Red Hat Enterprise Linux 9

安全强化

增强 Red Hat Enterprise Linux 9 系统的安全性

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 安全强化

增强 Red Hat Enterprise Linux 9 系统的安全性

法律通告

摘要

了解保护 Red Hat Enterprise Linux 服务器和工作站免受本地和远程入侵、利用和恶意活动的流程和实践。通过使用这些方法和工具，您可以为数据中心、工作场所和家庭创建更安全的计算环境。

目录

对红帽文档提供反馈 6

第 1 章 在安装过程中保护 RHEL **7**
1.1. BIOS 和 UEFI 安全 7
1.2. 磁盘分区 7
1.3. 在安装过程中限制网络连接 8
1.4. 安装所需的最少软件包 8
1.5. 安装后流程 8

第 2 章 在 FIPS 模式中安装系统 **10**
2.1. FEDERAL INFORMATION PROCESSING STANDARDS 140 和 FIPS 模式 10
2.2. 安装启用了 FIPS 模式的系统 11
2.3. 其他资源 12

第 3 章 使用系统范围的加密策略 **13**
3.1. 系统范围的加密策略 13
3.2. 将系统范围的加密策略切换到与早期版本兼容的模式 16
3.3. 在 WEB 控制台中设置系统范围的加密策略 16
3.4. 将系统切换到 FIPS 模式 18
3.5. 在容器中启用 FIPS 模式 19
3.6. 使用与 FIPS 140-3 不兼容的加密的 RHEL 应用程序列表 20
3.7. 将应用程序从下列系统范围的加密策略中排除 22
3.8. 使用子策略自定义系统范围的加密策略 23
3.9. 重新启用 SHA-1 24
3.10. 创建并设置自定义系统范围的加密策略 25

第 4 章 使用 RHEL 系统角色设置自定义加密策略 **27**
4.1. 使用 CRYPTO_POLICIES RHEL 系统角色设置自定义加密策略 27

第 5 章 通过 PKCS#11 将应用程序配置为使用加密硬件 **29**
5.1. 通过 PKCS #11 的加密硬件支持 29
5.2. 使用保存在智能卡中的 SSH 密钥 29
5.3. 配置应用程序以使用智能卡上的证书进行身份验证 30
5.4. 在 APACHE 中使用 HSM 保护私钥 31
5.5. 使用 HSM 保护 NGINX 中的私钥 31
5.6. 其他资源 32

第 6 章 使用 POLKIT 控制对智能卡的访问 **33**
6.1. 通过 POLKIT 的智能卡访问控制 33
6.2. 排除与 PC/SC 和 POLKIT 相关的问题 33
6.3. 向 PC/SC 显示关于 POLKIT 授权的更多详细信息 34
6.4. 其他资源 35

第 7 章 扫描系统以了解配置合规性和漏洞 **36**
7.1. RHEL 中的配置合规工具 36
7.2. 漏洞扫描 36
7.3. 配置合规性扫描 39
7.4. 修复系统，使其与特定基准一致 42
7.5. 使用 SSG ANSIBLE PLAYBOOK 修复系统以与特定基准保持一致 42
7.6. 创建修复 ANSIBLE PLAYBOOK，使系统与特定的基准一致 43
7.7. 为后续应用程序创建补救 BASH 脚本 44
7.8. 使用 SCAP WORKBENCH 用自定义配置文件扫描系统 45
7.9. 安装后立即部署符合安全配置文件的系统 49

7.10. 扫描容器和容器镜像以查找漏洞 52
7.11. 使用特定基准评估容器或容器镜像的安全性合规 52
7.12. RHEL 9 支持 SCAP 安全指南配置集 53
7.13. 其他资源 61

第 8 章 使用 KEYLIME 确保系统完整性 63
8.1. KEYLIME 的工作原理 63
8.2. 配置 KEYLIME 验证器 65
8.3. 将 KEYLIME 验证器配置为容器 68
8.4. 配置 KEYLIME 注册器 70
8.5. 将 KEYLIME 注册器配置为容器 72
8.6. 使用系统角色设置 KEYLIME 服务器 75
8.7. KEYLIME_SERVER RHEL 系统角色的变量 76
8.8. 配置 KEYLIME 租户 77
8.9. 配置 KEYLIME 代理 80
8.10. 部署对运行时监控的 KEYLIME 83
8.11. 在测试时为测量的引导部署主精简 86
8.12. KEYLIME 环境变量 88

第 9 章 使用 AIDE 检查完整性 100
9.1. 安装 AIDE 100
9.2. 使用 AIDE 执行完整性检查 100
9.3. 更新 AIDE 数据库 101
9.4. 文件完整性工具：AIDE 和 IMA 101
9.5. 其他资源 101

第 10 章 使用 LUKS 加密块设备 103
10.1. LUKS 磁盘加密 103
10.2. RHEL 中的 LUKS 版本 104
10.3. LUKS2 重新加密过程中数据保护选项 104
10.4. 使用 LUKS2 加密块设备上的现有数据 105
10.5. 使用带有分离标头的 LUKS2 在块设备上加密现有数据 107
10.6. 使用 LUKS2 加密空白块设备 109
10.7. 使用 STORAGE RHEL 系统角色创建一个 LUKS2 加密的卷 111

第 11 章 使用基于策略的解密配置加密卷的自动解锁 113
11.1. 网络绑定磁盘加密 113
11.2. 安装加密客户端 - CLEVIS 114
11.3. 部署 SELINUX 处于 ENFORCING 模式的 TANG 服务器 115
11.4. 轮转 TANG 服务器密钥并更新客户端上的绑定 116
11.5. 在 WEB 控制台中使用 TANG 密钥配置自动解锁 118
11.6. 基本的 NBDE 和 TPM2 加密客户端操作 121
11.7. 配置 LUKS 加密卷的手动注册 122
11.8. 使用 TPM 2.0 策略配置 LUKS 加密的卷的手动注册 124
11.9. 手动从 LUKS 加密卷中删除 CLEVIS PIN 126
11.10. 使用 KICKSTART 配置 LUKS 加密的卷的自动注册 127
11.11. 配置 LUKS 加密的可移动存储设备的自动解锁 129
11.12. 部署高可用性 NBDE 系统 129
11.13. NBDE 网络中虚拟机的部署 130
11.14. 使用 NBDE 为云环境构建可自动注册的虚拟机镜像 131
11.15. 将 TANG 部署为容器 131
11.16. NBDE_CLIENT 和 NBDE_SERVER RHEL 系统角色(CLEVIS 和 TANG)简介 132
11.17. 使用 NBDE_SERVER RHEL 系统角色设置多个 TANG 服务器 133
11.18. 使用 NBDE_CLIENT RHEL 系统角色设置多个 CLEVIS 客户端 134

第 12 章 审计系统 137
12.1. LINUX 审计 137
12.2. 审计系统架构 138
12.3. 为安全环境配置 AUDITD 138
12.4. 启动和控制 AUDITD 139
12.5. 了解审计日志文件 140
12.6. 使用 AUDITCTL 来定义和执行审计规则 144
12.7. 定义持久性审计规则 145
12.8. 符合标准的预配置的审计规则文件 145
12.9. 使用 AUGENRULES 来定义持久性规则 146
12.10. 禁用 AUGENRULES 146
12.11. 设置审计来监控软件更新 147
12.12. 使用审计监控用户登录时间 149
12.13. 其他资源 150

第 13 章 使用 FAPOLICYD 阻止和允许应用程序 151
13.1. FAPOLICYD 简介 151
13.2. 部署 FAPOLICYD 152
13.3. 使用其它信任源将文件标记为可信 153
13.4. 为 FAPOLICYD 添加自定义 ALLOW 和 DENY 规则 154
13.5. 启用 FAPOLICYD 完整性检查 157
13.6. 故障排除与 FAPOLICYD 相关的问题 158
13.7. 使用 FAPOLICYD RHEL 系统角色配置防止未知代码的执行 160
13.8. 其他资源 161

第 14 章 保护系统免受入侵 USB 设备的攻击 162
14.1. USBGUARD 162
14.2. 安装 USBGUARD 162
14.3. 使用 CLI 阻止和授权 USB 设备 163
14.4. 永久阻止和授权 USB 设备 164
14.5. 为 USB 设备创建自定义策略 165
14.6. 为 USB 设备创建结构化自定义策略 166
14.7. 授权用户和组使用 USBGUARD IPC 接口 168
14.8. 将 USBGUARD 授权事件记录到 LINUX 审计日志中 168
14.9. 其他资源 169

第 15 章 配置远程日志记录解决方案 170
15.1. RSYSLOG 日志记录服务 170
15.2. 安装 RSYSLOG 文档 170
15.3. 通过 TCP 配置服务器进行远程记录 170
15.4. 通过 TCP 配置远程日志记录到服务器 172
15.5. 配置 TLS 加密的远程日志记录 174
15.6. 配置服务器以通过 UDP 接收远程日志信息 179
15.7. 通过 UDP 配置远程日志记录到服务器 182
15.8. RSYSLOG 中的负载均衡帮助程序 184
15.9. 配置可靠的远程日志记录 184
15.10. 支持的 RSYSLOG 模块 187
15.11. 配置 NETCONSOLE 服务为将内核信息记录到远程主机 187
15.12. 其他资源 188

第 16 章 使用 LOGGING 系统角色 189
16.1. LOGGING RHEL 系统角色 189
16.2. 应用本地 LOGGING RHEL 系统角色 190
16.3. 使用本地 LOGGING RHEL 系统角色过滤日志 192

16.4. 使用 LOGGING RHEL 系统角色应用一个远程日志解决方案 194
16.5. 使用带有 TLS 的 LOGGING RHEL 系统角色 197
16.6. 使用带有 RELP 的 LOGGING RHEL 系统角色 204
16.7. 其他资源 211

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 点顶部导航栏中的 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 在安装过程中保护 RHEL

安全性甚至在您开始安装 Red Hat Enterprise Linux 之前就已经开始了。从一开始就安全地配置系统可以使以后更容易实施其他安全设置。

1.1. BIOS 和 UEFI 安全

对 BIOS（或与 BIOS 等效的）和引导加载程序的密码保护可防止具有系统物理访问权限的未授权用户使用可移动介质引导，或通过单用户模式获得 root 权限。为防止此类攻击，您应该根据有关工作站和机器位置的敏感性来采取安全措施。

例如，如果机器是在交易展示中使用并且不包含敏感信息，那么防止此类攻击可能并不重要。但是，如果带有公司网络的私有的、未加密的 SSH 密钥的员工的笔记本在同一个展会上无人看管，则可能会导致重大的安全漏洞，对整个公司造成影响。

但是，如果工作站位于只有授权的或可信任的人员才有权访问的地方，则可能不需要保护 BIOS 或引导加载程序。

1.1.1. BIOS 密码

密码保护计算机 BIOS 的两个主要原因是[1]:

