Red Hat Quay 3.11

Red Hat Quay 中的 Clair 的漏洞报告

Red Hat Quay 中的 Clair 的漏洞报告

Last Updated: 2024-06-19

Red Hat Quay 3.11 Red Hat Quay 中的 Clair 的漏洞报告

Red Hat Quay 中的 Clair 的漏洞报告

法律通告

摘要

Clair 入门

目录

前言

本指南中的内容概述了 Red Hat Quay 的 Clair，在独立 Red Hat Quay 和 Operator 部署上运行 Clair，以及高级 Clair 配置。

部分 I. RED HAT QUAY 上 CLAIR 的漏洞报告概述

本指南中的内容解释了 Red Hat Quay 中 Clair 的关键目的和概念。它还包含有关 Clair 发行版本和官方 Clair 容器的位置的信息。

第 1 章 CLAIR 安全扫描程序

Clair v4 (Clair)是一个开源应用，它利用静态代码分析来解析镜像内容并报告影响内容的漏洞。Clair 打包了 Red Hat Quay，并可用于独立和 Operator 部署。它可以在高度可扩展的配置中运行，其中组件可以根据企业环境单独进行扩展。

1.1. 关于 CLAIR

Clair 使用国家漏洞数据库(NVD)中的常见漏洞评分系统(CVSS)数据功能丰富的漏洞数据，这是与安全相关的信息，包括各种软件组件和系统中的已知漏洞和安全问题。使用 NVD 中的分数为 Clair 提供以下优点：

- 数据同步.Clair 可以定期将其漏洞数据库与 NVD 同步。这样可确保它具有最新的漏洞数据。
- 匹配并增强.Clair 将容器镜像中发现的漏洞的元数据和标识符与 NVD 中的数据进行比较。这个过程涉及将唯一标识符（如常见漏洞和暴露(CVE) ID）与 NVD 中的条目匹配。找到匹配项时，Clair 可以通过来自 NVD 的额外详情来增强其漏洞信息，如严重性分数、描述和引用。
- 严重性分数.NVD 为漏洞分配严重性分数，如通用漏洞评分系统(CVSS)分数，以指示与每个漏洞相关的潜在影响和风险。通过整合 NVD 的严重性分数，Clair 可以根据它检测到的漏洞的严重程度提供更多上下文。

如果 Clair 从 NVD 找到漏洞，对容器镜像中检测到的漏洞的严重性和潜在影响的详细说明和潜在影响将报告给 UI 上的用户。CVSS 增强数据提供了以下优点：

- 漏洞优先级.通过使用 CVSS 分数，用户可以根据其严重性对漏洞进行优先级排序，帮助他们首先解决最重要的问题。
- 评估风险。CVSS 分数可帮助 Clair 用户了解对其容器化应用程序造成漏洞的潜在风险。
- 通讯严重性.CVSS 分数为 Clair 用户提供了一种标准化的方法，来跨团队和机构沟通漏洞的严重性。
- 通知修复策略.CVSS 增强数据可在开发适当的补救策略时指导 Quay.io 用户。
- 合规性和报告.将 CVSS 数据集成到 Clair 生成的报告中可帮助组织展示其在解决安全漏洞方面以及符合行业标准和法规的承诺。

1.1.1. Clair 发行版本

Clair 的新版本会定期发布。构建 Clair 所需的源代码被打包为一个存档，并附加到每个发行版本。Clair 版本可在 Clair 发行版本中 找到。

发行工件还包括 **clairctl** 命令行界面工具，该工具使用开放主机从互联网获取更新器数据。

Clair 4.7.1
Clair 4.7.1 已被作为 Red Hat Quay 3.9.1 的一部分发布。进行了以下更改：

- 在这个版本中，您可以查看 Red Hat Enterprise Linux (RHEL)源中的未修补的漏洞。如果要查看未修补的漏洞，您可以将 **ignore_unpatched** 参数设置为 **false**。例如：

```
updaters:
  config:
    rhel:
      ignore_unpatched: false
```

要禁用此功能，您可以将 **ignore_unpatched** 设置为 **true**。

Clair 4.7
Clair 4.7 作为 Red Hat Quay 3.9 的一部分发布，包括对以下功能的支持：

- 对容器镜像中的 Golang 模块和 RubeGems 进行原生支持。
- 更改到 OSV.dev，作为任何编程语言软件包管理器的漏洞数据库来源。
  - 这包括 GitHub 安全公告或 PyPA 等流行源。
  - 这允许离线功能。
- 将 pyup.io 用于 Python，而 CRDA 用于 Java 被暂停。
- Clair 现在支持 Java、Node Golang、Python 和 Ruby 依赖项。

1.1.2. Clair 漏洞数据库

Clair 使用以下漏洞数据库来报告镜像中的问题：

- Ubuntu Oval 数据库
- Debian 安全跟踪器
- Red Hat Enterprise Linux (RHEL) Oval 数据库
- SUSE Oval 数据库
- Oracle Oval 数据库
- alpine SecDB 数据库
- VMware Photon OS 数据库
- Amazon Web Services (AWS) UpdateInfo
- 开源漏洞(OSV)数据库

1.1.3. Clair 支持的依赖项

Clair 支持识别和管理以下依赖项：

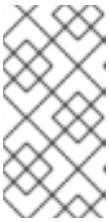
- Java
- Golang
- Python
- Ruby

这意味着，它可以分析和报告这些语言中项目依赖的第三方库和软件包。

当包含 Clair 不支持的语言的软件包的镜像被推送到您的存储库时，无法在这些软件包上执行漏洞扫描。用户不会收到不支持的依赖项或软件包的分析或安全报告。因此，应该考虑以下结果：

- 安全风险.没有扫描漏洞的依赖项或软件包可能会给您的组织带来安全风险。

- **合规问题**.如果您的组织具有特定的安全性或合规要求、未扫描或部分扫描，容器镜像可能会导致与某些法规不兼容。



注意

扫描的镜像会被索引，并创建了漏洞报告，但可能会忽略某些不支持的语言中的数据。例如，如果您的容器镜像包含 Lua 应用程序，则不会提供来自 Clair 的反馈，因为 Clair 不会检测到它。它可以检测容器镜像中使用的其他语言，并显示这些语言检测到的 CVE。因此，Clair 镜像 *会根据 Clair 支持的内容进行完全扫描*。

1.1.4. Clair 容器

红帽生态系统目录 上可以找到与 Red Hat Quay 捆绑的官方下游 Clair 容器。

官方上游容器作为容器在 Quay.io/projectquay/clair 处打包并发布。latest 标签跟踪 Git 开发分支。版本标签从对应的发行版本中构建。

1.2. CLAIR 严重性映射

Clair 提供了全面的漏洞评估和管理方法。其基本功能之一是对安全数据库严重性字符串进行规范化。这个过程通过将漏洞严重性映射到预定义的值，从而简化对漏洞严重性的评估。通过此映射，客户端可以有效地响应漏洞严重性，而无需破坏每个安全数据库的唯一严重性字符串。这些映射的严重性字符串与相应安全数据库中发现的字符串一致，确保漏洞评估的一致性和准确性。

1.2.1. Clair 严重性字符串

Clair 会警告用户有以下严重性字符串：

- Unknown
- negligible
- 低
- Medium
- High
- Critical

这些严重性字符串与相关安全数据库中找到的字符串类似。

alpine 映射
alpine SecDB 数据库不提供严重性信息。所有漏洞严重性都将为 Unknown。

alpine 严重性	Clair 严重性
*	Unknown

AWS 映射
AWS UpdateInfo 数据库提供严重性信息。

AWS 严重性	Clair 严重性
低	低
中	Medium
重要	High
critical	Critical

Debian 映射
Debian Oval 数据库提供严重性信息。

Debian 严重性	Clair 严重性
*	Unknown
Unimportant	低
低	Medium
Medium	High
High	Critical

Oracle 映射
Oracle Oval 数据库提供严重性信息。

Oracle 严重性	Clair 严重性
N/A	Unknown
低	低
中	Medium
重要	High
CRITICAL	Critical

RHEL 映射
RHEL Oval 数据库提供严重性信息。

RHEL 严重性	Clair 严重性
None	Unknown
低	低
Moderate（中度）	Medium
重要的	High
Critical	Critical

SUSE 映射
SUSE Oval 数据库提供严重性信息。

重要性	Clair 严重性
None	Unknown
低	低
Moderate（中度）	Medium
重要的	High
Critical	Critical

Ubuntu 映射
Ubuntu Oval 数据库提供严重性信息。

重要性	Clair 严重性
Untriaged	Unknown
negligible	negligible
低	低
Medium	Medium
High	High
Critical	Critical

OSV 映射

表 1.1. CVSSv3

基本分数	Clair 严重性
0.0	negligible
0.1-3.9	低
4.0-6.9	Medium
7.0-8.9	High
9.0-10.0	Critical

表 1.2. CVSSv2

基本分数	Clair 严重性
0.0-3.9	低
4.0-6.9	Medium
7.0-10	High

第 2 章 CLAIR 概念

以下小节提供了 Clair 的工作原理的概念概述。

2.1. 实践中的 CLAIR

Clair 分析被分为三个不同的部分：索引、匹配和通知。

2.1.1. 索引

Clair 的 indexer 服务在了解容器镜像的组成方面扮演了关键角色。在 Clair 中，名为 "manifests" 的容器镜像表示。清单用于理解镜像层的内容。为简化此过程，Clair 利用了开放容器项目(OCI)清单和层的事实，旨在提供内容寻址，从而减少重复性任务。

在索引过程中，会提取一个代表容器镜像的清单，并划分成其基本组件。索引程序的作业是取消镜像包含的软件包、其原始分发及其所依赖的软件包存储库。然后，在 Clair 的数据库中记录并存储在这些宝贵的信息。索引期间收集的见解作为生成全面漏洞报告的基础。此报告可无缝传输到匹配节点以进一步分析和操作，帮助用户对其容器镜像的安全性做出明智的决策。

