了功能	Amazon Inspector 更新了其用于 Amazon EC2 扫描和 Amazon ECR 扫描的受支持操作系统。Amazon EC2 扫描现在支持 Fedora 版本 42 和 Ubuntu 版本 25.04。Amazon ECR 扫描现在支持 Alpine 版本 3.22、Fedora 版本 42 和 Ubuntu 版本 25.04。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2025 年 6 月 18 日
新特征	Amazon Inspector 现在会扫描第一方应用程序源代码、第三方应用程序依赖关系以及基础设施即代码中的漏洞。有关更多信息，请参阅 Amazon Inspector 代码安全性。	2025 年 6 月 17 日
插件更新	Amazon Inspector 注意到这样一种情况，即 Amazon Inspector SSM 插件可能会为 CVE-2025-0913 和 CVE-2025-4673 生成漏洞调查发现。已确认Amazon Inspector SSM插件不受这些因素的影响 CVEs，并已部署更新来解决此检测问题。	2025 年 6 月 13 日
新特征	Amazon Inspector 现在可以显示活跃容器映像以及容器映像上次在集群上使用的时间。有关更多信息，请参阅将容器映像映射到正在运行的容器。	2025 年 5 月 16 日

支持的操作系统更新	Amazon Inspector 增加了对 BusyBox 的支持。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2025 年 5 月 13 日
更新了政策	Amazon Inspector 为名为 AmazonInspector2ServiceRolePolicy 的服务相关角色添加了新权限。此权限允许您描述 IP 地址和互联网网关。有关更多信息，请参阅 Amazon Inspector 的AWS 托管式策略。	2025 年 4 月 29 日
插件更新	Amazon Inspector 获悉存在这样一种情况，即 Amazon Inspector SSM 插件可能会针对 CVE-2025-22871 生成漏洞调查发现。已确认Amazon Inspector SSM插件不受这些因素的影响 CVEs，并已部署更新来解决此检测问题。	2025 年 4 月 21 日
插件更新	Amazon Inspector 获悉存在这样一种情况，即 Amazon Inspector SSM 插件可能会针对 CVE-2020-8911 、CVE-2020-8912 和 CVE-2024-45337 生成漏洞调查发现。经证实，Amazon Inspector 没有受到这些问题的影响， CVEs 并已部署更新来解决此检测问题。	2025 年 4 月 18 日

“Amazon Inspector SBOM 生成器”章节更新	Amazon Inspector 会更新 Amazon Inspector SBOM 生成器版本。有关更多信息，请参阅先前版本的 Amazon Inspector SBOM 生成器。	2025 年 4 月 16 日
“Amazon Inspector SBOM 生成器”章节更新	Amazon Inspector 在“Amazon Inspector SBOM 生成器”章节中新增了主题。本主题介绍了 Sbomgen 如何在软件物料清单中跟踪许可信息。有关更多信息，请参阅 Amazon Inspector SBOM 生成器许可证收集。	2025 年 4 月 16 日
托管式策略的更新	Amazon Inspector 添加了相关权限，允许对 Amazon ECS 和 Amazon EKS 操作进行只读访问。有关更多信息，请参阅 Amazon Inspector 的服务相关角色权限。	2025 年 3 月 25 日
支持的操作系统更新	Amazon Inspector 不再支持 SUSE Linux Enterprise Server 12.5 作为 Amazon EC2 和 Amazon ECR 扫描的一部分。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2025 年 3 月 21 日
支持的操作系统更新	Amazon Inspector 为 Amazon ECR 扫描增加了对 Chainguard 和 Wolfi 的支持。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2025 年 3 月 21 日

目录更新	Amazon Inspector 添加了关于标记 Amazon Inspector 资源的章节。有关更多信息，请参阅标记 Amazon Inspector 资源。	2025 年 2 月 25 日
目录更新	Amazon Inspector 在“Amazon Inspector SBOM 生成器”章节中新增了主题。有关更多信息，请参阅 Amazon Inspector SBOM 生成器全面操作系统收集。	2025 年 1 月 28 日
更新了功能	Amazon Inspector 将 nodejs202.x 和 python3.13 添加到其支持的 Lambda 标准扫描运行时列表中。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2025 年 1 月 24 日
更新了功能	Amazon Inspector 从其支持的 Amazon EC2 和 Amazon ECR 操作系统列表中移除了 Oracle Linux (Oracle) 7 和 SUSE Linux Enterprise Server (SLES) 15.5。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2024 年 12 月 31 日
更新了功能	Amazon Inspector 将 Ubuntu 24.10 添加到其支持的 Amazon EC2 和 Amazon ECR 操作系统列表中。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2024 年 12 月 12 日