防止更改 BIOS 设置

如果入侵者可以访问 BIOS，则他们可以将其设置为从 CD-ROM 或闪存驱动器引导。这使得他们能够进入救援模式或单用户模式，从而使他们可以在系统上启动任意进程或复制敏感数据。

防止系统引导

某些 BIOS 允许引导过程的密码保护。激活后，攻击者必须在 BIOS 启动引导加载程序前输入密码。

由于设置 BIOS 密码的方法因计算机制造商而异，因此请查阅计算机手册了解具体说明。

如果您忘记 BIOS 密码，可以通过主板上的跳线来重置，也可以通过断开 CMOS 电池来重置。因此，如果可能的话，最好锁好计算机机箱。但是，在尝试断开 CMOS 电池之前，请查阅计算机或主板的手册。

1.1.2. 非基于 BIOS 的系统安全性

其他系统和架构使用不同的程序来执行大致相当于 x86 系统上 BIOS 的低级别任务。*例如，统一可扩展固件接口* (*UEFI*)shell。

有关密码保护类似 BIOS 程序的说明，请查看制造商的说明。

1.2. 磁盘分区

对于裸机上安装以及支持调整虚拟磁盘硬件和包含已安装了操作系统的文件系统的虚拟化或云环境，推荐的磁盘分区实践有所不同。

为了确保在 **裸机安装** 上隔离和保护数据，请为 **/boot**、**/**、**/home**、**/tmp** 和 **/var/tmp/** 目录创建单独的分区：

/boot

这个分区是系统在启动过程中读取的第一个分区。引导装载程序和用于将系统引导到 Red Hat Enterprise Linux 9 的内核镜像存储在这个分区中。此分区不应加密。如果此分区包含在 / 中，并且该分区已加密或者不可用，那么您的系统将无法引导。

/home

当用户数据（**/home**）存储在 / 而不是独立分区中时，分区可能会填满，从而导致操作系统不稳定。另外，当将您的系统升级到 Red Hat Enterprise Linux 9 的下一版本时，当您可以把数据保存在 **/home** 分区时，因为在安装过程中不会覆盖它，这更为容易。如果 root 分区(/)损坏，则您的数据将永久丢失。通过使用单独的分区，对数据丢失有稍微多一点的保护。您还可以将此分区作为频繁备份的目标。

/tmp 和 /var/tmp/。

/tmp 和 **/var/tmp/** 目录都是用来存储不需要长期存储的数据。但是，如果大量数据填充了其中一个目录，则它可能会消耗掉您的所有存储空间。如果发生这种情况，且这些目录存储在 / 中，则您的系统可能会变得不稳定并崩溃。因此，将这些目录移到它们自己的分区中是一个不错的想法。

对于 **虚拟机或云实例**，单独的 **/boot**、**/home**、**/tmp** 和 **/var/tmp** 分区是可选的，因为如果开始填满了，您可以增加虚拟磁盘大小和 / 分区。设置监控，以定期检查 / 分区使用情况，以便在相应增加虚拟磁盘大小之前，其不会填满。

注意

在安装过程中，您可以选择加密分区。您必须提供密码短语。此密语充当解锁批量加密密钥的密钥，该密钥用于保护分区的数据。

1.3. 在安装过程中限制网络连接

安装 Red Hat Enterprise Linux 9 时，安装介质代表系统在特定时间的快照。因此，它可能没有最新的安全修复程序，并且可能容易受到某些问题的攻击，这些问题是在安装介质提供的系统发布后才修复的。

安装有潜在漏洞的操作系统时，始终将暴露限制在最近的必要网络区内。最安全的选择是"无网络"区，这意味着在安装过程中使计算机断开连接。在某些情况下，LAN 或内部网连接就足够了，而互联网连接的风险最大。要遵循最佳安全实践，请从网络安装 Red Hat Enterprise Linux 9 时，选择与您的软件仓库最接近的区域。

1.4. 安装所需的最少软件包

最好只安装您要使用的软件包，因为计算机上的每一款软件都可能包含漏洞。如果您要从 DVD 介质安装，请仔细选择要在安装过程中安装的软件包。如果您发现需要其他软件包，您可在以后将其添加到系统中。

1.5. 安装后流程

以下步骤是安装 Red Hat Enterprise Linux 9 后应立即执行的安全相关步骤。

- 更新您的系统。以 root 用户身份输入以下命令：

```
# dnf update
```

- 虽然在安装 Red Hat Enterprise Linux 时会自动启用防火墙服务 **firewalld**，但它可以被明确禁用，例如，在 Kickstart 配置中。在这种情况下，请重新启用防火墙。
  要启动 **firewalld**，请以 root 用户身份输入以下命令：

```
# systemctl start firewalld
# systemctl enable firewalld
```

- 要提高安全性，请禁用您不需要的服务。例如，如果您的计算机上没有安装打印机，请使用以下命令禁用 **cups** 服务：

```
# systemctl disable cups
```

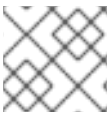
要查看活动状态的服务，请输入以下命令：

```
$ systemctl list-units | grep service
```

[1] 因为制造商之间的系统 BIOS 有所不同，因此有些制造商可能不支持这两种类型的密码保护，而另一些制造商可能支持一种类型，但不支持另一种类型。

第 2 章 在 FIPS 模式中安装系统

要启用联邦信息处理标准(FIPS) 140-3 强制的加密模块，您必须在 FIPS 模式下操作 RHEL 9。如果您的目标是遵循FIPS，在 FIPS 模式下开始安装是推荐的方法。



注意

RHEL 9 的加密模块尚未获得 FIPS 140-3 要求的认证。

2.1. FEDERAL INFORMATION PROCESSING STANDARDS 140 和 FIPS 模式

联邦信息处理标准(FIPS)公共 140 是国家标准与技术研究所开发的一系列计算机安全标准，以确保加密模块的质量。FIPS 140 标准可确保加密工具正确实现其算法。运行时加密算法和完整性自我测试是一些机制，用于确保系统使用符合标准要求的加密机制。

要确保 RHEL 系统生成并使用所有带有 FIPS 批准的算法的加密密钥，您必须将 RHEL 切换到 FIPS 模式。

您可以使用以下方法之一启用 FIPS 模式：

- 在 FIPS 模式下开始安装
- 安装后将系统切换到 FIPS 模式

如果您的目的是 FIPS 合规性，请在 FIPS 模式下开始安装。这可避免加密密钥材料重新生成和与转换已部署的系统相关的结果系统的合规性的重新评估。

要操作 FIPS 兼容系统，请在 FIPS 模式下创建所有加密密钥资料。另外，加密密钥材料不得离开 FIPS 环境，除非它被安全封装，且永远不会在非 FIPS 环境下解封。

使用 **fips-mode-setup** 工具将系统切换到 FIPS 模式不能保证符合 FIPS 140 标准。将系统设置为 FIPS 模式后，可能无法重新生成所有加密密钥。例如，在具有用户加密密钥的现有 IdM 域中，您无法重新生成所有密钥。如果您无法在 FIPS 模式下开始安装，在安装后，请始终启用 FIPS 模式作为第一步，然后再进行安装后配置步骤或安装任何工作负载。

fips-mode-setup 工具也在内部使用 **FIPS** 系统范围的加密策略。但是，在 **update-crypto-policies --set FIPS** 命令所执行的操作之上，**fips-mode-setup** 使用 **fips-finish-install** 工具来确保 FIPS dracut 模块的安装，它还在内核命令行中添加 **fips=1** 引导选项，并重新生成初始 RAM 磁盘。

另外，在 FIPS 模式下强制所需的限制取决于 **/proc/sys/crypto/fips_enabled** 文件的内容。如果文件包含 **1**，RHEL 核心加密组件切换到模式，在此模式中，它们只使用加密算法的 FIPS 批准的实现。如果 **/proc/sys/crypto/fips_enabled** 包含 **0**，则加密组件不会启用其 FIPS 模式。

FIPS 系统范围的加密策略有助于配置更高级别的限制。因此，支持加密灵活性的通信协议不会宣布系统在选择时拒绝的密码。例如，ChaCha20 算法不是 FIPS 批准的，并且 **FIPS** 加密策略确保 TLS 服务器和客户端不宣布 TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 TLS 密码套件，因为任何尝试使用此类密码都会失败。

如果您在 FIPS 模式下操作 RHEL，并使用提供其自己的与 FIPS 模式相关的配置选项的应用程序，则可以忽略这些选项以及相应的应用程序指导。在 FIPS 模式下运行的系统，系统范围的加密策略只强制执行与 FIPS 兼容的加密。例如，如果系统在 FIPS 模式下运行，则忽略 Node.js 配置选项 **--enable-fips**。如果您在没有对在 FIPS 模式下运行的系统使用 **--enable-fips** 选项，则不符合 FIPS-140 合规性要求。

注意

RHEL 9 的加密模块尚未获得美国国家标准与技术研究院(NIST)加密模块验证计划(CMVP)的 FIPS 140-3 要求认证。您可以在 合规活动和政府标准 知识库文章的 FIPS 140-2 和 FIPS 140-3 部分中看到加密模块的验证状态。

警告

在 FIPS 模式下运行的 RHEL 9.2 及更新的版本强制任何 TLS 1.2 连接都必须使用 Extended Master Secret (EMS)扩展(RFC 7627)，因为 FIPS 140-3 标准需要。因此，不支持 EMS 或 TLS 1.3 的旧客户端无法连接到在 FIPS 模式下运行的 RHEL 9 服务器，FIPS 模式下的 RHEL 9 客户端无法连接到只支持没有 EMS 的 TLS 1.2 的服务器。请参阅 使用 Red Hat Enterprise Linux 9.2 强制执行的 TLS 扩展 "Extended Master Secret"

其他资源

- 合规活动和政府标准 知识库文章中的 FIPS 140-2 和 FIPS 140-3 部分 。
- RHEL 系统范围的加密策略
- NIST 计算机安全资源中心上的 FIPS 出版物 .
- 联邦信息处理标准发布：FIPS 140-3

2.2. 安装启用了 FIPS 模式的系统

要启用联邦信息处理标准(FIPS) 140 强制的加密模块自我检查，请在系统安装过程中启用 FIPS 模式。

重要

仅在 RHEL 安装过程中启用 FIPS 模式可确保系统使用 FIPS 批准的算法生成所有密钥，并持续监控测试。

警告

完成 FIPS 模式的设置后，您无法在不将系统置于不一致状态的情况下关闭 FIPS 模式。如果您的场景需要这个更改，则唯一的正确方法是完全重新安装系统。

步骤

1. 在系统安装过程中，将 **fips=1** 选项添加到内核命令行。
2. 在软件选择阶段，请勿安装任何第三方软件。
3. 安装后，系统会自动以 FIPS 模式启动。