IndexReport 存储在 Clair 的数据库中。它可以是 匹配 节点来计算漏洞报告。

2.1.2. 匹配

使用 Clair 时，匹配节点负责将漏洞与提供的索引报告匹配。

匹配者负责保持漏洞数据库最新。matchers 运行一组更新程序，它会定期探测到新内容的数据源。在发现新漏洞时，会将其存储在数据库中。

匹配器 API 设计为在查询时始终提供最新的漏洞报告。漏洞报告总结了清单的内容以及影响内容的任何漏洞。

在发现新漏洞时，会将其存储在数据库中。

匹配器 API 设计为经常使用。它设计为在查询时始终提供最新的 **VulnerabilityReport**。**VulnerabilityReport** 总结了清单的内容以及影响内容的任何漏洞。

2.1.3. 通知程序服务

Clair 使用一个通知程序服务来跟踪新的安全数据库更新，并通知用户是否有新的或移除的漏洞会影响索引的清单。

当通知程序了解影响之前索引清单的新漏洞时，它使用 **config.yaml** 文件中配置的方法来发出有关新更改的通知。返回的通知会表达因为更改而发现的最严重漏洞。这可避免为同一安全数据库更新创建过量通知。

当用户收到通知时，它会针对匹配者发出一个新请求，以接收最新的漏洞报告。

您可以通过以下机制订阅通知：

- Webhook 提供
- AMQP 传输
- STOMP delivery

配置通知程序通过 Clair YAML 配置文件完成。

2.2. CLAIR 身份验证

在当前的迭代中，Clair v4 (Clair)在内部处理身份验证。

注意

以前的 Clair 版本使用 JWT 代理来授权身份验证。

身份验证是通过在配置的 **auth** 键下指定配置对象进行配置的。可能存在多个身份验证配置，但它们会优先按照以下顺序使用：

1. PSK.通过此身份验证配置，Clair 使用预共享密钥实施基于 JWT 的身份验证。
2. 配置。例如：

```
auth:
  psk:
    key: >-
      MDQ4ODBlNDAtNDc0ZC00MWUxLThhMzAtOTk0MzEwMGQwYTMxCg==
    iss: 'issuer'
```

在此配置中，**auth** 字段需要两个参数： 是，这是验证所有传入请求的签发者，以及验证请求的 base64 代码对称密钥。

2.3. CLAIR 更新器

Clair 使用名为 ***updaters*** 的 **Go** 软件包，其中包含获取和解析不同漏洞数据库的逻辑。

更新程序通常与一个匹配器配对，以解释是否存在以及漏洞与软件包相关的漏洞。管理员可能希望更频繁地更新漏洞数据库，或者不能从它们知道的数据库导入漏洞。

2.4. 有关 CLAIR 更新器的信息

下表提供了每个 **Clair** 更新器的详细信息，包括配置参数、简短描述、相关 **URL** 以及它们与之交互的相关组件。此列表并不完整，一些服务器可能会发布重定向，而某些请求 **URL** 则会被动态构建以确保准确的漏洞数据检索。

对于 **Clair**，每个更新程序负责获取和解析与特定软件包类型或分发相关的漏洞数据。例如，**Debian** 更新器侧重于基于 **Debian** 的 **Linux** 发行版，而 **AWS** 更新器则侧重于特定于 **Amazon Web Services** 的 **Linux** 发行版的漏洞。了解软件包类型对于漏洞管理非常重要，因为不同的软件包类型可能具有唯一的安全性问题，并且需要特定的更新和补丁。

注意

如果您使用带有 **Clair** 更新器 **URL** 的环境中的代理服务器，您必须识别哪些 **URL** 需要添加到代理允许列表中，以确保 **Clair** 可以访问它们。使用下表将更新器 **URL** 添加到代理允许列表中。

表 **2.1. Clair updater** 信息

Updater	描述	urls	组件
alpine	Alpine 更新器负责获取和解析与 Alpine Linux 发行版中软件包相关的漏洞数据。	• **https://secdb.alpinelinux.org/**	alpine Linux SecDB 数据库
aws	AWS 更新器专注于基于 AWS Linux 的软件包，确保特定于 Amazon Web Services 的自定义 Linux 发行版的漏洞信息保持最新状态。	• **http://repo.us-west-2.amazonaws.com/2018.03/updates/x86_64/mirror.list** • **https://cdn.amazonlinux.com/2/core/latest/x86_64/mirror.list** • **https://cdn.amazonlinux.com/al2023/core/mirrors/latest/x86_64/mirror.list**	Amazon Web Services (AWS) UpdateInfo
Debian	Debian 更新器对于跟踪与基于 Debian 的 Linux 发行版相关的软件包中的漏洞至关重要。	• **https://deb.debian.org/** • **https://security-tracker.debian.org/tracker/data/json**	Debian 安全跟踪器
clair.cvss	Clair 通用漏洞评分系统(CVSS)更新程序侧重于维护有关漏洞及其相关 CVSS 分数的数据。这不依赖于特定的软件包类型，而是与漏洞的严重性和风险评估相关联。	• **https://nvd.nist.gov/feeds/json/cve/1.1/**	JSON 格式的通用漏洞和暴露(CVE)数据的国家漏洞数据库(NVD)源
oracle	Oracle 更新程序专用于 Oracle Linux 软件包，维护影响 Oracle Linux 系统的漏洞中的数据。	• **https://linux.oracle.com/security/oval/com.oracle.elsa-*.xml.bz2**	Oracle Oval 数据库

Updater	描述	urls	组件
Photon	Photon 更新器处理 VMware Photon OS 中的软件包。	• **https://packages.vmware.com/photon/photon_oval_definitions/**	VMware Photon OS oval 定义
rhel	Red Hat Enterprise Linux (RHEL) 更新程序负责维护 Red Hat Enterprise Linux 发行版中软件包的漏洞数据。	• **https://access.redhat.com/security/cve/** • **https://access.redhat.com/security/data/oval/v2/PULP_MANIFEST**	Red Hat Enterprise Linux (RHEL) Oval 数据库
rhcc	Red Hat Container Catalog (RHCC)更新器已连接到红帽的容器镜像。此更新可确保与红帽容器化软件相关的漏洞信息保持最新。	• **https://access.redhat.com/security/data/metrics/cvemap.xml**	资源处理程序配置控制器(RHCC)数据库
SUSE	SUSE 更新器管理 SUSE Linux 分发系列中的软件包的漏洞信息，包括 openSUSE、SUSE Enterprise Linux 等。	• **https://support.novell.com/security/oval/**	SUSE Oval 数据库
ubuntu	Ubuntu 更新器专用于跟踪与基于 Ubuntu 的 Linux 发行版关联的软件包中的漏洞。Ubuntu 是 Linux 生态系统中常用的发行版。	• **https://security-metadata.canonical.com/oval/com.ubuntu.*.cve.oval.xml** • **https://api.launchpad.net/1.0/**	Ubuntu Oval 数据库
OSV	开源漏洞(OSV)更新器专门负责跟踪开源软件组件中的漏洞。OSV 是一个关键资源，提供各种开源项目中发现的安全问题的详细信息。	• **https://osv-vulnerabilities.storage.googleapis.com/**	开源漏洞数据库

2.5. 配置更新器

updaters 可以通过 **clair-config.yaml** 文件中的 **updaters.sets** 键进行配置。

重要

- 如果没有填充 **set** 字段，则默认为使用所有集合。在使用所有集合时，**Clair** 会尝试访问每个更新器的 **URL** 或 **URL**。如果使用代理环境，您必须将这些 **URL** 添加到代理允许列表中。
- 如果在匹配器进程（这是默认设置）中自动运行 **updaters**，则运行更新器的周期会在 **matcher** 的配置字段中配置。

2.5.1. 选择特定的更新器集

使用以下引用为 **Red Hat Quay** 部署选择一个或多个更新程序。

为多个更新程序配置 **Clair**

多个特定更新器

```
#...
updaters:
  sets:
    - alpine
    - aws
    - osv
#...
```

为 **Alpine** 配置 **Clair**

alpine config.yaml 示例

```
#...
updaters:
  sets:
    - alpine
#...
```

为 **AWS** 配置 **Clair**

AWS config.yaml 示例

```
#...
updaters:
  sets:
    - aws
#...
```

为 **Debian** 配置 **Clair**

Debian config.yaml 示例

```
#...
updaters:
  sets:
    - debian
#...
```

为 **Clair CVSS** 配置 **Clair**

Clair CVSS config.yaml 示例

```
#...
updaters:
  sets:
    - clair.cvss
#...
```

为 **Oracle** 配置 **Clair**

Oracle config.yaml 示例

```
#...
updaters:
  sets:
    - oracle
#...
```

为 **Photon** 配置 **Clair**

photon config.yaml 示例

```
#...
updaters:
  sets:
    - photon
#...
```

为 **SUSE** 配置 **Clair**

SUSE config.yaml 示例

```
#...
updaters:
  sets:
    - suse
#...
```

为 **Ubuntu** 配置 **Clair**

Ubuntu config.yaml 示例

```
#...
updaters:
  sets:
    - ubuntu
#...
```

为 **OSV** 配置 **Clair**

OSV config.yaml 示例

```
#...
updaters:
  sets:
    - osv
#...
```

2.5.2. 为完整的 Red Hat Enterprise Linux (RHEL)覆盖选择更新器集

对于 **Red Hat Enterprise Linux (RHEL)**的完整覆盖漏洞，您必须使用以下更新器集：

- **RHEL.**此更新程序可确保您拥有影响 **RHEL** 的漏洞的最新信息。
- **RHCC.**此更新器跟踪与红帽容器镜像相关的漏洞。
- **Clair.cvss.**此更新程序通过提供常见漏洞和风险(CVE)分数来全面查看漏洞的严重性和风险评估。
- **OSV.**此更新程序侧重于跟踪开源软件组件中的漏洞。建议根据在 **RHEL** 产品中使用 **Java** 和 **Go** 的频率。

RHEL updaters 示例

```
#...
updaters:
  sets:
    - rhel
    - rhcc
```

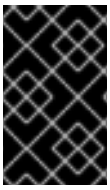
```
    - clair.cvss
    - osv
#...
```

2.5.3. 高级更新器配置

在某些情况下，用户可能希望为特定行为配置更新程序，例如，如果您想要允许列表特定生态系统用于开源漏洞(OSV)更新程序。

高级更新器配置可能对代理部署或 **air gapped** 部署很有用。通过将一个键放在 **updaters** 对象的 **config** 环境变量下，可以传递这些场景中的具体更新程序的配置。用户应检查其 **Clair** 日志到双检查名称。