目录更新	Amazon Inspector 在“Amazon Inspector SBOM 生成器”章节中新增了主题。有关更多信息，请参阅 Amazon Inspector SBOM 生成器。	2024 年 12 月 9 日
更新了功能	Amazon Inspector 更新 `amazon:inspector:sbom_generator` 表以添加和移除命名空间。有关更多信息，请参阅将 CycloneDX 命名空间与 Amazon Inspector 结合使用。	2024 年 12 月 9 日
更新了功能	Amazon Inspector 更新了其 CI/CD 集成功能，以支持使用进行扫描操作。 CodePipeline有关更多信息，请参阅将 Amazon Inspector 的扫描操作与一起使用 CodePipeline。	2024 年 11 月 26 日
目录更新	Amazon Inspector 重新组织了目录，加入了 Amazon Inspector SBOM 生成器的章节。有关更多信息，请参阅 Amazon Inspector SBOM 生成器。	2024 年 11 月 22 日
更新了功能	Amazon Inspector 将 Fedora 39 从其支持的 Amazon EC2 和 Amazon ECR 操作系统列表中移除。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2024 年 11 月 22 日

更新了功能	Amazon Inspector 将 Alpine 3.17 从其支持的 Amazon ECR 操作系统列表中移除。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2024 年 11 月 22 日
更新了功能	Amazon Inspector 将 Sbomgen 版本添加到 Amazon Inspector SBOM 生成器的先前版本中。	2024 年 11 月 19 日
更新了功能	Amazon Inspector 添加 AL2 为支持的运行时。有关更多信息，请参阅 Amazon Inspector 支持的操作系统和编程语言。	2024 年 8 月 26 日
更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新语句支持 Amazon Inspector 在 AWS Lambda中返回函数标签。	2024 年 7 月 31 日
更新了功能	Amazon Inspector 发布了新的安全控件。有关更多信息，请参阅《AWS Security Hub CSPM 用户指南》中的 Amazon Inspector 控件。	2024 年 7 月 11 日
更新了功能	Amazon Inspector SBOM 生成器现在会扫描 Dockerfile 和 Docker 容器映像，以查找是否存在可能引入安全漏洞的错误配置。有关更多信息，请参阅 Amazon Inspector Dockerfile 检查。	2024 年 6 月 10 日

更新了功能	Amazon Inspector 更新了其 CI/CD 集成功能以支持 CodeCatalyst 操作，因此您可以将 Amazon Inspector 漏洞扫描添加到您的 CodeCatalyst 工作流程中。有关更多信息，请参阅使用 CodeCatalyst 操作。	2024 年 6 月 7 日
更新了功能	Amazon Inspector 包括一个可用于以 CSV 文件格式下载 CIS 扫描结果的选项。有关更多信息，请参阅 Amazon EC2 实例的 Center for Internet Security（CIS）扫描中的查看和下载 CIS 扫描结果。	2024 年 5 月 3 日
更新了功能	Amazon Inspector 更新了其 CI/CD 集成特征以支持 GitHub Actions，因此您可以将 Amazon Inspector 漏洞扫描添加到 GitHub 工作流中。有关更多信息，请参阅结合 Amazon Inspector 使用 GitHub Actions。	2024 年 4 月 29 日
更新了功能	Amazon Inspector 更新了托管式策略 `AmazonInspector2FullAccess`，因此它会创建服务相关角色 `AWSServiceRoleForAmazonInspector2Agentless`。如此，用户就能在启用 Amazon Inspector 时执行基于代理的扫描和无代理扫描。	2024 年 4 月 24 日

更新了功能	Amazon Inspector 会将已关闭调查发现的保留期从 30 天更新为 7 天。有关更多信息，请参阅了解 Amazon Inspector 中的调查发现。	2024 年 2 月 12 日
更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新语句支持 Amazon Inspector 对实例启动 CIS 扫描。	2024 年 1 月 23 日
新策略	Amazon Inspector 添加了一项新的策略，即 AmazonInspector2ManagedCisPolicy 策略，您可以将其用作实例配置文件的一部分，以便对实例进行 CIS 扫描。	2024 年 1 月 23 日
新特征	现在，当您拉取容器映像时，Amazon Inspector 将刷新容器映像的 ECR 重新扫描持续时间。要根据推送或拉取日期更改重新扫描持续时间，请参阅配置 ECR 重新扫描持续时间。	2024 年 1 月 23 日
新特征	Amazon Inspector 现在可以对 EC2 实例运行 Center for Internet Security（CIS）扫描。有关更多信息，请参阅 Amazon Inspector CIS 扫描。	2024 年 1 月 23 日