验证

- 系统启动后，检查是否启用了 FIPS 模式：

```
$ fips-mode-setup --check
FIPS mode is enabled.
```

其他资源

- RHEL 安装程序的引导选项文档中的 编辑引导选项 部分

2.3. 其他资源

- 将系统切换到 FIPS 模式
- 在容器中启用 FIPS 模式
- 使用与 FIPS 140-3 不兼容的加密的 RHEL 9 应用程序列表

第 3 章 使用系统范围的加密策略

系统范围的加密策略是一个系统组件，它配置核心加密子系统，包括 TLS、IPsec、SSH、DNSSec 和 Kerberos 协议。它提供了一小组策略，管理员可以选择这些策略。

3.1. 系统范围的加密策略

设置系统范围的策略时，RHEL 中的应用程序会遵守它，并拒绝使用不符合该策略的算法和协议，除非您明确要求应用程序这样做。也就是说，在运行系统提供的配置时，策略适用于应用程序的默认行为，但在需要时您可以覆盖它。

RHEL 9 包含以下预定义的策略：

DEFAULT

默认的系统范围加密策略级别为当前威胁模型提供了安全设置。它允许 TLS 1.2 和 1.3 协议，以及 IKEv2 和 SSH2 协议。如果 RSA 密钥和 Diffie-Hellman 参数至少是 2048 位，则可以接受它们。

LEGACY

确保与 Red Hat Enterprise Linux 6 及更早版本的最大兼容性；由于攻击面增加而不太安全。SHA-1 允许用作 TLS 哈希、签名和算法。CBC-mode 密码可以和 SSH 一起使用。使用 GnuTLS 的应用程序允许使用 SHA-1 签名的证书。它允许 TLS 1.2 和 1.3 协议，以及 IKEv2 和 SSH2 协议。如果 RSA 密钥和 Diffie-Hellman 参数至少是 2048 位，则可以接受它们。

FUTURE

更严格的前瞻性安全级别，旨在测试未来可能的策略。此策略不允许在 DNSSec 或 HMAC 中使用 SHA-1。SHA2-224 和 SHA3-224 哈希被拒绝。禁用 128 位密码。CBC-mode 密码被禁用，除了 Kerberos 中。它允许 TLS 1.2 和 1.3 协议，以及 IKEv2 和 SSH2 协议。如果 RSA 密钥和 Diffie-Hellman 参数至少是 3072 位，则可以接受它们。如果您的系统在公共互联网上进行通信，您可能会遇到互操作性问题。

FIPS

符合 FIPS 140 要求。**fips-mode-setup** 工具将 RHEL 系统切换到 FIPS 模式，在内部使用此策略。切换到 **FIPS** 策略不能保证符合 FIPS 140 标准。在将系统设置为 FIPS 模式后，还必须重新生成所有加密密钥。这在很多情况下是不可能的。
RHEL 还提供 **FIPS:OSPP** 系统范围的子策略，其中包含对通用标准(CC)认证所需的加密算法的进一步限制。设置了此子策略后，系统的互操作性会降低。例如，您无法使用比 3072 位少的 RSA 和 DH 密钥、其它的 SSH 算法和几个 TLS 组。设置 **FIPS:OSPP** 也会阻止连接到 Red Hat Content Delivery Network (CDN)结构。另外，您无法将活动目录(AD)集成到使用 **FIPS:OSPP**的 IdM 部署中，使用 **FIPS:OSPP** 的 RHEL 主机和 AD 域之间的通信可能无法工作，或者某些 AD 帐户可能无法进行身份验证。

请注意，在设置了 **FIPS:OSPP** 加密子策略后，您的系统不符合 CC 。使 RHEL 系统符合 CC 标准的唯一正确方法是安装 **cc-config** 软件包。有关已认证的 RHEL 版本的列表、验证报告以及 CC 指南的链接，请参阅合规活动和政府标准知识库文章中的 通用标准 部分。

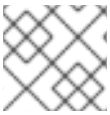
红帽不断调整所有策略级别，以便所有库都提供安全的默认值，但使用 **LEGACY** 策略时除外。虽然 **LEGACY** 配置文件不提供安全默认值，但它不包括任何易被利用的算法。因此，在 Red Hat Enterprise Linux 生命周期内，任何所提供的策略中启用的算法或可接受的密钥大小可能会发生变化。

此变更反映了新的安全标准和新的安全研究。如果您必须确保 Red Hat Enterprise Linux 的整个生命周期内与特定系统的互操作性，对于与该系统交互的组件，您应该选择不使用系统范围的加密策略，或使用自定义加密策略重新启用特定的算法。

重要

因为客户门户网站 API 中的证书使用的加密密钥不满足 **FUTURE** 系统范围的加密策略的要求，所以 **redhat-support-tool** 程序目前无法使用这个策略级别。

要临时解决这个问题，在连接到客户门户网站 API 时使用 **DEFAULT** 加密策略。



注意

只有在应用程序支持它们时，策略级别中描述为允许的特定算法和密码才可用。

管理加密策略的工具
要查看或更改当前系统范围的加密策略，请使用 **update-crypto-policies** 工具，例如：

```
$ update-crypto-policies --show
DEFAULT
# update-crypto-policies --set FUTURE
Setting system policy to FUTURE
```

要确保应用了加密策略的修改，请重启系统。

强**大的加密默**认删**除不安全的密**码**套件和**协议
以下列表包含从 RHEL 中核心加密库中删除的密码套件和协议。由于它们没有在源中存在，或者其支持在构建过程中被禁用了，所以应用程序无法使用它们。

- DES（自 RHEL 7 开始）
- 所有导出评级密码套件（自 RHEL 7 开始）
- 签名中的 MD5（自 RHEL 7 开始）
- SSLv2（自 RHEL 7 开始）
- SSLv3（自 RHEL 8 开始）
- 所有 ECC 曲线 < 224 位（自 RHEL 6 开始）
- 所有二进制字段 ECC curves（自 RHEL 6 开始）

在所有策略级别**禁用算法**
以下算法在 RHEL 9 中包括的 **LEGACY**、**DEFAULT**、**FUTURE** 和 **FIPS** 加密策略中被禁用。只能通过应用一个自定义加密策略或一个明确的单个应用程序的配置来启用它们，但生成的配置在 生产支持覆盖范围之外。

- 早于版本 1.2 的 TLS （自 RHEL 9 开始，在 RHEL 8 中为 < 1.0）
- 早于 版本 1.2 的 DTLS （自 RHEL 9 开始，在 RHEL 8 中为 < 1.0）
- DH 的参数 < 2048 位（自 RHEL 9 开始，在 RHEL 8 中是 < 1024 位）
- RSA 的密钥大小 < 2048 位（自 RHEL 9 开始，在 RHEL 8 中是 < 1024 位）
- DSA（自 RHEL 9 开始，在 RHEL 8 中是 < 1024 位）
- 3DES（自 RHEL 9 开始）

- RC4（自 RHEL 9 开始）
- FFDHE-1024 (自 RHEL 9 开始)
- RbacConfig-DSS（自 RHEL 9 开始）
- Camellia（自 RHEL 9 开始）
- ARIA
- IKEv1（自 RHEL 8 开始）

加密策略中启用的算法
每个加密策略都启用特定的密码套件和协议：

	LEGACY	DEFAULT	FIPS	FUTURE
IKEv1	否	否	否	否
3DES	否	否	否	否
RC4	否	否	否	否
DH	最少 2048 位	最少 2048 位	最少 2048 位	最少 3072 位
RSA	最少 2048 位	最少 2048 位	最少 2048 位	最少 3072 位
DSA	否	否	否	否
TLS v1.1 和更早版本	否	否	否	否
TLS v1.2 及更新版本	是	是	是	是
数字签名和证书中的 SHA-1	是	否	否	否
CBC 模式密码	是	否[a]	否[b]	否[c]
密钥小于 256 位的对称密码	是	是	是	否

[a] SSH 禁用 CBC 密码

[b] 除了 Kerberos 外，所有协议都禁用了 CBC 密码

[c] 除了 Kerberos 外，所有协议都禁用了 CBC 密码

其他资源

- **update-crypto-policies(8)** 手册页

3.2. 将系统范围的加密策略切换到与早期版本兼容的模式

Red Hat Enterprise Linux 9 中的默认系统范围的加密策略不允许使用旧的不安全协议进行通讯。对于需要与 Red Hat Enterprise Linux 6 兼容，以及需要与更早的版本兼容的情况，可以使用不太安全的 **LEGACY** 策略级别。

警告

切换到 **LEGACY** 策略级别会导致系统和应用程序的安全性较低。

步骤

1. 要将系统范围的加密策略切换到 **LEGACY** 级别，请以 **root** 用户身份输入以下命令：

```
# update-crypto-policies --set LEGACY
Setting system policy to LEGACY
```

其他资源

- 有关可用的加密策略级别列表，请参阅 **update-crypto-policies(8)** 手册页。
- 有关定义自定义加密策略的信息，请参阅 **update-crypto-policies(8)** 手册页中的 **自定义策略** 部分，以及 **crypto-policies(7)** 手册页中的 **加密策略定义格式** 部分。

3.3. 在 WEB 控制台中设置系统范围的加密策略

您可以在 RHEL web 控制台界面中直接设置系统范围的加密策略和子策略。除了四个预定义的系统范围的加密策略外，您现在还可以通过图形界面应用以下策略和子策略的组合：

DEFAULT:SHA1

启用了 **SHA-1** 算法的 **DEFAULT** 策略。

LEGACY:AD-SUPPORT

具有不太安全设置的 **LEGACY** 策略提高了活动目录服务的互操作性。

FIPS:OSPP

信息技术安全评估标准的通用标准需要具有进一步限制的 **FIPS** 策略。

警告

因为 **FIPS:OSPP** 系统范围的子策略包含对通用标准(CC)认证所需的加密算法的进一步限制，在设置它后，系统的互操作性降低。例如，您无法使用比 3072 位少的 RSA 和 DH 密钥、其它的 SSH 算法和几个 TLS 组。设置 **FIPS:OSPP** 也会阻止连接到 Red Hat Content Delivery Network (CDN)结构。另外，您无法将活动目录(AD)集成到使用 **FIPS:OSPP**的 IdM 部署中，使用 **FIPS:OSPP** 的 RHEL 主机和 AD 域之间的通信可能无法工作，或者某些 AD 帐户可能无法进行身份验证。