以下 **YAML** 片断详细介绍了一些 **Clair** 更新器可用的各种设置



重要

对于更多用户，不需要高级更新器配置。

配置 **alpine** 更新器

```
#...
updaters:
  sets:
    - apline
  config:
    alpine:
      url: https://secdb.alpinelinux.org/
#...
```

配置 **debian** 更新器

```
#...
updaters:
  sets:
    - debian
  config:
    debian:
      mirror_url: https://deb.debian.org/
      json_url: https://security-tracker.debian.org/tracker/data/json
#...
```

配置 **clair.cvss updater**

```
#...
updaters:
  config:
    clair.cvss:
      url: https://nvd.nist.gov/feeds/json/cve/1.1/
#...
```

配置 **oracle updater**

```
#...
updaters:
  sets:
    - oracle
  config:
    oracle-2023-updater:
      url:
        - https://linux.oracle.com/security/oval/com.oracle.elsa-2023.xml.bz2
    oracle-2022-updater:
      url:
        - https://linux.oracle.com/security/oval/com.oracle.elsa-2022.xml.bz2
#...
```

配置 **photon** 更新r

```
#...
updaters:
  sets:
    - photon
  config:
    photon:
      url: https://packages.vmware.com/photon/photon_oval_definitions/
#...
```

配置 **rhel updater**

```
#...
updaters:
  sets:
    - rhel
  config:
    rhel:
      url: https://access.redhat.com/security/data/oval/v2/PULP_MANIFEST
      ignore_unpatched: true (1)
#...
```

(1)

布尔值.是否包含有关没有可用相应补丁或更新的漏洞的信息。

配置 **rhcc** 更新器

```
#...
updaters:
  sets:
    - rhcc
  config:
    rhcc:
      url: https://access.redhat.com/security/data/metrics/cvemap.xml
#...
```

配置 **suse updater**

```
#...
updaters:
  sets:
    - suse
  config:
    suse:
      url: https://support.novell.com/security/oval/
#...
```

配置 **CamelAwsS updater**

```
#...
updaters:
  config:
    ubuntu:
      url: https://api.launchpad.net/1.0/
      name: ubuntu
      force: (1)
        - name: focal (2)
          version: 20.04 (3)
#...
```

(1) 用于强制在生成的 **UpdaterSet** 中包含特定发行版和版本详情，而不考虑 **API** 响应中的状态。如果要确保更新器配置中一致包含特定的发行版和版本，则很有用。

(2) 指定要强制包含在 **UpdaterSet** 中的发行版名称。

(3) 指定您要强制到 **UpdaterSet** 的发行版版本。

配置 **osv** 更新器

```
#...
updaters:
  sets:
```

```
    - osv
  config:
    osv:
      url: https://osv-vulnerabilities.storage.googleapis.com/
      allowlist: 1
        - npm
        - pypi
#...
```

1

允许的生态系统列表。当保留未设置时，允许所有生态系统。必须为小写。有关支持的生态系统列表，请查看 定义生态系统 的文档。

2.5.4. 禁用 Clair Updater 组件

在某些情况下，用户可能希望禁用 **Clair** 更新器组件。在断开连接的环境中运行 **Red Hat Quay** 时，需要禁用更新程序。

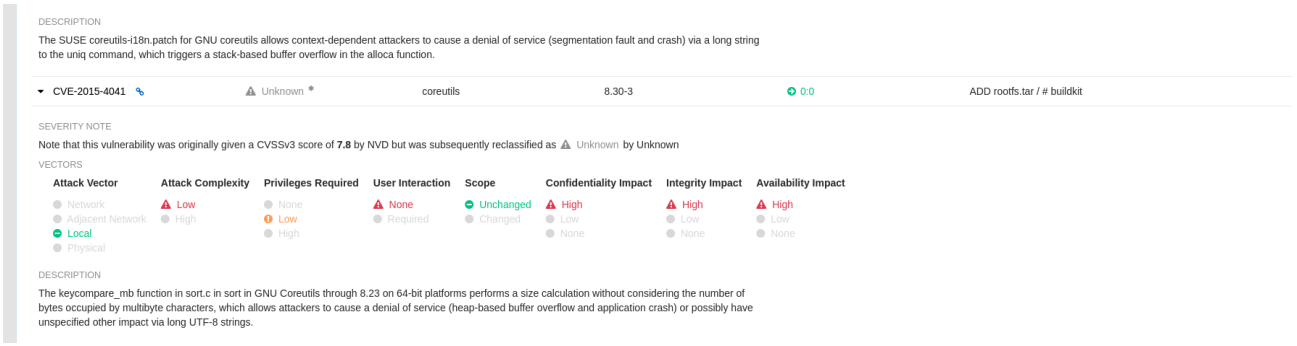
在以下示例中，**Clair** 更新器被禁用：

```
#...
matcher:
  disable_updaters: true
#...
```

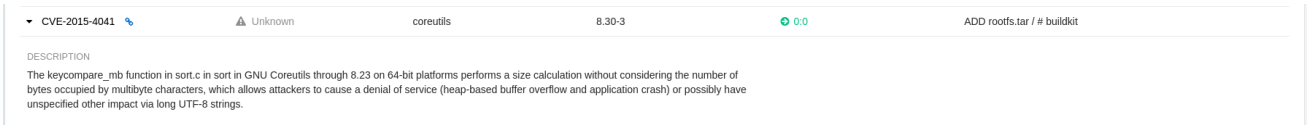
2.6. 国家漏洞数据库中的 CVE 评级

从 **Clair v4.2** 开始，在 **Red Hat Quay UI** 中可以看到通用漏洞评分系统(CVSS)功能数据。另外，**Clair v4.2** 为检测到的漏洞在国家漏洞数据库中添加 **CVSS** 分数。

在这个版本中，如果漏洞的 **CVSS** 分数在发布分数的 **2** 个级别内，**Red Hat Quay UI** 默认会显示发行版分数。例如：



这与之前的接口不同，这只显示以下信息：



2.7. FEDERAL INFORMATION PROCESSING STANDARD (FIPS)就绪度和合规性

美国国家标准与技术研究所**(NIST)**开发的联邦信息处理标准**(FIPS)**被认为是保护和加密敏感数据的高度规范性领域，特别是银行、医疗和公共部门等高度监管区域。**Red Hat Enterprise Linux (RHEL)**和 **OpenShift Container Platform** 通过提供 ***FIPS 模式*** 来支持 **FIPS**，系统只允许使用特定 **FIPS** 验证的加密模块，如 **openssl**。这样可确保 **FIPS** 合规性。

2.7.1. 启用 FIPS 合规性

使用以下步骤在 **Red Hat Quay** 部署中启用 **FIPS** 合规性。

前提条件

- 如果您正在运行独立部署 **Red Hat Quay**，您的 **Red Hat Enterprise Linux (RHEL)**部署是版本 **8** 或更高版本，启用了 **FIPS**。
- 如果要在 **OpenShift Container Platform** 上部署 **Red Hat Quay**，**OpenShift Container Platform** 是 **4.10** 或更高版本。
- 您的 **Red Hat Quay** 版本为 **3.5.0** 或更高版本。

- 如果您在 **IBM Power** 或 **IBM Z** 集群上的 **OpenShift Container Platform** 中使用 **Red Hat Quay**：
  - **OpenShift Container Platform** 版本 **4.14** 或更高版本是必需的
  - **Red Hat Quay** 版本 **3.10** 或更高版本是必需的
- 您有 **Red Hat Quay** 部署的管理特权。

流程

- 在 **Red Hat Quay config.yaml** 文件中，将 **FEATURE_FIPS** 配置字段设置为 **true**。例如：

```
---
FEATURE_FIPS = true
---
```

将 **FEATURE_FIPS** 设置为 **true** 时，**Red Hat Quay** 会使用 **FIPS** 兼容的哈希功能运行。

部分 II. RED HAT QUAY 中的 CLAIR

本指南包含在独立和 **OpenShift Container Platform Operator** 部署中在 **Red Hat Quay** 上运行 **Clair** 的步骤。

第 3 章 在独立 RED HAT QUAY 部署中设置 CLAIR

对于独立的 **Red Hat Quay** 部署，您可以手动设置 **Clair**。

流程

1. 在 **Red Hat Quay** 安装目录中，为 **Clair** 数据库数据创建一个新目录：

```
$ mkdir /home/<user-name>/quay-poc/postgres-clairv4
```

2. 输入以下命令为 **postgres-clairv4** 文件设置适当的权限：

```
$ setfacl -m u:26:-wx /home/<user-name>/quay-poc/postgres-clairv4
```

3. 输入以下命令部署 **Clair PostgreSQL** 数据库：

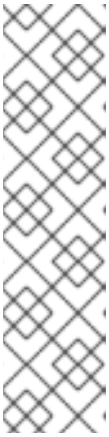
```
$ sudo podman run -d --name postgresql-clairv4 \
  -e POSTGRESQL_USER=clairuser \
  -e POSTGRESQL_PASSWORD=clairpass \
  -e POSTGRESQL_DATABASE=clair \
  -e POSTGRESQL_ADMIN_PASSWORD=adminpass \
  -p 5433:5432 \
  -v /home/<user-name>/quay-poc/postgres-clairv4:/var/lib/pgsql/data:Z \
  registry.redhat.io/rhel8/postgresql-13:1-109
```

4. 为您的 **Clair** 部署安装 **PostgreSQL uuid-ossp** 模块：

```
$ podman exec -it postgresql-clairv4 /bin/bash -c 'echo "CREATE EXTENSION IF NOT EXISTS \"uuid-ossp\"" | psql -d clair -U postgres'
```

输出示例

```
CREATE EXTENSION
```



注意

Clair 要求将 **uuid-ossp** 扩展添加到其 **PostgreSQL** 数据库中。对于具有适当特权的用户，创建扩展将由 **Clair** 自动添加。如果用户没有正确的特权，必须在启动 **Clair** 前添加扩展。

如果扩展不存在，在 **Clair** 尝试启动时会显示以下错误： **ERROR:** 载入 **"uuid-ossp"** 扩展。**(SQLSTATE 42501)**。