新特征	Amazon Inspector 现在可以扫描您的 CI/CD 管道中的容器镜像。有关更多信息，请参阅 CI/CD 与 Amazon Inspector 的集成。	2023 年 11 月 30 日
新策略	Amazon Inspector 添加了一项新策略，允许 Amazon Inspector 对 EC2 实例中的 Amazon EBS 快照进行无代理扫描。有关该策略的更多信息，请参阅无代理扫描。	2023 年 11 月 27 日
新特征	Amazon Inspector 现在支持通过无代理扫描在没有 SSM 代理的情况下扫描支持的 Linux Amazon EC2 实例。有关更多信息，请参阅无代理扫描。	2023 年 11 月 27 日
新的支持的资源	Amazon Inspector 现在支持扫描 MacOS Amazon EC2 实例。请参阅支持的操作系统：Amazon EC2 扫描，了解支持的 MacOS 版本。	2023 年 10 月 5 日
新区域	Amazon Inspector 现已在亚太地区（雅加达）、非洲（开普敦）、亚太地区（大阪）和欧洲地区（苏黎世）发布。	2023 年 9 月 29 日
新特征	现在，您可以使用排除标签将 EC2 实例从 Amazon Inspector 扫描中排除。	2023 年 9 月 14 日

新特征	Amazon Inspector 增加了新的权限，允许 Amazon Inspector 扫描属于 Elastic Load Balancing 目标组的 Amazon EC2 实例的网络配置。	2023 年 8 月 31 日
新特征	Amazon Inspector 现在可为程序包脆弱性调查发现提供脆弱性情报详细信息。	2023 年 7 月 31 日
更新了功能	Amazon Inspector 增加了新的权限，允许只读用户为其资源导出软件材料清单 (SBOM)。	2023 年 6 月 29 日
新特征	现在，您可以导出 Amazon Inspector 正在扫描的资源的 SBOM。	2023 年 6 月 13 日
新特征	Lambda 代码扫描现已全面推出。新增功能允许您对 Lambda 代码扫描结果中发现的代码进行加密。此外，Lambda 代码扫描现在还可提供代码修复重写建议。	2023 年 6 月 13 日
更新了功能	Amazon Inspector 在 AmazonInspector2ReadOnlyAccess 策略中增加了一个新语句。新语句允许只读用户检索其账户的 Lambda 代码扫描状态和结果的详细信息。	2023 年 5 月 2 日
新特征	Amazon Inspector 增加了脆弱性数据库搜索功能，支持检查 Amazon Inspector 是否涵盖了特定的 CVE。	2023 年 5 月 1 日

更新了功能	Amazon Inspector 为该AmazonInspector2ServiceRolePolicy政策添加了新的权限，允许亚马逊检查员在您激活 Lambda 扫描时在您的账户中创建 AWS CloudTrail 与服务相关的渠道。这样，Amazon Inspector 就可以监控您账户中的 CloudTrail 事件。	2023 年 4 月 30 日
更新了功能	Amazon Inspector 在AmazonInspector2FullAccess策略中增加了一个新语句。新语句允许用户从 Lambda 代码扫描中检索代码脆弱性调查发现的详细信息。	2023 年 4 月 17 日
更新了功能	Amazon Inspector 在AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新语句支持 Amazon Inspector 向 Amazon EC2 Systems Manager 发送有关您为 Amazon EC2 深度检查定义的自定义路径的信息。	2023 年 4 月 17 日
新特征	Amazon Inspector 以 Amazon Inspector 深度检查的形式增加了对 Linux EC2 实例的额外支持，该检查可扫描您的实例，查找应用程序编程语言程序包中的程序包漏洞。	2023 年 4 月 17 日

更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新的语句允许 Amazon Inspector 请求对 AWS Lambda 函数中的开发者代码进行扫描，并从亚马逊 CodeGuru 安全部门接收扫描数据。此外，Amazon Inspector 还增加了审查 IAM policy 的权限。Amazon Inspector 使用这些信息扫描 Lambda 函数中是否存在代码脆弱性。	2023 年 2 月 28 日
新特征	Amazon Inspector 以 Lambda 代码扫描的形式增加了对 Lambda 函数的额外支持，这会扫描 Lambda 函数的开发者代码中是否存在安全脆弱性。	2023 年 2 月 28 日
更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新语句允许 Amazon Inspector 检索 CloudWatch 有关上次调用 AWS Lambda 函数的时间的信息。使用这些信息将扫描重点放在您环境中过去 90 天内处于活动状态的 Lambda 函数上。	2023 年 2 月 20 日