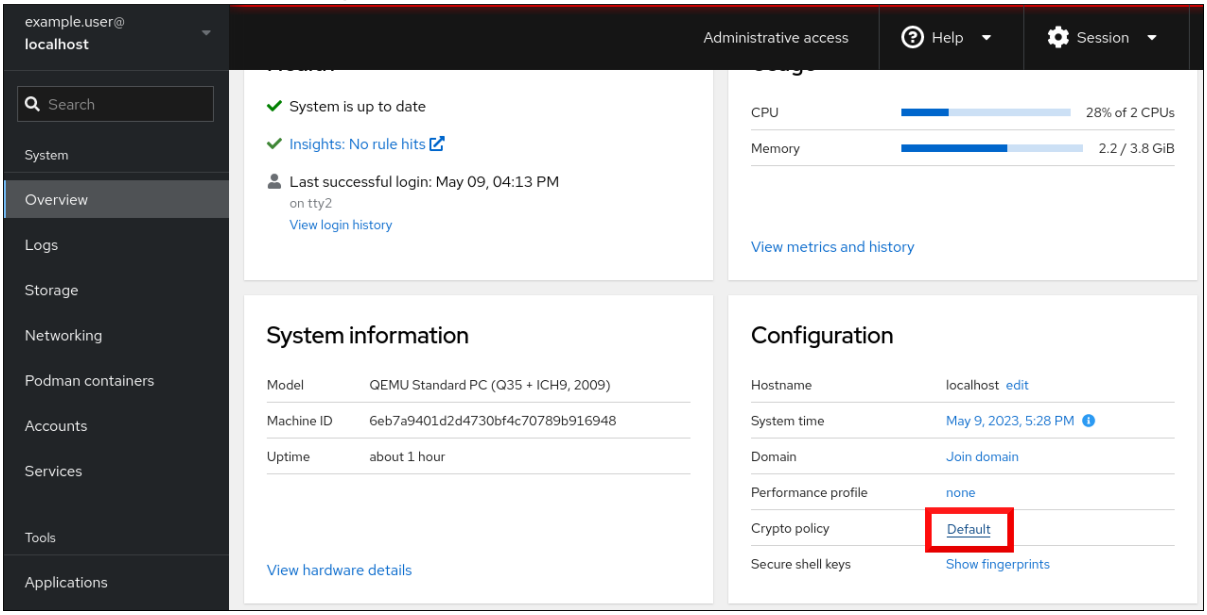
请注意，在设置了 **FIPS:OSPP** 加密子策略后，您的 **系统不符合 CC**。使 RHEL 系统符合 CC 标准的唯一正确方法是安装 **cc-config** 软件包。有关已认证的 RHEL 版本的列表、验证报告以及 CC 指南的链接，请参阅 国家信息保障合作伙伴(NIAP) 网站上托管的合规活动和政府标准知识库文章的 通用标准 部分。

先决条件

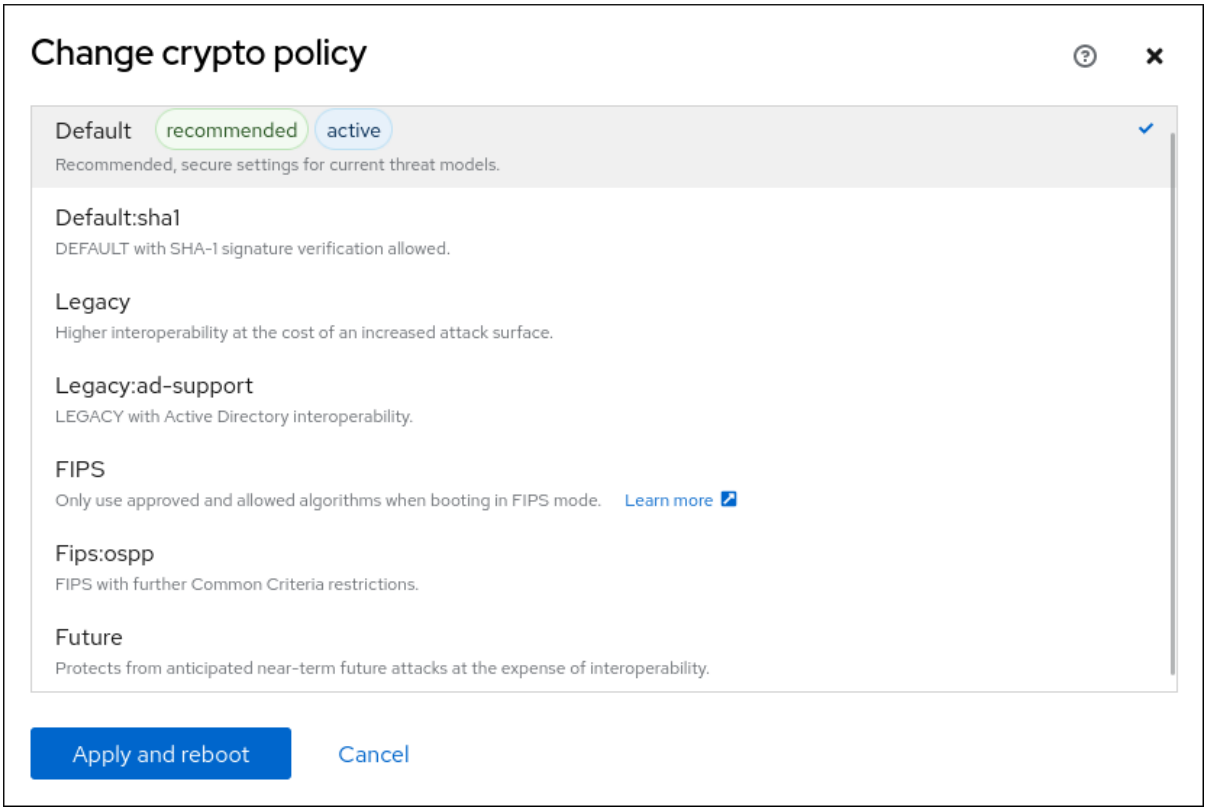
- 已安装 RHEL 9 web 控制台。详情请参阅安装和启用 Web 控制台。
- 您有 **root** 特权或权限来使用 **sudo** 输入管理命令的命令。

流程

1. 登录到 web 控制台。如需更多信息，请参阅 Web 控制台的日志记录。
2. 在 **Overview** 页面的 **Configuration** 卡中，点 **Crypto 策略**旁的当前策略值。



3. 在 **Change crypto policy** 对话框窗口中，点您要在系统中开始使用的策略。



4. 点应**用并重新引**导按钮。

验证

- 重启后，重新登录到 web 控制台，并检查 **Crypto policy** 值是否与您选择的值相符。或者，您可以输入 **update-crypto-policies --show** 命令来在终端中显示当前系统范围的加密策略。

3.4. 将系统切换到 FIPS 模式

系统范围的加密策略包含一个策略级别，它根据联邦信息处理标准(FIPS)出版物 140 的要求启用加密算法。在内部启用或禁用 FIPS 模式的 **fips-mode-setup** 工具使用 **FIPS** 系统范围的加密策略。

使用 FIPS 系统范围的加密策略将系统切换到 **FIPS** 模式不能保证符合 FIPS 140 标准。将系统设置为 FIPS 模式后，可能无法重新生成所有加密密钥。例如，在具有用户加密密钥的现有 IdM 域中，您无法重新生成所有密钥。

fips-mode-setup 工具在内部使用 **FIPS** 策略。但是，在使用 **--set FIPS** 选项的 **update-crypto-policies** 命令之上，**fips-mode-setup** 使用 **fips-finish-install** 工具确保 FIPS dracut 模块的安装，它还在内核命令行中添加 **fips=1** 引导选项，并重新生成初始 RAM 磁盘。

重要

仅 在 RHEL 安装过程中启用 FIPS 模式 可确保系统使用 FIPS 批准的算法生成所有密钥，并持续监控测试。

警告

完成 FIPS 模式的设置后，您无法在不将系统置于不一致状态的情况下关闭 FIPS 模式。如果您的场景需要这个更改，则唯一的正确方法是完全重新安装系统。

注意

RHEL 9 的加密模块尚未获得 FIPS 140-3 要求的认证。

步骤

1. 将系统切换到 FIPS 模式：

```
# fips-mode-setup --enable
Kernel initramdisks are being regenerated. This might take some time.
Setting system policy to FIPS
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
FIPS mode will be enabled.
Please reboot the system for the setting to take effect.
```

2. 重启您的系统以允许内核切换到 FIPS 模式：

```
# reboot
```

验证

- 重启后，您可以检查 FIPS 模式的当前状态：

```
# fips-mode-setup --check
FIPS mode is enabled.
```

其他资源

- **fips-mode-setup(8)** 手册页
- 在 FIPS 模式中安装系统
- 美国国家标准与技术研究院(NIST)网站上的 对加密模块的安全要求。

3.5. 在容器中启用 FIPS 模式

要启用联邦信息处理标准发布 140-2 (FIPS 模式)所要求的完整的加密模块自我检查，主机系统内核必须运行在 FIPS 模式下。**podman** 工具会在支持的容器上自动启用 FIPS 模式。

注意

fips-mode-setup 命令无法在容器中正常工作，在这种情况下无法用来启用或检查 FIPS 模式。

注意

RHEL 9 的加密模块尚未获得 FIPS 140-3 要求的认证。

先决条件

- 主机系统必须处于 FIPS 模式。

步骤

- 在启用了 FIPS 模式的系统上，**podman** 工具会在支持的容器上自动启用 FIPS 模式。

其他资源

- 将系统切换到 FIPS 模式。
- 在 FIPS 模式中安装系统

3.6. 使用与 FIPS 140-3 不兼容的加密的 RHEL 应用程序列表

要传递所有相关加密认证，如 FIPS 140-3，请使用核心加密组件集中的库。除 **libgcrypt** 外，这些库也会遵循 RHEL 系统范围的加密策略。

如需了解核心加密组件的概述、有关如何选择它们的信息、它们如何集成到操作系统中、它们如何支持硬件安全模块和智能卡，以及如何对它们应用加密认证，请参阅 RHEL 核心加密组件 。

使用与 FIPS 140-3 不兼容的加密的 RHEL 9 应用程序列表

Bacula

实施 CRAM-MD5 身份验证协议。

Cyrus SASL

使用 SCRAM-SHA-1 身份验证方法。

Dovecot

使用 SCRAM-SHA-1。

Emacs

使用 SCRAM-SHA-1。

FreeRADIUS

使用 MD5 和 SHA-1 进行身份验证协议。

Ghostscript

自定义加密实现（MD5、RC4、SHA-2、AES）来加密和解密文档

GRUB2

支持需要 SHA-1 的传统固件协议，并包含 **libgcrypt** 库。

iPXE

实施 TLS 堆栈。

Kerberos

保留对 SHA-1 的支持（与 Windows 互操作性）。

Lasso

lasso_wsse_username_token_derive_key() KDF（key derivation function）使用 SHA-1。

MariaDB、MariaDB Connector

mysql_native_password 身份验证插件使用 SHA-1。

MySQL

mysql_native_password 使用 SHA-1。

OpenIPMI

RAKP-HMAC-MD5 验证方法没有被 FIPS 批准使用，且不适用于 FIPS 模式。

Ovmf（UEFI 固件）、Edk2、shim

全加密堆栈(OpenSSL 库的嵌入式副本)。

Perl

使用 HMAC, HMAC-SHA1, HMAC-MD5, SHA-1, SHA-224,...