5. 如果 **Quay** 容器正在运行，并在配置模式中重启它，将现有配置载入为卷：

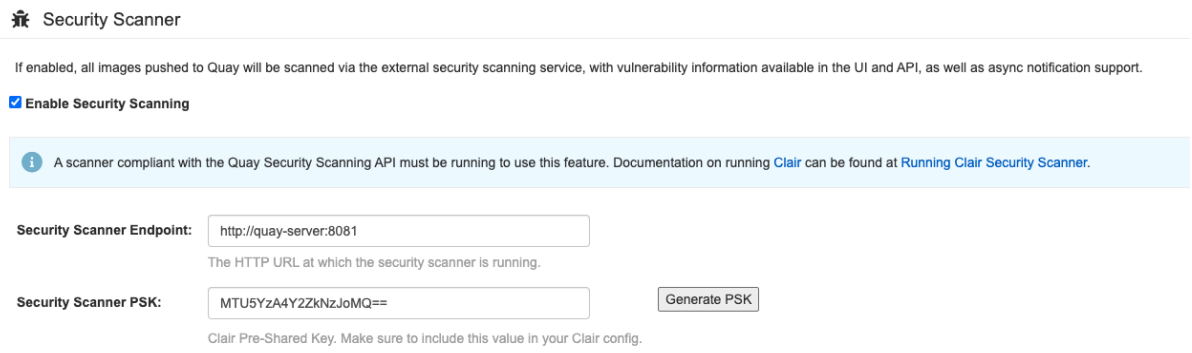
```
$ sudo podman run --rm -it --name quay_config \
  -p 80:8080 -p 443:8443 \
  -v $QUAY/config:/conf/stack:Z \
  {productrepo}/{quayimage}:{productminv} config secret
```

6. 登录到配置工具，再点 **UI** 的 **Security Scanner** 部分中的 **Enable Security Scanning**。

7. 使用 **quay-server** 系统上尚未使用的端口，为 **Clair** 设置 **HTTP** 端点，例如 **8081**。

8. 使用 **Generate PSK** 按钮创建一个预共享密钥**(PSK)**。

安全扫描器 **UI**



9. 验证并下载 **Red Hat Quay** 的 **config.yaml** 文件，然后停止运行配置编辑器的 **Quay** 容器。

10. 将新配置捆绑包提取到 **Red Hat Quay** 安装目录中，例如：

```
$ tar xvf quay-config.tar.gz -d /home/<user-name>/quay-poc/
```

11. 为您的 **Clair** 配置文件创建一个文件夹，例如：

```
$ mkdir /etc/opt/clairv4/config/
```

12. 进入 **Clair** 配置文件夹：

```
$ cd /etc/opt/clairv4/config/
```

13. 创建 **Clair** 配置文件，例如：

```
http_listen_addr: :8081
introspection_addr: :8088
log_level: debug
indexer:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
matcher:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  max_conn_pool: 100
  migrations: true
  indexer_addr: clair-indexer
notifier:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  delivery_interval: 1m
  poll_interval: 5m
  migrations: true
auth:
  psk:
    key: "MTU5YzA4Y2ZkNzJoMQ=="
    iss: ["quay"]
# tracing and metrics
trace:
  name: "jaeger"
  probability: 1
  jaeger:
    agent:
      endpoint: "localhost:6831"
```

```
    service_name: "clair"
metrics:
  name: "prometheus"
```

有关 **Clair** 配置格式的更多信息，请参阅 **Clair** 配置参考。

14. 使用容器镜像启动 **Clair**，从您创建的文件中挂载到配置中：

```
$ sudo podman run -d --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
-e CLAIR_MODE=combo \
-v /etc/opt/clairv4/config:/clair:Z \
registry.redhat.io/quay/clair-rhel8:v3.11.1
```

注意

也可以运行多个 **Clair** 容器，但为了在单一容器之外部署场景，强烈建议使用 **Kubernetes** 或 **OpenShift Container Platform** 等容器编配器。

3.1. 使用带有上游 RED HAT QUAY 的 CLAIR

对于大多数用户，需要独立于当前版本的 **Clair** 升级**(4.7.2)**。然而，在某些情况下，出于各种原因，客户可能希望从 上游存储库 中拉取 **Clair** 镜像，如特定程序错误修复或尝试尚未发布下游的新功能。您可以使用 **Red Hat Quay** 运行 **Clair** 的上游版本。

重要

Clair 的上游版本还没有经过全面测试，以便与 **Red Hat Quay** 的兼容性。因此，这个组合可能会导致部署出现问题。

流程

1. 如果 **Clair** 正在运行，输入以下命令来停止 **Clair**：

```
$ podman stop <clairv4_container_name>
```

2. 导航到 上游存储库，找到要使用的 **Clair** 版本，并将它拉取到本地机器。例如：

```
$ podman pull quay.io/projectquay/clair:nightly-2024-02-03
```

3. 使用容器镜像启动 **Clair**，从您创建的文件中挂载到配置中：

```
$ podman run -d --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
-e CLAIR_MODE=combo \
-v /etc/opt/clairv4/config:/clair:Z \
quay.io/projectquay/clair:nightly-2024-02-03
```

第 4 章 OPENSHIFT CONTAINER PLATFORM 上的 CLAIR

要在 **OpenShift Container Platform** 上的 **Red Hat Quay** 部署上设置 **Clair v4 (Clair)**，建议使用 **Red Hat Quay Operator**。默认情况下，**Red Hat Quay Operator** 会随 **Red Hat Quay** 部署一起安装或升级 **Clair** 部署，并自动配置 **Clair**。

第 5 章 测试 CLAIR

使用以下步骤在独立 **Red Hat Quay** 部署或基于 **OpenShift Container Platform Operator** 的部署中测试 **Clair**。

先决条件

- 您已部署了 **Clair** 容器镜像。

流程

1. 输入以下命令拉取示例镜像：

```
$ podman pull ubuntu:20.04
```

2. 输入以下命令将镜像标记到 **registry**：

```
$ sudo podman tag docker.io/library/ubuntu:20.04 <quay-server.example.com>/<user-name>/ubuntu:20.04
```

3. 输入以下命令将镜像推送到 **Red Hat Quay registry**：

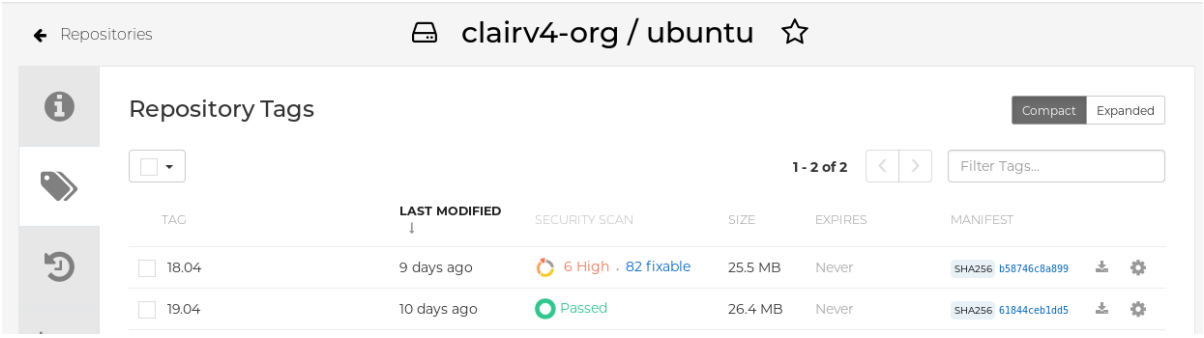
```
$ sudo podman push --tls-verify=false quay-server.example.com/quayadmin/ubuntu:20.04
```

4. 通过 **UI** 登录您的 **Red Hat Quay** 部署。

5. 单击存储库名称，如 **quayadmin/ubuntu**。

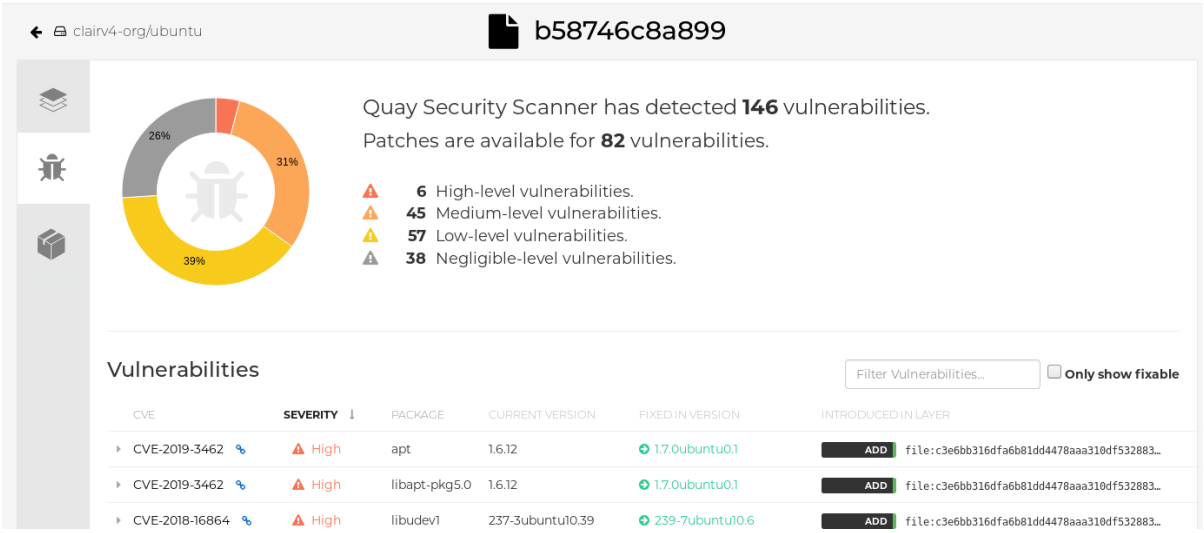
6. 在导航窗格中，单击 **Tags**。

报告概述



7. 点镜像报告（如 **45** 介质 ）来显示更详细的报告：

报告详情



注意

在某些情况下，**Clair** 会显示镜像重复报告，如 **ubi8/nodejs-12** 或 **ubi8/nodejs-16**。这是因为名称相同的漏洞用于不同的软件包。这个行为应该带有 **Clair** 漏洞报告，且不会作为程序错误解决。