更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新语句。新语句允许 Amazon Inspector 检索有关 AWS Lambda 函数的信息。Amazon Inspector 使用这些信息扫描 Lambda 函数中是否存在安全脆弱性。	2022 年 11 月 28 日
新特征	Amazon Inspector 增加了对扫描 AWS Lambda 功能的支持。	2022 年 11 月 28 日
更新了内容	增加了将调查发现报告从 Amazon Inspector 导出到 Amazon Simple Storage Service (Amazon S3) 存储桶的程序、策略示例和提示。	2022 年 10 月 14 日
新增内容	添加了有关使用亚马逊 Inspector 控制台评估 Amazon Inspector 对您 AWS 环境的覆盖范围的信息。这些信息包括环境中各个资源的状态值说明。	2022 年 10 月 7 日
新特征	Amazon Inspector 现在提供有关如何修复程序包脆弱性的更多详细信息。调查发现详细信息增加了新字段。新字段提供了是否可以通过程序包更新获得修复的上下文信息。如果有可用的修复方法，则调查发现的建议的补救措施部分会显示您可以运行的修复命令。	2022 年 9 月 2 日

更新了功能	Amazon Inspector 在 AmazonInspector2ServiceRolePolicy 策略中增加了一个新操作。此新操作允许 Amazon Inspector 描述 SSM 关联的执行情况。Amazon Inspector 还增加了额外的资源范围，允许 Amazon Inspector 使用 `AmazonInspector2` 拥有的 SSM 文档创建、更新、删除和启动 SSM 关联。	2022 年 8 月 31 日
新特征	Amazon Inspector 现在支持对 Windows 实例进行扫描。Amazon Inspector 现在可以扫描运行支持的 Windows 操作系统的 SSM 托管实例。Windows 主机扫描由 Amazon Inspector SSM 插件执行，该插件通过 Amazon Inspector 自动创建的新 SSM 关联安装和调用。	2022 年 8 月 31 日
更新了功能	Amazon Inspector 更新了AmazonInspector2ServiceRolePolicy政策的资源范围，允许亚马逊 Inspector 收集其他 AWS 分区中的软件库存。	2022 年 8 月 12 日
更新了功能	在 AmazonInspector2ServiceRolePolicy 策略中，Amazon Inspector 重组了操作的资源范围，允许 Amazon Inspector 创建、删除和更新 SSM 关联。	2022 年 8 月 10 日

新特征	Amazon Inspector 现在支持更改 ECR 自动重新扫描持续时间设置。Amazon ECR 自动重新扫描持续时间设置决定了 Amazon Inspector 持续监控推送到存储库的映像的时间。当映像存在时间超过扫描持续时间时，Amazon Inspector 将不再扫描该映像并将关闭它的所有现有结果。所有新账户的 ECR 自动重新扫描持续时间都将自动设置为生命周期。之前创建的账户的 ECR 自动重新扫描持续时间为 30 天，但现在您可以将扫描持续时间设置为 30 天、180 天或生命周期。	2022 年 6 月 25 日
新功能	Amazon Inspector 添加了一项新的 AWS 托管策略，即AmazonInspector2ReadOnlyAccess政策，允许对亚马逊 Inspector 功能进行只读访问。	2022 年 1 月 21 日
正式发布	这是 Amazon Inspector 用户指南的第一个公开发行版。	2021 年 11 月 29 日

Amazon Inspector 安全研究

Amazon Inspector 会持续监控和识别 NPM 注册表中的恶意软件包，以保护您的应用程序免受供应链攻击。

最新更新：2026-02-06 12:00:00 世界标准时间

检测摘要

- 生命周期总计：已识别出 191,801 个恶意软件包

- 本月：发现了 147 个新的恶意软件包
- 上个月：发现了 527 个新的恶意软件包
- 本周：发现了 147 个新的恶意软件包
- 上周：发现了 96 个新的恶意软件包

最近的恶意 Package 报告（最近 10 个）

包名称	MAL-ID	检测日期
web3-sinon	MAL-2026-807	2026-02-06
web3-chain-sinon	MAL-2026-806	2026-02-06
对齐数组	MAL-2026-805	2026-02-06
面包屑服务	MAL-2026-804	2026-02-06
@sbseg-plugin/ qbo-web-app-ui	MAL-2026-802	2026-02-06
@rsgweb /utils	MAL-2026-801	2026-02-06
@rsgweb /tina	MAL-2026-800	2026-02-06
@rsgweb /rockstar-account	MAL-2026-799	2026-02-06
@rsgweb/modules-core-www-page	MAL-2026-798	2026-02-06
@rsgweb/modules-core-feedback	MAL-2026-797	2026-02-06

AWS 术语表

有关最新的 AWS 术语，请参阅 AWS 词汇表 参考中的 AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。