Pidgin

实现 DES 和 RC4 密码。

PKCS #12 文件处理 (OpenSSL、GnuTLS、NSS、Firefox、Java)

PKCS #12 的所有用法都不兼容 FIPS，因为用于计算整个文件 HMAC 的 Key Derivation Function (KDF) 都不是 FIPS。因此，PKP #12 文件被视为纯文本，用于 FIPS 合规性。对于 key-transport 的目的，使用 FIPS 批准的加密方案嵌套 PKCS #12 (.p12) 文件。

Poppler

如果原始 PDF 中存在这些算法（例如 MD5、RC4 和 SHA-1），则可使用签名、密码和加密保存 PDF。

PostgreSQL

实现 Blowfish、DES 和 MD5。KDF 使用 SHA-1。

QAT Engine

加密原语的混合硬件和软件实现(RSA、EC、DH、AES、...)

Ruby

提供不安全的 MD5 和 SHA-1 库函数。

Samba

保留对 RC4 和 DES 的支持（与 Windows 互操作性）。

Syslinux

BIOS 密码使用 SHA-1。

Unbound

DNS 规范要求 DNSSEC 解析器使用 DNSKEY 记录中的 SHA-1-based 算法进行验证。

Valgrind

AES、SHA 哈希。[2]

zip

自定义加密实现（不安全 PKWARE 加密算法）以使用密码加密和解密存档。

其他资源

- 合规活动和政府标准 知识库文章中的FIPS 140-2 和 FIPS 140-3 部分

- RHEL 核心加密组件 知识库文章

3.7. 将应用程序从下列系统范围的加密策略中排除

您可以通过在应用程序中直接配置受支持的密码套件和协议来自定义应用程序所使用的加密设置。

您还可以从 **/etc/crypto-policies/back-ends** 目录中删除与应用程序相关的符号链接，并使用您自定义的加密设置来替换它。此配置可防止对使用排除后端的应用程序使用系统范围的加密策略。此外，红帽不支持此修改。

3.7.1. 选择不使用系统范围的加密策略的示例

wget

要自定义 **wget** 网络下载器所使用的加密设置，请使用 **--secure-protocol** 和 **--ciphers** 选项。例如：

```
$ wget --secure-protocol=TLSv1_1 --ciphers="SECURE128" https://example.com
```

如需更多信息，请参阅 **wget(1)** 手册页中的 HTTPS(SSL/TLS)选项部分。

curl

要指定 **curl** 工具使用的密码，请使用 **--ciphers** 选项，并提供以冒号分隔的密码列表作为值。例如：

```
$ curl https://example.com --ciphers '@SECLEVEL=0:DES-CBC3-SHA:RSA-DES-CBC3-SHA'
```

如需更多信息，请参阅 **curl(1)** 手册页。

Firefox

尽管您无法在 **Firefox** Web 浏览器中选择不使用系统范围的加密策略，但您可以在 Firefox 的配置编辑器中进一步限制受支持的密码和 TLS 版本。在地址栏中输入 **about:config** ，并根据需要修改 **security.tls.version.min** 选项的值。将 **security.tls.version.min** 设置为 **1**，允许将 TLS 1.0 作为最低要求，**security.tls.version.min 2** 启用 TLS 1.1，如此等等。

OpenSSH

要为您的 OpenSSH 服务器选择不使用系统范围的加密策略，请在位于 **/etc/ssh/sshd_config.d/** 目录中的置入配置文件中指定加密策略，具有小于 50 的 2 位数前缀，以便按字典顺序排列在 **50-redhat.conf** 文件之前，并带有 **.conf** 后缀，例如 **49-crypto-policy-override.conf**。

详情请查看 **sshd_config(5)** 手册页。

要为您的 OpenSSH 客户端选择不使用系统范围的加密策略，请执行以下任务之一：

- 对于给定的用户，使用 **~/.ssh/config** 文件中特定于用户的配置覆盖全局 **ssh_config**。
- 对于整个系统，在 **/etc/ssh/ssh_config.d/** 目录中的置入配置文件中指定加密策略，具有小于 50 的两位数前缀，以便按字典顺序排列在 **50-redhat.conf** 文件之前，并具有 **.conf** 后缀，例如 **49-crypto-policy-override.conf**。

详情请查看 **ssh_config(5)** 手册页。

Libreswan

有关详细信息，请参阅 安全网络 文档中的 配置不使用系统范围加密策略的 IPsec 连接 。

其他资源

- **update-crypto-policies(8)** 手册页

3.8. 使用子策略自定义系统范围的加密策略

使用这个流程来调整启用的加密算法或协议集。

您可以在现有系统范围的加密策略之上应用自定义子策略，或者从头开始定义此类策略。

范围策略的概念允许为不同的后端启用不同的算法集合。您可以将每个配置指令限制到特定的协议、库或服务。

另外，指令也可以使用星号来指定使用通配符的多个值。

/etc/crypto-policies/state/CURRENT.pol 文件列出了通配符扩展后当前应用了系统范围加密策略中的所有设置。要使您的加密策略更严格，请考虑使用 **/usr/share/crypto-policies/policies/FUTURE.pol** 文件中列出的值。

您可以在 **/usr/share/crypto-policies/policies/modules/** 目录中找到示例子策略。这个目录中的子策略文件还包含注释掉的行中的描述。

流程

1. 签出到 **/etc/crypto-policies/policies/modules/** 目录：

```
# cd /etc/crypto-policies/policies/modules/
```

2. 为您的调整创建子策略，例如：

```
# touch MYCRYPTO-1.pmod
# touch SCOPES-AND-WILDCARDS.pmod
```

重要

在策略模块的文件名中使用大写字母。

3. 在您选择的文本编辑器中打开策略模块并插入修改系统范围加密策略的选项，例如：

```
# vi MYCRYPTO-1.pmod
```

```
min_rsa_size = 3072
hash = SHA2-384 SHA2-512 SHA3-384 SHA3-512
```

```
# vi SCOPES-AND-WILDCARDS.pmod
```

```
# Disable the AES-128 cipher, all modes
cipher = -AES-128-*

# Disable CHACHA20-POLY1305 for the TLS protocol (OpenSSL, GnuTLS, NSS, and
OpenJDK)
cipher@TLS = -CHACHA20-POLY1305
```

```
# Allow using the FFDHE-1024 group with the SSH protocol (libssh and OpenSSH)
group@SSH = FFDHE-1024+

# Disable all CBC mode ciphers for the SSH protocol (libssh and OpenSSH)
cipher@SSH = -*-CBC

# Allow the AES-256-CBC cipher in applications using libssh
cipher@libssh = AES-256-CBC+
```

4. 将更改保存到模块文件中。
5. 将您的策略调整应用到 **DEFAULT** 系统范围加密策略级别：

```
# update-crypto-policies --set DEFAULT:MYCRYPTO-1:SCOPES-AND-WILDCARDS
```

6. 要使您的加密设置对已经运行的服务和应用程序有效，请重启系统：

```
# reboot
```

验证

- 检查 **/etc/crypto-policies/state/CURRENT.pol** 文件是否包含您的更改，例如：

```
$ cat /etc/crypto-policies/state/CURRENT.pol | grep rsa_size
min_rsa_size = 3072
```

其他资源

- **update-crypto-policies(8)** 手册页中的 **自定义策略** 部分
- **crypto-policies(7)** 手册页中的 **加密策略定义格式** 部分
- 红帽博客文章 在 RHEL 8.2 中如何自定义加密策略

3.9. 重新启用 SHA-1

使用 SHA-1 算法创建和验证签名在 **DEFAULT** 加密策略中受到限制。如果您的场景需要使用 SHA-1 来验证现有或第三方加密签名，您可以通过应用 **SHA1** 子策略来启用该签名，RHEL 9 默认提供它。请注意，它较弱了系统的安全性。

先决条件

- 系统使用 **DEFAULT** 系统范围的加密策略。

步骤

1. 将 **SHA1** 子策略应用到 **DEFAULT** 加密策略：

```
# update-crypto-policies --set DEFAULT:SHA1
Setting system policy to DEFAULT:SHA1
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
```

2. 重启系统：

```
# reboot
```

验证

- 显示当前加密策略：

```
# update-crypto-policies --show
DEFAULT:SHA1
```

重要

使用 **update-crypto-policies --set LEGACY** 切换到 **LEGACY** 加密策略命令也会启用 SHA-1 进行签名。但是，**LEGACY** 加密策略还启用了其他弱加密算法，使您的系统变得更加容易受到攻击。这个临时解决方案只适用于需要启用其他传统加密算法比 SHA-1 签名的情况。

其他资源

- 从 RHEL 9 到 RHEL 6 系统的 SSH 无法正常工作 KCS 文章
- 使用 SHA-1 签名的软件包无法安装或升级 KCS 文章

3.10. 创建并设置自定义系统范围的加密策略

以下步骤演示了通过完整的策略文件来自定义系统范围的加密策略。

步骤

1. 为自定义创建一个策略文件：

```
# cd /etc/crypto-policies/policies/
# touch MYPOLICY.pol
```

或者，从复制四个预定义策略级别中的一个开始：

```
# cp /usr/share/crypto-policies/policies/DEFAULT.pol /etc/crypto-
policies/policies/MYPOLICY.pol
```

2. 在您选择的文本编辑器中编辑带有自定义加密策略的文件以满足您的要求，例如：

```
# vi /etc/crypto-policies/policies/MYPOLICY.pol
```

3. 将系统范围的加密策略切换到自定义级别：

```
# update-crypto-policies --set MYPOLICY
```

4. 要使您的加密设置对已经运行的服务和应用程序有效，请重启系统：

```
# reboot
```

其他资源

- **update-crypto-policies(8)** 手册页中的 **自定义策略** 部分和 **crypto-policies(7)** 手册页中的 **加密策略定义格式** 部分
- 如何在 RHEL 中自定义加密策略 红帽博客

[2] 在软件 hardware-offload 操作中重新实施，如 AES-NI 或 SHA-1 和 SHA-2 on ARM。

第 4 章 使用 RHEL 系统角色设置自定义加密策略

作为管理员，您可以使用 **crypto_policies** RHEL 系统角色，使用 Ansible Core 软件包。在许多不同的系统中快速且一致地配置自定义加密策略。