部分 III. 高级 CLAIR 配置

使用本节配置高级 **Clair** 功能。

第 6 章 UNMANAGED CLAIR 配置

Red Hat Quay 用户可以使用 **Red Hat Quay OpenShift Container Platform Operator** 运行非受管 **Clair** 配置。此功能允许用户创建非受管 **Clair** 数据库，或在没有非受管数据库的情况下运行自定义 **Clair** 配置。

非受管 **Clair** 数据库允许 **Red Hat Quay Operator** 在地理复制环境中工作，其中 **Operator** 的多个实例必须与同一数据库通信。当用户需要在集群外存在的高可用性**(HA) Clair** 数据库时，也可以使用非受管 **Clair** 数据库。

6.1. 使用非受管 CLAIR 数据库运行自定义 CLAIR 配置

使用以下步骤将 **Clair** 数据库设置为 **unmanaged**。

流程

- 在 **Quay Operator** 中，将 **QuayRegistry** 自定义资源的 **clairpostgres** 组件设置为 **managed: false** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay370
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: clairpostgres
      managed: false
```

6.2. 使用非受管 CLAIR 数据库配置自定义 CLAIR 数据库

OpenShift Container Platform 上的 **Red Hat Quay** 允许用户提供自己的 **Clair** 数据库。

使用以下步骤创建自定义 **Clair** 数据库。

注意

以下流程使用 **SSL/TLS** 认证设置 **Clair**。要查看没有使用 **SSL/TSL** 认证设置 **Clair** 的类似流程，请参阅**"**使用受管 **Clair** 配置配置自定义 **Clair** 数据库**"**。

流程

1. 输入以下命令，创建包含 **clair-config.yaml** 的 **Quay** 配置捆绑包 **secret**：

```
$ oc create secret generic --from-file config.yaml=./config.yaml --from-file
extra_ca_cert_rds-ca-2019-root.pem=./rds-ca-2019-root.pem --from-file clair-
config.yaml=./clair-config.yaml --from-file ssl.cert=./ssl.cert --from-file ssl.key=./ssl.key
config-bundle-secret
```

Clair config.yaml 文件示例

```
indexer:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem
sslmode=verify-ca
  layer_scan_concurrency: 6
  migrations: true
  scanlock_retry: 11
log_level: debug
matcher:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem
sslmode=verify-ca
  migrations: true
metrics:
  name: prometheus
notifier:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem
sslmode=verify-ca
  migrations: true
```

注意

- 数据库证书挂载到 **clair-config.yaml** 中的 **Clair** 应用程序 **pod** 的 **/run/certs/rds-ca-2019-root.pem** 下。在配置 **clair-config.yaml** 时必须指定它。
- Clair on OpenShift config 中包括了一个 **clair-config.yaml** 示例。

2. 将 **clair-config.yaml** 文件添加到捆绑包 **secret** 中，例如：

```
apiVersion: v1
kind: Secret
metadata:
  name: config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: <base64 encoded Quay config>
  clair-config.yaml: <base64 encoded Clair config>
  extra_ca_cert_<name>: <base64 encoded ca cert>
  ssl.crt: <base64 encoded SSL certificate>
  ssl.key: <base64 encoded SSL private key>
```

注意

更新时，提供的 **clair-config.yaml** 文件将挂载到 **Clair pod**。任何未提供的字段都使用 **Clair** 配置模块自动填充默认值。

3. 您可以通过点 **Build History** 页面中的提交或运行 **oc get pods -n <namespace >** 来检查 **Clair pod** 的状态。例如：

```
$ oc get pods -n <namespace>
```

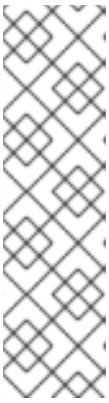
输出示例

```
NAME                                READY   STATUS    RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Running   0          7s
```

第 7 章 使用受管 CLAIR 数据库运行自定义 CLAIR 配置

在某些情况下，用户可能希望使用受管 **Clair** 数据库运行自定义 **Clair** 配置。这在以下情况中很有用：

- 当用户希望禁用特定的更新器资源时。
- 当用户在断开连接的环境中运行 **Red Hat Quay** 时。有关在断开连接的环境中运行 **Clair** 的更多信息，请参阅 断开连接的环境中的 **Clair**。



注意

- 如果您在断开连接的环境中运行 **Red Hat Quay**，则 **clair-config.yaml** 的 **airgap** 参数必须设置为 **true**。
- 如果您在断开连接的环境中运行 **Red Hat Quay**，您应该禁用所有更新组件。

7.1. 将 CLAIR 数据库设置为受管

使用以下步骤将 **Clair** 数据库设置为 **managed**。

流程

- 在 **Quay Operator** 中，将 **QuayRegistry** 自定义资源的 **clairpostgres** 组件设置为 **managed: true** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay370
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
```

```
    managed: false
  - kind: clairpostgres
    managed: true
```

7.2. 使用受管 CLAIR 配置配置自定义 CLAIR 数据库

OpenShift Container Platform 上的 **Red Hat Quay** 允许用户提供自己的 **Clair** 数据库。

使用以下步骤创建自定义 **Clair** 数据库。

流程

1. 输入以下命令，创建包含 **clair-config.yaml** 的 **Quay** 配置捆绑包 **secret**：

```
$ oc create secret generic --from-file config.yaml=./config.yaml --from-file
extra_ca_cert_rds-ca-2019-root.pem=./rds-ca-2019-root.pem --from-file clair-
config.yaml=./clair-config.yaml config-bundle-secret
```

Clair config.yaml 文件示例

```
indexer:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslmode=disable
  layer_scan_concurrency: 6
  migrations: true
  scanlock_retry: 11
log_level: debug
matcher:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslmode=disable
  migrations: true
metrics:
  name: prometheus
notifier:
  connstring: host=quay-server.example.com port=5432 dbname=quay
user=quayrdsdb password=quayrdsdb sslmode=disable
  migrations: true
```

注意

- 数据库证书挂载到 **clair-config.yaml** 中的 **Clair** 应用程序 **pod** 的 **/run/certs/rds-ca-2019-root.pem** 下。在配置 **clair-config.yaml** 时必须指定它。
- Clair on OpenShift config 中包括了一个 **clair-config.yaml** 示例。

2. 将 **clair-config.yaml** 文件添加到捆绑包 **secret** 中，例如：

```
apiVersion: v1
kind: Secret
metadata:
  name: config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: <base64 encoded Quay config>
  clair-config.yaml: <base64 encoded Clair config>
```

注意

- 更新时，提供的 **clair-config.yaml** 文件将挂载到 **Clair pod**。任何未提供的字段都使用 **Clair** 配置模块自动填充默认值。

3. 您可以通过点 **Build History** 页面中的提交或运行 **oc get pods -n <namespace >** 来检查 **Clair pod** 的状态。例如：

```
$ oc get pods -n <namespace>
```

输出示例

```
NAME                                                READY   STATUS    RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Running   0          7s
```

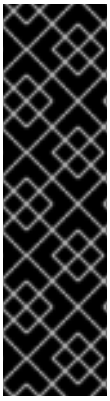
第 8 章 在断开连接的环境中的 CLAIR

注意

目前，**IBM Power** 和 **IBM Z** 不支持在断开连接的环境中部署 **Clair**。

Clair 使用一组称为 ***updaters*** 的组件来处理从各种漏洞数据库获取和解析数据。默认情况下，设置了 **updaters**，以直接从互联网拉取漏洞数据，并可立即使用。但是，有些用户可能需要 **Red Hat Quay** 在断开连接的环境中运行，或者不需要直接访问互联网的环境。**Clair** 支持断开连接的环境，方法是使用不同类型的更新工作流来考虑网络隔离。这通过使用 **clairctl** 命令行界面工具工作，它通过使用开放主机从互联网获取更新器数据，安全地将数据传送到隔离主机，然后对隔离主机上的更新器数据非常重要。

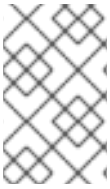
使用本指南在断开连接的环境中部署 **Clair**。



重要

由于已知问题 **PROJQUAY-6577**，**Red Hat Quay Operator** 无法正确呈现自定义的 **Clair config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 **Clair** 配置，而不依赖于 **Operator** 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 **Clair** 扫描 的说明。



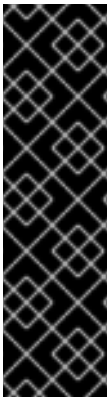
注意

目前，**Clair** 增强数据是 **CVSS** 数据。目前，在断开连接的环境中不支持增强数据。

有关 **Clair** 更新器的更多信息，请参阅**"Clair updaters"**。

8.1. 在断开连接的 OPENSHIFT CONTAINER PLATFORM 集群中设置 CLAIR

使用以下步骤在断开连接的 **OpenShift Container Platform** 集群中设置置备的 **Clair pod**。



重要

由于已知问题 PROJQUAY-6577，**Red Hat Quay Operator** 无法正确呈现自定义的 **Clair config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 **Clair** 配置，而不依赖于 **Operator** 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 **Clair** 扫描 的说明。

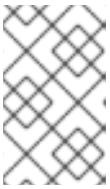
8.1.1. 为 OpenShift Container Platform 部署安装 clairctl 命令行工具

使用以下步骤为 **OpenShift Container Platform** 部署安装 **clairctl CLI** 工具。

流程

1. 输入以下命令，在 **OpenShift Container Platform** 集群中为 **Clair** 部署安装 **clairctl** 程序：

```
$ oc -n quay-enterprise exec example-registry-clair-app-64dd48f866-6ptgw -- cat /usr/bin/clairctl > clairctl
```



注意

不正式使用，可以下载 **clairctl** 工具

2. 设置 **clairctl** 文件的权限，以便用户可以执行并运行它，例如：

```
$ chmod u+x ./clairctl
```

8.1.2. 为 OpenShift Container Platform 上的 Clair 部署检索并解码 Clair 配置 secret

使用以下步骤为 **OpenShift Container Platform** 上置备的 **OpenShift Container Platform** 置备 **Clair** 实例检索并解码配置 **secret**。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。

流程

1. 输入以下命令来检索和解码配置 **secret**，然后将其保存到 **Clair** 配置 **YAML** 中：

```
$ oc get secret -n quay-enterprise example-registry-clair-config-secret  -o "jsonpath=
{$.data['config\.yaml']}" | base64 -d > clair-config.yaml
```

2. 更新 **clair-config.yaml** 文件，使 **disable_updaters** 和 **airgap** 参数设置为 **true**，例如：

```
---
indexer:
  airgap: true
---
matcher:
  disable_updaters: true
---
```

8.1.3. 从连接的 Clair 实例导出更新程序捆绑包

使用以下步骤从可访问互联网的 **Clair** 实例导出更新器捆绑包。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 **Clair** 配置 **secret**，并将其保存到 **Clair config.yaml** 文件中。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。

流程

- 在可以访问互联网的 **Clair** 实例中，将 **clairctl CLI** 工具与配置文件一起使用，以导出更新器捆绑包。例如：

```
$ ./clairctl --config ./config.yaml export-updaters updates.gz
```

8.1.4. 在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问

使用以下步骤在断开连接的 **OpenShift Container Platform** 集群中配置对 **Clair** 数据库的访问。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 **Clair** 配置 **secret**，并将其保存到 **Clair config.yaml** 文件中。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 **Clair** 实例导出了更新器捆绑包。

流程

1. 使用 **oc CLI** 工具确定 **Clair** 数据库服务，例如：

```
$ oc get svc -n quay-enterprise
```

输出示例

```
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
example-registry-clair-app          ClusterIP     172.30.224.93    <none>
80/TCP,8089/TCP                4d21h
example-registry-clair-postgres     ClusterIP     172.30.246.88    <none>        5432/TCP
4d21h
...
```

2. 转发 **Clair** 数据库端口，使其可从本地机器访问。例如：

```
$ oc port-forward -n quay-enterprise service/example-registry-clair-postgres 5432:5432
```

3. 更新 **Clair config.yaml** 文件，例如：

```
indexer:
  connstring: host=localhost port=5432 dbname=postgres user=postgres
```

```
password=postgres sslmode=disable 1
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
  scanner:
    repo:
      rhel-repository-scanner: 2
        repo2cpe_mapping_file: /data/cpe-map.json
    package:
      rhel_containerscanner: 3
        name2repos_mapping_file: /data/repo-map.json
```

1 在多 **connstring** 字段中使用 **localhost** 替换 **host** 的值。

2 有关 **rhel-repository-scanner** 参数的更多信息，请参阅 **"Mapping repository to Common Product Enumeration information"**。

3 有关 **rhel_containerscanner** 参数的更多信息，请参阅 **"Mapping repository to Common Product Enumeration information"**。

8.1.5. 将更新器捆绑包导入到断开连接的 OpenShift Container Platform 集群中

使用以下步骤将 **updaters** 捆绑包导入到断开连接的 **OpenShift Container Platform** 集群中。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 **Clair** 配置 **secret**，并将其保存到 **Clair config.yaml** 文件中。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 **Clair** 实例导出了更新器捆绑包。

- 您已将更新程序捆绑包传输到断开连接的环境中。

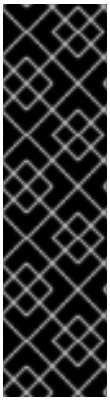
流程

- 使用 **clairctl CLI** 工具将更新器捆绑包导入到 **OpenShift Container Platform** 部署的 **Clair** 数据库中。例如：

```
$ ./clairctl --config ./clair-config.yaml import-updaters updates.gz
```

8.2. 为断开连接的 OPENSHIFT CONTAINER PLATFORM 集群设置 CLAIR 的自我管理部署

使用以下步骤为断开连接的 **OpenShift Container Platform** 集群设置 **Clair** 的自我管理部署。



重要

由于已知问题 **PROJQUAY-6577**，**Red Hat Quay Operator** 无法正确呈现自定义的 **Clair config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 **Clair** 配置，而不依赖于 **Operator** 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 **Clair** 扫描 的说明。

8.2.1. 在 OpenShift Container Platform 上安装用于自我管理的 Clair 部署的 clairctl 命令行工具

使用以下步骤在 **OpenShift Container Platform** 上安装用于自我管理的 **Clair** 部署的 **clairctl CLI** 工具。

流程

1. 使用 **podman cp** 命令为自我管理的 **Clair** 部署安装 **clairctl** 程序，例如：

```
$ sudo podman cp clairv4:/usr/bin/clairctl ./clairctl
```

2. 设置 **clairctl** 文件的权限，以便用户可以执行并运行它，例如：

```
$ chmod u+x ./clairctl
```

8.2.2. 为断开连接的 OpenShift Container Platform 集群部署自我管理的 Clair 容器

使用以下步骤为断开连接的 **OpenShift Container Platform** 集群部署自我管理的 **Clair** 容器。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。

流程

1. 为您的 **Clair** 配置文件创建一个文件夹，例如：

```
$ mkdir /etc/clairv4/config/
```

2. 创建 **Clair** 配置文件，并将 **disable_updaters** 参数设置为 **true**，例如：

```
---
indexer:
  airgap: true
---
matcher:
  disable_updaters: true
---
```

3. 使用容器镜像启动 **Clair**，从您创建的文件中挂载到配置中：

```
$ sudo podman run -it --rm --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
-e CLAIR_MODE=combo \
-v /etc/clairv4/config:/clair:Z \
registry.redhat.io/quay/clair-rhel8:v3.11.1
```

8.2.3. 从连接的 Clair 实例导出更新程序捆绑包

使用以下步骤从可访问互联网的 **Clair** 实例导出更新器捆绑包。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。

- 您已部署了 **Clair**。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。

流程

- 在可以访问互联网的 **Clair** 实例中，将 **clairctl CLI** 工具与配置文件一起使用，以导出更新器捆绑包。例如：

```
$ ./clairctl --config ./config.yaml export-updaters updates.gz
```

8.2.4. 在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问

使用以下步骤在断开连接的 **OpenShift Container Platform** 集群中配置对 **Clair** 数据库的访问。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已部署了 **Clair**。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 **Clair** 实例导出了更新器捆绑包。

流程

1. 使用 **oc CLI** 工具确定 **Clair** 数据库服务，例如：

```
$ oc get svc -n quay-enterprise
```

输出示例

```
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
example-registry-clair-app          ClusterIP      172.30.224.93    <none>
80/TCP,8089/TCP                4d21h
example-registry-clair-postgres     ClusterIP      172.30.246.88    <none>       5432/TCP
4d21h
...
```

2. 转发 **Clair** 数据库端口，使其可从本地机器访问。例如：

```
$ oc port-forward -n quay-enterprise service/example-registry-clair-postgres 5432:5432
```

3. 更新 **Clair config.yaml** 文件，例如：

```
indexer:
  connstring: host=localhost port=5432 dbname=postgres user=postgres
password=postgres sslmode=disable (1)
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
  scanner:
    repo:
      rhel-repository-scanner: (2)
        repo2cpe_mapping_file: /data/cpe-map.json
    package:
      rhel_containerscanner: (3)
        name2repos_mapping_file: /data/repo-map.json
```

(1) 在多 **connstring** 字段中使用 **localhost** 替换 **host** 的值。

(2) 有关 **rhel-repository-scanner** 参数的更多信息，请参阅 **"Mapping repository to Common Product Enumeration information"**。

(3) 有关 **rhel_containerscanner** 参数的更多信息，请参阅 **"Mapping repository to Common Product Enumeration information"**。

8.2.5. 将更新器捆绑包导入到断开连接的 OpenShift Container Platform 集群中

使用以下步骤将 **updaters** 捆绑包导入到断开连接的 **OpenShift Container Platform** 集群中。

先决条件

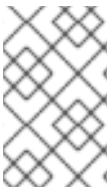
- 您已安装了 **clairctl** 命令行工具工具。
- 您已部署了 **Clair**。
- 在 **Clair config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 **Clair** 实例导出了更新器捆绑包。
- 您已将更新程序捆绑包传输到断开连接的环境中。

流程

- 使用 **clairctl CLI** 工具将更新器捆绑包导入到 **OpenShift Container Platform** 部署的 **Clair** 数据库中：

```
$ ./clairctl --config ./clair-config.yaml import-updaters updates.gz
```

8.3. 将软件仓库映射到通用产品枚举信息



注意

目前，**IBM Power** 和 **IBM Z** 不支持将软件仓库映射到通用产品枚举信息。

Clair 的 **Red Hat Enterprise Linux (RHEL)**扫描程序依赖于通用产品枚举**(CPE)**文件将 **RPM** 软件包映射到对应的安全数据，以生成匹配的结果。这些文件归产品安全性及每日更新。

必须存在 **cp** 文件，或者必须允许访问该文件，以便扫描程序正确处理 **RPM** 软件包。如果文件不存在，则不会扫描容器镜像中安装的 **RPM** 软件包。

表 **8.1. Clair cp** 映射文件

CPE	JSON 映射文件的链接
repos2cpe	Red Hat Repository-to-CPE JSON
names2repos	Red Hat Name-to-Repos JSON.