4.1. 使用 CRYPTO_POLICIES RHEL 系统角色设置自定义加密策略

您可以使用 **crypto_policies** 系统角色，从一个控制节点配置大量的受管节点。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- name: Configure crypto policies
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure crypto policies
      ansible.builtin.include_role:
        name: rhel-system-roles.crypto_policies
      vars:
        - crypto_policies_policy: FUTURE
        - crypto_policies_reboot_ok: true
```

   您可以将 *FUTURE* 值替换为您首选的加密策略，例如：**DEFAULT**、**LEGACY** 和 **FIPS:OSPP**.

   crypto_policies_reboot_ok: true 设置导致系统在系统角色更改加密策略后重启。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

   请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

警告

因为 **FIPS:OSPP** 系统范围的子策略包含对通用标准(CC)认证所需的加密算法的进一步限制，在设置它后，系统的互操作性降低。例如，您无法使用比 3072 位少的 RSA 和 DH 密钥、其它的 SSH 算法和几个 TLS 组。设置 **FIPS:OSPP** 也会阻止连接到 Red Hat Content Delivery Network (CDN)结构。另外，您无法将活动目录(AD)集成到使用 **FIPS:OSPP**的 IdM 部署中，使用 **FIPS:OSPP** 的 RHEL 主机和 AD 域之间的通信可能无法工作，或者某些 AD 帐户可能无法进行身份验证。

请注意，在设置了 **FIPS:OSPP** 加密子策略后，您的 **系统不符合 CC**。使 RHEL 系统符合 CC 标准的唯一正确方法是安装 **cc-config** 软件包。有关已认证的 RHEL 版本的列表、验证报告以及 CC 指南的链接，请参阅 国家信息保障合作伙伴(NIAP) 网站上托管的合规活动和政府标准知识库文章的 通用标准 部分。

验证

1. 在控制节点上，创建另一个 playbook，例如名为 **verify_playbook.yml** ：

```
---
- name: Verification
  hosts: managed-node-01.example.com
  tasks:
    - name: Verify active crypto policy
      ansible.builtin.include_role:
        name: rhel-system-roles.crypto_policies
    - debug:
        var: crypto_policies_active
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/verify_playbook.yml
```

3. 运行 playbook：

```
$ ansible-playbook ~/verify_playbook.yml
TASK [debug] *************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

crypto_policies_active 变量显示受管节点上活跃的策略。

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** file
- **/usr/share/doc/rhel-system-roles/crypto_policies/** directory

第 5 章 通过 PKCS#11 将应用程序配置为使用加密硬件

分离有关专用加密设备的 secret 信息部分，如用于最终用户身份验证的智能卡和加密令牌，以及用于服务器应用程序的硬件安全模块(HSM)，提供额外的安全层。在 RHEL 中，通过 PKCS #11 API 对加密硬件的支持在不同的应用程序之间是一致的，并且加密硬件上的机密隔离不是一项复杂的任务。

5.1. 通过 PKCS #11 的加密硬件支持

公钥加密标准(PKCS)#11向保存加密信息并执行加密功能的加密设备定义了一个应用编程接口(API)。

PKCS #11 引入了 *加密令牌*，它是一个以统一方式向应用程序提供每个硬件或软件设备的对象。因此，应用程序会查看通常被人使用的智能卡，以及通常被计算机作为 PKCS #11 加密令牌使用的硬件安全模块等设备。

PKCS #11 令牌可以存储各种对象类型，包括证书、数据对象以及公有、私有或机密密钥。这些对象通过 PKCS #11 Uniform Resource Identifier (URI)模式来唯一标识。

PKCS #11 URI 是一种标准方法，其根据对象属性来识别 PKCS #11 模块中的特定对象。这可让您以 URI 格式，使用同样的配置字符串来配置所有的库和应用程序。

RHEL 默认为智能卡提供 OpenSC PKCS #11 驱动程序。但是，硬件令牌和 HSM 可以有自己的 PKCS #11 模块，这些模块在系统中没有对应项。您可以使用 **p11-kit** 工具注册这样的 PKCS #11 模块，它作为系统中注册的智能卡驱动程序的包装器。

要使您自己的 PKCS #11 模块在系统上正常工作，请在 **/etc/pkcs11/modules/** 目录中添加一个新的文本文件

您可以通过在 **/etc/pkcs11/modules/** 目录中创建一个新的文本文件，来将自己的 PKCS #11 模块添加到系统。例如，**p11-kit** 中的 OpenSC 配置文件如下所示：

```
$ cat /usr/share/p11-kit/modules/opensc.module
module: opensc-pkcs11.so
```

其他资源

- PKCS #11 URI 方案
- 控制对智能卡的访问

5.2. 使用保存在智能卡中的 SSH 密钥

Red Hat Enterprise Linux 可让您使用保存在 OpenSSH 客户端智能卡中的 RSA 和 ECDSA 密钥。使用这个步骤使用智能卡而不是使用密码启用验证。

前提条件

- 在客户端中安装了 **opensc** 软件包，**pcscd** 服务正在运行。

流程

1. 列出所有由 OpenSC PKCS #11 模块提供的密钥，包括其 PKCS #11 URIs，并将输出保存到 *key.pub* 文件：

```
$ ssh-keygen -D pkcs11: > keys.pub
```

```
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-
path=/usr/lib64/pkcs11/opensc-pkcs11.so
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. 要使用远程服务器上的智能卡（*example.com*）启用验证，将公钥传送到远程服务器。使用带有上一步中创建的 *key.pub* 的 **ssh-copy-id** 命令：

```
$ ssh-copy-id -f -i keys.pub username@example.com
```

3. 要使用在第 1 步的 **ssh-keygen -D** 命令输出中的 ECDSA 密钥连接到 *example.com*，您只能使用 URI 中的一个子集，它是您的密钥的唯一参考，例如：

```
$ ssh -i "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so" example.com
Enter PIN for 'SSH key':
[example.com] $
```

4. 您可以使用 **~/.ssh/config** 文件中的同一 URI 字符串使配置持久：

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so"
$ ssh example.com
Enter PIN for 'SSH key':
[example.com] $
```

因为 OpenSSH 使用 **p11-kit-proxy** 包装器，并且 OpenSC PKCS #11 模块是注册到 PKCS#11 Kit 的，所以您可以简化前面的命令：

```
$ ssh -i "pkcs11:id=%01" example.com
Enter PIN for 'SSH key':
[example.com] $
```

如果您跳过 PKCS #11 URI 的 **id=** 部分，则 OpenSSH 会加载代理模块中可用的所有密钥。这可减少输入所需的数量：

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
[example.com] $
```

其他资源

- Fedora 28：在 OpenSSH 中更好地支持智能卡
- **p11-kit(8)**, **opensc.conf(5)**, **pcscd(8)**, **ssh(1)**, 和 **ssh-keygen(1)** man pages

5.3. 配置应用程序以使用智能卡上的证书进行身份验证

通过在应用程序中使用智能卡进行身份验证可能会提高安全性并简化自动化。您可以使用以下方法将公钥加密标准(PKCS) #11 URI 集成到应用程序中：

- **Firefox** web 浏览器会自动加载 **p11-kit-proxy** PKCS #11 模块。这意味着系统中的每个支持的智能卡都会被自动检测到。对于使用 TLS 客户端身份验证，不需要额外的设置，当服务器请求它们时，会自动使用智能卡中的密钥和证书。

- 如果您的应用程序使用 **GnuTLS** 或 **NSS** 库，则它已支持 PKCS #11 URI。另外，依赖 **OpenSSL** 库的应用程序可以通过 **openssl-pkcs11** 软件包提供的 **pkcs11** 引擎访问加密硬件模块，包括智能卡。

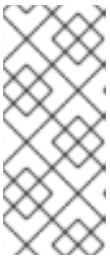
- 需要使用智能卡上的私钥，且不使用 **NSS**、**GnuTLS** 或 **OpenSSL** 的应用程序可以直接使用 **p11-kit** API 来使用加密硬件模块，包括智能卡，而不是使用特定 PKCS #11 模块的 PKCS #11 API。

- 使用 **wget** 网络下载工具，您可以指定 PKCS #11 URI，而不是本地存储的私钥和证书的路径。这可能会简化为需要安全存储私钥和证书的任务创建脚本。例如：

```
$ wget --private-key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --
certificate 'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
```

- 在使用 **curl** 工具时，您也可以指定 PKCS #11 URI：

```
$ curl --key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --cert
'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
```



注意

由于 PIN 是一种安全措施，它控制对保存在智能卡中的密钥的访问，而配置文件中包含纯文本形式的 PIN，因此请考虑采取额外的保护来防止攻击者读取 PIN。例如，您可以使用 **pin-source** 属性并提供 **file:**从文件中读取 PIN 的 URI。请参阅 RFC 7512:PKCS #11 URI Scheme Query Attribute Semantics 了解更多信息。请注意，不支持将命令路径用作 **pin-source** 属性的值。

其他资源

- **curl (1)**, **wget (1)**, 和 **p11-kit (8)** 手册页

5.4. 在 APACHE 中使用 HSM 保护私钥

Apache HTTP 服务器可以使用存储在硬件安全模块(HSM)上的私钥，这有助于防止密钥泄漏和中间人攻击。请注意，对于繁忙的服务器，这通常需要高性能的 HSM 。

对于 HTTPS 协议形式的安全通信，**Apache** HTTP 服务器(**httpd**)使用 OpenSSL 库。OpenSSL 本身不支持 PKCS #11 。要使用 HSMs，您必须安装 **openssl-pkcs11** 软件包，该软件包通过引擎界面提供对 PKCS #11 模块的访问。您可以使用 PKCS #11 URI 而不是常规文件名在 **/etc/httpd/conf.d/ssl.conf** 配置文件中指定服务器密钥和证书，例如：

```
SSLCertificateFile    "pkcs11:id=%01;token=softhsm;type=cert"
SSLCertificateKeyFile "pkcs11:id=%01;token=softhsm;type=private?pin-value=111111"
```

安装 **httpd-manual** 软件包以获取 **Apache** HTTP 服务器的完整文档，包括 TLS 配置。**/etc/httpd/conf.d/ssl.conf** 配置文件中的指令在 **/usr/share/httpd/manual/mod_ssl.html** 文件中进行了详细的描述。

5.5. 使用 HSM 保护 NGINX 中的私钥

Nginx HTTP 服务器可以使用存储在硬件安全模块(HSM)上的私钥，这有助于防止密钥泄漏和中间人攻击。请注意，对于繁忙的服务器，这通常需要高性能的 HSM 。

因为 **Nginx** 也使用 OpenSSL 进行加密操作，所以对 PKCS #11 的支持必须通过 **openssl-pkcs11** 引擎。**nginx** 目前只支持从 HSM 加载私钥，证书必须作为常规文件单独提供。修改 **/etc/nginx/nginx.conf** 配置文件中 **server** 部分的 **ssl_certificate** 和 **ssl_certificate_key** 选项：