除了将 **CVE** 信息上传到断开连接的 **Clair** 安装的数据库外，还必须在本地提供映射文件：

- 对于独立的 **Red Hat Quay** 和 **Clair** 部署，必须将映射文件加载到 **Clair pod**。
- 对于 **OpenShift Container Platform** 部署的 **Red Hat Quay**，您必须将 **Clair** 组件设置为 **unmanaged**。然后，必须手动部署 **Clair**，将配置设置为加载映射文件的本地副本。

8.3.1. 将软件仓库映射到通用产品枚举示例配置

使用 **Clair** 配置中的 **repo2cpe_mapping_file** 和 **name2repos_mapping_file** 字段，使其包含 **CPE JSON** 映射文件。例如：

```
indexer:
 scanner:
   repo:
     rhel-repository-scanner:
       repo2cpe_mapping_file: /data/cpe-map.json
   package:
     rhel_containerscanner:
       name2repos_mapping_file: /data/repo-map.json
```

如需更多信息，请参阅如何准确匹配 **OVAL** 安全数据以安装 **RPM**。

第 9 章 CLAIR 配置概述

Clair 由一个结构化的 YAML 文件配置。每个 Clair 节点都需要指定它将在哪些模式下运行的模式，以及通过 CLI 标志或环境变量到配置文件的路径。例如：

```
$ clair -conf ./path/to/config.yaml -mode indexer
```

或者

```
$ clair -conf ./path/to/config.yaml -mode matcher
```

上述命令会使用相同的配置文件启动两个 Clair 节点。一个运行索引工具，另一个则运行匹配的设施。

如果您以 组合 模式运行 Clair，则必须在配置中提供索引器、匹配器和通知程序配置块。

9.1. 在代理环境中使用 CLAIR 的信息

如果需要，可以指定 Go 标准库相关的环境变量，例如：

- HTTP_PROXY

```
$ export HTTP_PROXY=http://<user_name>:<password>@<proxy_host>:<proxy_port>
```

- HTTPS_PROXY.

```
$ export HTTPS_PROXY=https://<user_name>:<password>@<proxy_host>:
<proxy_port>
```

- SSL_CERT_DIR

```
$ export SSL_CERT_DIR=/<path>/<to>/<ssl>/<certificates>
```

- NO_PROXY

```
$ export NO_PROXY=<comma_separated_list_of_hosts_and_domains>
```

-

如果您使用带有 **Clair** 更新器 **URL** 的环境中的代理服务器，您必须识别哪些 **URL** 需要添加到代理允许列表中，以确保 **Clair** 可以访问它们。例如： **osv updater** 需要访问 **https://osv-vulnerabilities.storage.googleapis.com** 来获取生态系统数据转储。在这种情况下，**URL** 必须添加到代理允许列表中。有关 **updater URL** 的完整列表，请参阅**"Clair updater URL"**。

您还必须确保将标准 **Clair URL** 添加到代理允许列表中：

- **https://search.maven.org/solrsearch/select**
- **https://catalog.redhat.com/api/containers/**
- **https://access.redhat.com/security/data/metrics/repository-to-cpe.json**
- **https://access.redhat.com/security/data/metrics/container-name-repos-map.json**

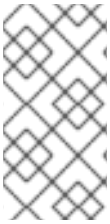
在配置代理服务器时，请考虑启用 **Clair** 和这些 **URL** 之间的无缝通信所需的任何身份验证要求或特定的代理设置。通过全面记录并解决这些注意事项，您可以确保在通过代理路由更新器流量的同时，有效处理 **Clair** 功能。

9.2. CLAIR 配置参考

以下 **YAML** 显示了一个 **Clair** 配置示例：

```
http_listen_addr: ""
introspection_addr: ""
log_level: ""
tls: {}
indexer:
  connstring: ""
  scanlock_retry: 0
  layer_scan_concurrency: 5
  migrations: false
  scanner: {}
  airgap: false
matcher:
  connstring: ""
  indexer_addr: ""
  migrations: false
  period: ""
```

```
    disable_updaters: false
    update_retention: 2
matchers:
    names: nil
    config: nil
updaters:
    sets: nil
    config: nil
notifier:
    connstring: ""
    migrations: false
    indexer_addr: ""
    matcher_addr: ""
    poll_interval: ""
    delivery_interval: ""
    disable_summary: false
    webhook: null
    amqp: null
    stomp: null
auth:
  psk: nil
trace:
    name: ""
    probability: null
    jaeger:
        agent:
            endpoint: ""
        collector:
            endpoint: ""
            username: null
            password: null
        service_name: ""
        tags: nil
        buffer_max: 0
metrics:
    name: ""
    prometheus:
        endpoint: null
    dogstatsd:
        url: ""
```



注意

为了完整，以上 **YAML** 文件列出了每个键。按照原样使用此配置文件将导致一些选项正常设置它们的默认值。

9.3. CLAIR 常规字段

下表描述了 **Clair** 部署可用的常规配置字段。

字段	Typhttp_listen_ae	描述
http_listen_addr	字符串	配置公开 HTTP API 的位置。 默认： **:6060**
introspection_addr	字符串	配置 Clair 的指标和健康端点公开的位置。
log_level	字符串	设置日志记录级别。需要以下字符串之一： **debug-color**,**debug**,**info**,**warn**,**error**,**fatal**,**panic**
tls	字符串	包含提供 TLS/SSL 和 HTTP/2 的 HTTP API 配置的映射。
.cert	字符串	要使用的 TLS 证书。必须是 full-chain 证书。

常规 **Clair** 字段的配置示例

以下示例显示了 **Clair** 配置。

常规 **Clair** 字段的配置示例

```
# ...
http_listen_addr: 0.0.0.0:6060
introspection_addr: 0.0.0.0:8089
log_level: info
# ...
```

9.4. CLAIR INDEXER 配置字段

下表描述了 **Clair** 的 **indexer** 组件的配置字段。

字段	类型	描述
indexer	对象	提供 Clair 索引节点配置。

字段	类型	描述
.airgap	布尔值	为 indexers 和 fetchers 禁用对互联网的 HTTP 访问。允许私有 IPv4 和 IPv6 地址。数据库连接不受影响。
.connstring	字符串	Postgres 连接字符串。接受格式作为 URL 或 libpq 连接字符串。
.index_report_request_concurrency	整数	速率限制索引报告创建请求的数量。把它设置为 **0**，以自动调整这个值。设置负值表示无限。自动大小是可用内核数的倍数。 如果超过并发，API 会返回 **429** 状态代码。
.scanlock_retry	整数	一个正整数，代表秒。并发索引器锁定在清单扫描上，以避免冲突。这个值调整等待索引器轮询锁定的频率。
.layer_scan_concurrency	整数	正整数限制并发层扫描的数量。Indexers 将同时匹配清单的层。这个值调整索引程序并行扫描的层数。
.migrations	布尔值	索引节点是否处理到其数据库的迁移。
.scanner	字符串	索引器配置。 扫描程序允许将配置选项传递给层扫描程序。如果设计这样做，扫描程序会将此配置传递给它。
.scanner.dist	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。
.scanner.package	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。
.scanner.repo	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。

indexer 配置示例

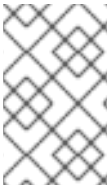
以下示例显示了 **Clair** 的 **hypothetical indexer** 配置。

indexer 配置示例

```
# ...
indexer:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
# ...
```

9.5. CLAIR MATCHER 配置字段

下表描述了 **Clair** 的 **matcher** 组件的配置字段。



注意

与 匹配器 配置字段不同。

字段	类型	描述
matcher	对象	提供 Clair 匹配节点配置。
.cache_age	字符串	控制提示用户缓存响应的时长。
.connstring	字符串	Postgres 连接字符串。接受格式作为 URL 或 libpq 连接字符串。
.max_conn_pool	整数	限制数据库连接池大小。 Clair 允许自定义连接池大小。这个数字直接设定同时允许的活跃数据库连接的数量。 以后的版本将忽略此参数。用户应该通过连接字符串进行配置。
.indexer_addr	字符串	匹配者联系索引程序来创建漏洞报告。需要此索引器的位置。 默认值为 **30m**。

字段	类型	描述
.migrations	布尔值	匹配节点是否处理到其数据库的迁移。
.period	字符串	决定新安全公告的更新频率。 默认值为 **30m**。
.disable_updaters	布尔值	是否运行后台更新。 Default: **False**
.update_retention	整数	设置在垃圾回收周期之间保留的更新操作数量。这应该根据数据库大小限制设置为安全 MAX 值。 默认值为 **10m**。 如果提供了小于 **0** 的值，则禁用垃圾回收。**2** 是最低值，可确保与通知进行比较更新。

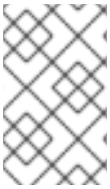
matcher 配置示例

matcher 配置示例

```
# ...
matcher:
  connstring: >-
    host=<DB_HOST> port=5432 dbname=<matcher> user=<DB_USER> password=D<B_PASS>
    sslmode=verify-ca sslcert=/etc/clair/ssl/cert.pem sslkey=/etc/clair/ssl/key.pem
    sslrootcert=/etc/clair/ssl/ca.pem
  indexer_addr: http://clair-v4/
  disable_updaters: false
  migrations: true
  period: 6h
  update_retention: 2
# ...
```

9.6. CLAIR MATCHERS 配置字段

下表描述了 **Clair** 的 **matchers** 组件的配置字段。



注意

与 匹配配置字段 不同。

表 **9.1. matchers** 配置字段

字段	类型	描述
matchers	字符串数组	为树内 匹配器提供配置。
.names	字符串	一个字符串值列表，用于告知匹配者工厂关于启用的匹配者信息。如果值设为 **null**，则默认匹配者列表将运行。以下字符串被接受：**alpine-matcher**,**aws-matcher**,**debian-matcher**,**gobin**,**java-maven**,**oracle**,**photon**,**python**, rhel ,**rhel -container-matcher**,**ruby**, suse ,**suse**,**ubuntu-matcher**
.config	字符串	为特定匹配器提供配置。 一个映射键是包含子对象的 matcher 名称键，该对象将提供给匹配者工厂构造器。例如：

matchers 配置示例

以下示例显示了一个假设的 **Clair** 部署，它只需要 **alpine,aws,debian,oracle matchers**。

matchers 配置示例

```
# ...
matchers:
  names:
  - "alpine-matcher"
  - "aws"
  - "debian"
  - "oracle"
# ...
```

9.7. CLAIR UPDATERS 配置字段

下表描述了 **Clair** 的 **updaters** 组件 的配置字段。

表 **9.2.** 更新器配置字段

字段	类型	描述
updaters	对象	为 matcher 的更新管理器提供配置。
.sets	字符串	一个值列表，告知更新管理器要运行的更新程序。 如果值设为 **null**，则默认更新程序将运行以下内容：**alpine,aws,clair.cvss,debian,oracle,photon,osv,rhel,rhcc suse,0.11.0-** 如果留空，则运行零更新程序。
.config	字符串	为特定更新器集提供配置。 由 updater 集名称的映射键包含子对象，该对象将提供给 updater 设置的构造器。有关每个 updater 的子对象列表，请参阅"高级更新器配置"。

updaters 配置示例

在以下配置中，仅配置 **rhel** 集。也定义了特定于 **rhel** 更新器的 **ignore_unpatched** 变量。

updaters 配置示例

```
# ...
updaters:
  sets:
    - rhel
  config:
    rhel:
      ignore_unpatched: false
# ...
```