```
ssl_certificate     /path/to/cert.pem
ssl_certificate_key "engine:pkcs11:pkcs11:token=softhsm;id=%01;type=private?pin-value=111111";
```

请注意，在 **Nginx** 配置文件中，PKCS #11 URI 需要 **engine:pkcs 11:** 前缀。这是因为其它 **pkcs11** 前缀引用引擎名称。

5.6. 其他资源

- **pkcs11.conf(5)** 手册页。

第 6 章 使用 POLKIT 控制对智能卡的访问

要涵盖智能卡中内置的机制（如 PIN 、PIN 平板和生物识别技术）无法防止的可能的威胁，以及更精细的控制，RHEL 使用 **polkit** 框架来控制对智能卡的访问控制。

系统管理员配置 **polkit** 以适合特定的场景，如非特权或非本地用户或服务的智能卡访问。

6.1. 通过 POLKIT 的智能卡访问控制

个人计算机/智能卡(PC/SC)协议指定将智能卡及其读卡器整合成计算系统的标准。在 RHEL 中，**pcc-lite** 软件包提供了中间件来访问使用 PC/SC API 的智能卡。此软件包的一部分 **pcscd** (PC/SC 智能卡)守护进程，确保系统可以使用 PC/SC 协议访问智能卡。

因为在智能卡内置的访问控制机制（如 PIN、PIN 平板 和生物识别技术）没有涵盖所有可能的威胁，所以 RHEL 使用 **polkit** 框架进行更强大的访问控制。**polkit** 授权管理器可以对特权操作授予访问权限。除了授予对磁盘的访问权限外，您还可以使用 **polkit** 来指定保护智能卡的策略。例如，您可以定义哪些用户可以使用智能卡执行哪些操作。

安装 **pcsc-lite** 软件包并启动 **pcscd** 守护进程后，系统会强制使用 **/usr/share/polkit-1/actions/** 目录中定义的策略。默认的系统范围的策略位于 **/usr/share/polkit-1/actions/org.debian.pcsc-lite.policy** 文件中。polkit 策略文件使用 XML 格式，其语法在 **polkit(8)** 手册页中进行了描述。

polkitd 服务监控 **/etc/polkit-1/rules.d/** 和 **/usr/share/polkit-1/rules.d/** 这些目录中存储的规则文件的任何更改。该文件包含 JavaScript 格式的授权规则。系统管理员可以在这两个目录中添加自定义规则文件，并且 **polkitd** 根据其文件名按照字母顺序读取它们。如果两个文件具有相同的名称，则首先读取 **/etc/polkit-1/rules.d/** 中的文件。

其他资源

- **polkit(8)** 、**polkitd(8)** 和 **pcscd(8)** 手册页。

6.2. 排除与 PC/SC 和 POLKIT 相关的问题

安装 **pcsc-lite** 软件包并启动 **pcscd** 守护进程后自动强制的 polkit 策略，即使用户没有与智能卡直接交互也会在用户会话中要求身份验证。在 GNOME 中，您可以看到以下错误信息：

```
Authentication is required to access the PC/SC daemon
```

请注意，当安装与智能卡相关的其他软件包（如 **opensc** ）时，系统会将 **pcsc-lite** 软件包作为依赖项安装。

如果您的场景不需要与智能卡进行任何交互，并且您希望阻止 PC/SC 守护进程的授权请求，您可以删除 **pcsc-lite** 软件包。尽可能保持所需软件包的最小安装是良好的安全实践。

如果您使用智能卡，请通过检查 **/usr/share/polkit-1/actions/org.debian.pcsc-lite.policy** 中系统提供的策略中的规则来进行故障排除。您可以将自定义规则文件添加到 **/etc/polkit-1/rules.d/** 目录中的策略中，例如 **03-allow-pcscd.rules**。请注意，规则文件使用 JavaScript 语法，策略文件采用 XML 格式。

要了解系统显示的授权请求，请检查 Journal 日志，例如：

```
$ journalctl -b | grep pcsc
...
Process 3087 (user: 1001) is NOT authorized for action: access_pcsc
...
```

前面的日志条目表示用户没有被授权根据策略执行操作。您可以通过在 **/etc/polkit-1/rules.d/** 中添加相应的规则来解决此拒绝。

您还可以搜索与 **polkitd** 单元相关的日志条目，例如：

```
$ journalctl -u polkit
...
polkitd[NNN]: Error compiling script /etc/polkit-1/rules.d/00-debug-pcscd.rules
...
polkitd[NNN]: Operator of unix-session:c2 FAILED to authenticate to gain authorization for action
org.debian.pcsc-lite.access_pcsc for unix-process:4800:14441 [/usr/libexec/gsd-smartcard] (owned
by unix-user:group)
...
```

在前面的输出中，第一个条目表示规则文件中包含一些语法错误。第二个条目表示用户未能获得对 **pcscd** 的访问权限。

您还可以通过一个简短的脚本列出使用 PC/SC 协议的所有应用程序。创建一个可执行文件，如 **pcsc-apps.sh**，然后插入以下代码：

```
#!/bin/bash

cd /proc

for p in [0-9]*
do
 if grep libpcsclite.so.1.0.0 $p/maps &> /dev/null
 then
  echo -n "process: "
  cat $p/cmdline
  echo " ($p)"
 fi
done
```

以 **root** 身份运行脚本：

```
# ./pcsc-apps.sh
process: /usr/libexec/gsd-smartcard (3048)
enable-sync --auto-ssl-client-auth --enable-crashpad (4828)
...
```

其他资源

- **journalctl**,**polkit(8)**, **polkitd(8)** 和 **pcscd(8)** 手册页。

6.3. 向 PC/SC 显示关于 POLKIT 授权的更多详细信息

在默认配置中，**polkit** 授权框架仅将有限的信息发送到 Journal 日志。您可以通过添加新的规则来扩展与 PC/SC 协议相关的 **polkit** 日志条目。

先决条件

- 您已在系统上安装了 **pcsc-lite** 软件包。

- **pcscd** 守护进程正在运行。

步骤

1. 在 **/etc/polkit-1/rules.d/** 目录中创建一个新文件：

```
# touch /etc/polkit-1/rules.d/00-test.rules
```

2. 在您选择的编辑器中编辑该文件，例如：

```
# vi /etc/polkit-1/rules.d/00-test.rules
```

3. 插入以下行：

```
polkit.addRule(function(action, subject) {
  if (action.id == "org.debian.pcsc-lite.access_pcsc" ||
   action.id == "org.debian.pcsc-lite.access_card") {
 polkit.log("action=" + action);
 polkit.log("subject=" + subject);
  }
});
```

保存文件并退出编辑器。

4. 重启 **pcscd** 和 **polkit** 服务：

```
# systemctl restart pcscd.service pcscd.socket polkit.service
```

验证

1. 为 **pcscd** 发出一个授权请求。例如，打开 Firefox Web 浏览器，或者使用 **opensc** 软件包提供的 **pkcs11-tool -L** 命令。

2. 显示扩展的日志条目，例如：

```
# journalctl -u polkit --since "1 hour ago"
polkitd[1224]: <no filename>:4: action=[Action id='org.debian.pcsc-lite.access_pcsc']
polkitd[1224]: <no filename>:5: subject=[Subject pid=2020481 user=user'
groups=user,wheel,mock,wireshark seat=null session=null local=true active=true]
```

其他资源

- **polkit(8)** 和 **polkitd(8)** 手册页。

6.4. 其他资源

- 控制对智能卡访问 的红帽博客文章。

第 7 章 扫描系统以了解配置合规性和漏洞

合规审计是一个确定给定对象是否遵循合规策略中指定的所有规则的流程。合规策略由安全专业人员定义的，他们通常以检查清单的形式指定计算环境应使用的必要设置。

跨组织甚至同一组织内不同系统之间的合规政策可能有很大差异。这些政策之间的差异取决于每个系统的用途及其对组织的重要性。自定义软件设置和部署特征也需要自定义策略检查表。

7.1. RHEL 中的配置合规工具

您可以使用以下配置合规工具在 Red Hat Enterprise Linux 中执行一个全自动的合规审计。这些工具基于安全内容自动化协议(SCAP)标准，专为自动定制合规策略而设计。

SCAP 工作台

scap-workbench 图形工具旨在在单个本地或远程系统上执行配置和漏洞扫描。您还可以根据这些扫描和评估，使用它来生成安全报告。

OpenSCAP

OpenSCAP 库附带的 **oscap** 命令行工具旨在对本地系统上执行配置和漏洞扫描，以验证配置合规性内容，并根据这些扫描和评估生成报告和指南。

重要

在使用 **OpenSCAP** 时可能会遇到内存消耗问题，这可能会导致程序过早停止，并阻止生成任何结果文件。详情请查看 OpenSCAP 内存消耗问题 知识库文章。

SCAP 安全指南(SSG)

scap-security-guide 软件包为 Linux 系统提供安全策略的集合。该指南包括一个实用强化建议目录，在适用的情况下与政府的要求相关联。该项目弥补了一般性政策要求和具体实施指南间的差距。

脚本检查引擎(SCE)

通过 SCE （其是 SCAP 协议的扩展），管理员可以使用脚本语言（如 Bash、Python 和 Ruby）编写其安全内容。SCE 扩展在 **openscap-engine-sce** 软件包中提供。SCE 本身不是 SCAP 标准的一部分。

要在多个系统上远程执行自动合规审计，您可以使用 Red Hat Satellite 的 OpenSCAP 解决方案。

其他资源

- **oscap(8)**、**scap-workbench(8)** 和 **scap-security-guide(8)** 手册页
- 红帽安全演示：创建自定义安全策略内容到 Automate 安全合规性
- 红帽安全演示：使用 RHEL 安全技术保护您自己
- 管理 Red Hat Satellite Guide 指南中的安全合规性管理 .

7.2. 漏洞扫描

7.2.1. 红帽安全咨询 OVAL 源

Red Hat Enterprise Linux 安全审计功能是基于安全内容自动化协议(SCAP)标准的。SCAP 是一种多用途规格框架，支持自动化配置、漏洞和补丁检查、技术控制合规性活动和安全衡量。

SCAP 规范创建一个生态系统，其中安全内容的格式是众所周知的且标准化的，尽管扫描程序或策略编辑器的实现并不是强制性的。这使得组织能够一次性构建它们的安全策略（SCAP 内容），无论他们使用了多少家安全供应商。

开放式漏洞评估语言(OVAL)是 SCAP 最基本、最古老的组件。与其他工具和自定义脚本不同，OVAL 以声明式方法描述资源的必需状态。OVAL 代码从不直接执行，而是使用称为扫描器的 OVAL 解释器工具。OVAL 的声明性质可确保评估的系统状态不会被意外修改。

与所有其他 SCAP 组件一样，OVAL 也是基于 XML。SCAP 标准定义了多个文档格式。每一个都包括一种不同的信息，用于不同的目的。

红帽产品安全团队 通过跟踪和调查影响红帽客户的所有安全问题，来帮助客户评估和管理风险。它在红帽客户门户网站上提供及时、简洁的补丁和安全公告。红帽创建和支持 OVAL 补丁定义，提供机器可读的安全公告版本。

由于平台、版本及其他因素之间存在差异，红帽产品安全严重性等级评级无法直接与第三方提供的通用漏洞评分系统(CVSS)基准评级一致。因此，我们建议您使用 RHSA OVAL 定义，而不是第三方提供的定义。

RHSA OVAL 定义 可以单独提供，也可以作为一个完整的软件包提供，并在红帽客户门户网站上提供新安全公告的一小时内进行更新。

每个 OVAL 补丁定义将一对一地映射到红帽安全公告(RHSA)。由于 RHSA 可以包含对多个漏洞的修复，因此每个漏洞都通过其通用漏洞和风险(CVE)名称单独列出，并在我们的公共 bug 数据库中有一个指向其条目的链接。

RHSA OVAL 定义旨在检查系统上安装的 RPM 软件包是否存易受攻击的版本。可以扩展这些定义以包括进一步的检查，例如，查找软件包是否在易受攻击的配置中被使用。这些定义旨在涵盖红帽所提供的软件和更新。需要其他定义来检测第三方软件的补丁状态。

注意

Red Hat Enterprise Linux 合规服务的 Red Hat Insights 可帮助 IT 安全和合规性管理员评估、监控和报告 Red Hat Enterprise Linux 系统安全策略合规性。您还可以完全在合规服务 UI 中创建和管理 SCAP 安全策略。

其他资源

- 红帽和OVAL的兼容性
- 红帽和CVE的兼容性
- 产品安全概述中的通知和建议
- 安全数据指标

7.2.2. 扫描系统漏洞

oscap命令行实用程序使您能够扫描本地系统，验证配置合规性内容，并根据这些扫描和评估生成报告和指南。此工具充当 OpenSCAP 库的前端，并根据它所处理的 SCAP 内容类型将其功能分组到模块（子命令）。

前提条件

- **openscap-scanner** 和 **bzip2** 软件包已安装。

流程

1. 下载系统的最新 RHSA OVAL 定义：

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL9/rhel-9.oval.xml.bz2 | bzip2 --decompress > rhel-9.oval.xml
```