9.8. CLAIR 通知程序配置字段

Clair 的一般通知程序配置字段如下。

字段	类型	描述
通知程序	对象	提供 Clair 通知程序节点配置。
.connstring	字符串	postgres 连接字符串。接受格式为 URL 或 libpq 连接字符串。
.migrations	布尔值	通知节点是否处理到其数据库的迁移。
.indexer_addr	字符串	通知程序联系一个索引程序，以创建或获取受漏洞影响的清单。需要此索引器的位置。
.matcher_addr	字符串	通知程序联系一个匹配程序，以列出更新操作并获得 diffs。需要此匹配器的位置。
.poll_interval	字符串	通知程序将查询匹配者更新操作的频率。
.delivery_interval	字符串	通知程序尝试发送创建或之前失败的通知的频率。
.disable_summary	布尔值	控制是否应将通知总结为每个清单中的一个。

通知程序配置示例

以下 通知程序 片断用于最小配置。

通知程序配置示例

```
# ...
notifier:
  connstring: >-
    host=DB_HOST port=5432 dbname=notifier user=DB_USER password=DB_PASS
    sslmode=verify-ca sslcert=/etc/clair/ssl/cert.pem sslkey=/etc/clair/ssl/key.pem
    sslrootcert=/etc/clair/ssl/ca.pem
  indexer_addr: http://clair-v4/
  matcher_addr: http://clair-v4/
```

```
  delivery_interval: 5s
  migrations: true
  poll_interval: 15s
  webhook:
    target: "http://webhook/"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    headers: ""
  amqp: null
  stomp: null
# ...
```

9.8.1. Clair Webhook 配置字段

以下 **Webhook** 字段可用于 **Clair** 通知程序环境。

表 **9.3. Clair Webhook** 字段

.webhook	对象	配置用于 Webhook 发送的通知程序。
.webhook.target	字符串	提供 webhook 的 URL。
.webhook.callback	字符串	可以检索通知的回调 URL。通知 ID 将附加到此 URL 中。 这通常是托管 Clair 通知程序的位置。
.webhook.headers	字符串	将标头名称与值列表关联的映射。

Webhook 配置示例

Webhook 配置示例

```
# ...
notifier:
# ...
  webhook:
    target: "http://webhook/"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
# ...
```

9.8.2. Clair amqp 配置字段

以下高级消息队列协议(AMQP)字段可用于 Clair 通知程序环境。

.amqp	对象	配置用于 AMQP 发送的通知程序。 [NOTE] ==== Clair 不自行声明任何 AMQP 组件。所有尝试使用交换或队列的尝试都是仅被动的，并将失败。代理管理员应提前设置交换和队列。===
.amqp.direct	布尔值	如果为 **true**，则通知程序会将单个通知（而非回调）发送到配置的 AMQP 代理。
.amqp.rollup	整数	当将 **amqp.direct** 设置为 **true** 时，这个值告知通知要直接发送多少通知。例如，如果 **direct** 设置为 **true**，并且 **amqp.rollup** 设置为 **5**，则通知程序不会在单个 JSON 有效负载向代理中提供 5 个通知。将值设为 **0** 实际的效果是将其设置为 **1**。
.amqp.exchange	对象	要连接的 AMQP 交换。
.amqp.exchange.name	字符串	要连接的交换的名称。
.amqp.exchange.type	字符串	交换的类型。通常，以下之一：**直接、发出、主题、标题**.
.amqp.exchange.durability	布尔值	配置的队列是否持久。
.amqp.exchange.auto_delete	布尔值	配置的队列是否使用 **auto_delete_policy**。
.amqp.routing_key	字符串	每个通知发送的路由密钥的名称。
.amqp.callback	字符串	如果 **amqp.direct** 设置为 **false**，则会在发送到代理的通知回调中提供此 URL。此 URL 应指向 Clair 的通知 API 端点。

.amqp.uris	字符串	要连接的一个或多个 AMQP 代理的列表，按优先级顺序连接。
.amqp.tls	对象	配置 TLS/SSL 连接到 AMQP 代理。
.amqp.tls.root_ca	字符串	可以读取 root CA 的文件系统路径。
.amqp.tls.cert	字符串	可读取 TLS/SSL 证书的文件系统路径。 [NOTE] ==== Clair 还允许 **SSL_CERT_DIR**，如 Go **crypto/x509** 软件包所记录。 ====
.amqp.tls.key	字符串	可以读取 TLS/SSL 私钥的文件系统路径。

AMQP 配置示例

以下示例显示了 **Clair** 的 **hypothetical AMQP** 配置。

AMQP 配置示例

```
# ...
notifier:
# ...
  amqp:
    exchange:
       name: ""
       type: "direct"
       durable: true
       auto_delete: false
    uris: ["amqp://user:pass@host:10000/vhost"]
    direct: false
    routing_key: "notifications"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    tls:
     root_ca: "optional/path/to/rootca"
     cert: "madatory/path/to/cert"
     key: "madatory/path/to/key"
# ...
```

9.8.3. Clair STOMP 配置字段

Clair 通知程序环境提供了以下简单文本导向型消息协议**(STOMP)**字段。

.stomp	对象	为 STOMP 交付配置通知。
.stomp.direct	布尔值	如果为 **true**，则通知程序会向配置的 STOMP 代理提供单独的通知（而非回调）。
.stomp.rollup	整数	如果 **stomp.direct** 设为 **true**，则该值限制在单个直接发送中发送的通知数量。例如，如果 **direct** 设为 **true**，并且 **滚动** 设置为 **5**，则通知程序不会在单个 JSON 有效负载中向代理发送 5 个通知。将值设为 **0** 实际的效果是将其设置为 **1**。
.stomp.callback	字符串	如果 **stomp.callback** 设置为 **false**，则通知回调中提供的 URL 将发送到代理。此 URL 应指向 Clair 的通知 API 端点。
.stomp.destination	字符串	将通知发送到的 STOMP 目的地。
.stomp.uris	字符串	要以优先级顺序连接到的一个或多个 STOMP 代理的列表。
.stomp.tls	对象	配置 TLS/SSL 连接到 STOMP 代理。
.stomp.tls.root_ca	字符串	可以读取 root CA 的文件系统路径。 [NOTE] ==== Clair 还尊重 **SSL_CERT_DIR**，如 Go **crypto/x509** 软件包所记录。 ====
.stomp.tls.cert	字符串	可读取 TLS/SSL 证书的文件系统路径。
.stomp.tls.key	字符串	可以读取 TLS/SSL 私钥的文件系统路径。
.stomp.user	字符串	配置 STOMP 代理的登录详情。
.stomp.user.login	字符串	要连接的 STOMP 登录。

.stomp	对象	为 STOMP 交付配置通知。
.stomp.user.passcode	字符串	要连接的 STOMP passcode。

STOMP 配置示例

以下示例显示了 **Clair** 的 **hypothetical STOMP** 配置。

STOMP 配置示例

```
# ...
notifier:
# ...
  stomp:
    desitnation: "notifications"
    direct: false
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    login:
      login: "username"
      passcode: "passcode"
    tls:
     root_ca: "optional/path/to/rootca"
     cert: "madatory/path/to/cert"
     key: "madatory/path/to/key"
# ...
```

9.9. CLAIR 授权配置字段

以下授权配置字段可用于 **Clair**。

字段	类型	描述
auth	对象	定义 Clair 的外部和基于服务 JWT 的身份验证。如果定义了多个 **auth** 机制，Clair 会选择一个。目前，不支持多个机制。
.psk	字符串	定义预共享密钥身份验证。
.psk.key	字符串	在所有方签名和验证 JWT 之间的共享 base64 编码密钥。

字段	类型	描述
.psk.iss	字符串	要验证的 JWT 签发者列表。空列表接受 JWT 声明中的任何签发者。

授权配置示例

以下 授权 片断用于最小配置。

授权配置示例

```
# ...
auth:
  psk:
    key: MTU5YzA4Y2ZkNzJoMQ== 1
    iss: ["quay"]
# ...
```

9.10. CLAIR TRACE 配置字段

以下 **trace** 配置字段可用于 **Clair**。

字段	类型	描述
trace	对象	基于 OpenTelemetry 定义分布式追踪配置。
.name	字符串	应用程序跟踪的名称将属于。
.probability	整数	可能会出现 trace 的概率。
.jaeger	对象	为 Jaeger tracing 定义值。
.jaeger.agent	对象	定义用于配置发送到 Jaeger 代理的值。
.jaeger.agent.endpoint	字符串	可提交 trace 的 & **lt;host>:<post** > 语法中的地址。

字段	类型	描述
.jaeger.collector	对象	为配置发送到 Jaeger 收集器定义值。
.jaeger.collector.endpoint	字符串	可提交 trace 的 & **lt;host>:<post** > 语法中的地址。
.jaeger.collector.username	字符串	Jaeger 用户名。
.jaeger.collector.password	字符串	Jaeger 密码。
.jaeger.service_name	字符串	在 Jaeger 中注册的服务名称。
.jaeger.tags	字符串	用于提供额外的元数据的键值对。
.jaeger.buffer_max	整数	在发送到 Jaeger 后端以进行存储和分析前可以缓冲的最大 span 数量。

trace 配置示例

以下示例显示了 **Clair** 的 **hypothetical trace** 配置。

trace 配置示例

```
# ...
trace:
  name: "jaeger"
  probability: 1
  jaeger:
    agent:
      endpoint: "localhost:6831"
    service_name: "clair"
# ...
```

9.11. CLAIR 指标配置字段

以下指标配置字段可用于 **Clair**。

字段	类型	描述
metrics	对象	基于 OpenTelemetry 定义分布式追踪配置。
.name	字符串	使用的指标的名称。
.prometheus	字符串	配置 Prometheus 指标导出器。
.prometheus.endpoint	字符串	定义提供指标的路径。

指标配置示例

以下示例显示了 **Clair** 的 **hypothetical** 指标配置。

指标配置示例

```
# ...
metrics:
  name: "prometheus"
  prometheus:
    endpoint: "/metricsz"
# ...
```