2. 扫描系统漏洞并将结果保存到 *vulnerability.html* 文件中：

```
# oscap oval eval --report vulnerability.html rhel-9.oval.xml
```

验证

- 在您选择的浏览器中检查结果，例如：

```
$ firefox vulnerability.html &
```

其他资源

- **oscap(8)** 手册页
- Red Hat OVAL 定义
- OpenSCAP 内存消耗问题

7.2.3. 扫描远程系统的漏洞

您还可以使用通过 SSH 协议的 **oscap-ssh** 工具，使用 OpenSCAP 扫描程序来检查远程系统的漏洞。

前提条件

- **openscap-utils** 和 **bzip2** 软件包已安装在您用于扫描的系统中。
- **openscap-scanner** 软件包已安装在远程系统上。
- SSH 服务器在远程系统上运行。

流程

1. 下载系统的最新 RHSA OVAL 定义：

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL9/rhel-9.oval.xml.bz2 | bzip2 --decompress > rhel-9.oval.xml
```

2. 扫描 SSH 在端口 22 上运行、用户名为 *joesec*、主机名为 *machine1* 的远程系统上的漏洞，并将结果保存到 *remote-vulnerability.html* 文件中：

```
# oscap-ssh joesec@machine1 22 oval eval --report remote-vulnerability.html rhel-9.oval.xml
```

其他资源

- **oscap-ssh(8)**

- Red Hat OVAL 定义
- OpenSCAP 内存消耗问题

7.3. 配置合规性扫描

7.3.1. RHEL 中的配置合规性

您可以使用配置合规性扫描来遵循特定组织定义的基准。例如，如果您与美国政府合作，您可能需要使您的系统与操作系统保护配置文件(OSPP)保持一致，如果您是一个支付处理器，您可能需要使您的系统与支付卡行业数据安全标准(PCI-DSS)保持一致。您还可以执行配置合规性扫描来强化您的系统安全。

红帽建议您遵循 SCAP 安全指南软件包中提供的安全内容自动化协议(SCAP)的内容，因为它符合红帽针对受影响组件的最佳实践。

SCAP 安全指南软件包提供了符合 SCAP 1.2 和 SCAP 1.3 标准的内容。**openscap 扫描器**实用程序与SCAP安全指南包中提供的SCAP 1.2和SCAP 1.3内容兼容。

重要

执行配置合规性扫描不能保证系统是合规的。

SCAP 安全指南套件以数据流文档的形式为多个平台提供配置文件。数据流是包含定义、基准、配置文件和单个规则的文件。每条规则都规定了合规的适用性和要求。RHEL 提供多个配置文件来遵守安全策略。除了行业标准之外，红帽数据流还包含用于修复失败规则的信息。

合规性扫描资源的结构

```
Data stream
  ├── xccdf
  |     ├── benchmark
  |           ├── profile
  |           |    ├──rule reference
  |           |    └──variable
  |           ├── rule
  |                ├── human readable data
  |                ├── oval reference
  ├── oval         ├── ocil reference
  ├── ocil         ├── cpe reference
  └── cpe          └── remediation
```

配置文件是基于安全策略的一组规则，如 OSPP、PCI-DSS 和健康保险可移植性和责任法案(HIPAA)。这可让您以自动化的方式审核系统，以符合安全标准。

您可以修改（定制）配置文件来自定义某些规则，例如密码长度。有关配置文件定制的更多信息，请参阅使用 SCAP Workbench 自定义安全配置文件。

7.3.2. OpenSCAP 扫描的可能结果

根据应用到 OpenSCAP 扫描的数据流和配置文件，以及系统的各种属性，每个规则可能会产生一个特定的结果。以下是可能的结果，以及其含义的简要解释：

Pass

扫描没有发现与此规则有任何冲突。

Fail

扫描发现与此规则有冲突。

Not checked

OpenSCAP 对此规则不执行自动评估。手动检查您的系统是否符合此规则。

Not applicable

此规则不适用于当前配置。

Not selected

此规则不是配置文件的一部分。OpenSCAP 不评估此规则，也不会在结果中显示这些规则。

Error

扫描遇到了错误。要获得更多信息，您可以输入带有 **--verbose DEVEL** 选项的 **oscap** 命令。考虑打开 bug 报告。

Unknown

扫描遇到了意外情况。要获得更多信息，您可以输入带有 **'--verbose DEVEL** 选项的 **oscap** 命令。考虑打开 bug 报告。

7.3.3. 查看配置文件是否符合配置合规

在决定使用配置文件进行扫描或补救前，您可以使用 **oscap info** 子命令列出它们并检查其详细描述。

前提条件

- **openscap-scanner** 和 **scap-security-guide** 软件包已安装。

流程

1. 列出 SCAP 安全指南项目所提供的带有安全合规配置文件的所有可用文件：

```
$ ls /usr/share/xml/scap/ssg/content/
ssg-rhel9-ds.xml
```

2. 使用 **oscap info** 子命令显示有关所选数据流的详细信息。包含数据流的 XML 文件由其名称中的 **-ds** 字符串表示。在 **Profiles** 部分，您可以找到可用的配置文件及其 ID 列表：

```
$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
Profiles:
...
  Title: Australian Cyber Security Centre (ACSC) Essential Eight
    Id: xccdf_org.ssgproject.content_profile_e8
  Title: Health Insurance Portability and Accountability Act (HIPAA)
    Id: xccdf_org.ssgproject.content_profile_hipaa
  Title: PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 9
    Id: xccdf_org.ssgproject.content_profile_pci-dss
...
```

3. 从数据流文件中选择一个配置文件，并显示所选配置文件的更多详情。为此，可使用带有 **--profile** 选项的 **oscap info** ，后跟上一命令输出中显示的 ID 的最后一部分。例如，HIPPA 配置文件的 ID 是：**xccdf_org.ssgproject.content_profile_hipaa**，**--profile** 选项的值为 **hipaa** ：

```
$ oscap info --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

```
...
Profile
 Title: [RHEL9 DRAFT] Health Insurance Portability and Accountability Act (HIPAA)
 Id: xccdf_org.ssgproject.content_profile_hipaa

 Description: The HIPAA Security Rule establishes U.S. national standards to protect
individuals' electronic personal health information that is created, received, used, or
maintained by a covered entity. The Security Rule requires appropriate administrative,
physical and technical safeguards to ensure the confidentiality, integrity, and security of
electronic protected health information.  This profile configures Red Hat Enterprise Linux 9 to
the HIPAA Security Rule identified for securing of electronic protected health information.
Use of this profile in no way guarantees or makes claims against legal compliance against
the HIPAA Security Rule(s).
```

其他资源

- **scap-security-guide(8)** man page
- OpenSCAP 内存消耗问题

7.3.4. 评估配置是否符合特定基准

要确定您的系统是否符合特定基准，请按照以下步骤操作：

前提条件

- **openscap-scanner** 和 **scap-security-guide** 软件包已安装
- 您知道系统应遵守的基准中的配置文件的 ID。要查找 ID，请参阅 查看配置合规性配置文件。

步骤

1. 使用所选配置文件评估系统的合规性，并将扫描结果保存在 *report.html* HTML 文件中，例如：

```
$ oscap xccdf eval --report report.html --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

2. 可选：使用 **machine1** 主机名、在端口 **22** 上运行的 SSH 扫描远程系统，以及 **joesec** 用户名合规性，并将结果保存到 **remote-report.html** 文件中：

```
$ oscap-ssh joesec@machine1 22 xccdf eval --report remote_report.html --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

其他资源

- **scap-security-guide(8)** man page
- **/usr/share/doc/scap-security-guide/** 目录中的 **SCAP 安全指南** 文档
- **/usr/share/doc/scap-security-guide/guides/ssg-rhel9-guide-index.html** - [Red Hat Enterprise Linux 9 的安全配置指南]与 **scap-security-guide-doc** 软件包一起安装
- OpenSCAP 内存消耗问题

7.4. 修复系统，使其与特定基准一致

您可以修复 RHEL 系统，以与特定基准保持一致。本例使用 Health Insurance Portability and Accountability Act (HIPAA)配置文件，但您可以进行修复，以与 SCAP 安全指南提供的任何其他配置集保持一致。有关列出可用配置文件的详情，请查看 查看配置合规的配置文件 部分。

警告

如果不小心使用，在启用了 **Remediate** 选项的情况下运行系统评估可能会导致系统无法正常工作。红帽不提供任何自动的方法来恢复由安全补救机制所做的更改。在 RHEL 系统上的默认配置中支持修复。如果在安装后更改了您的系统，运行补救可能无法使其与所需安全配置兼容。

先决条件

- **scap-security-guide** 软件包已安装在您的 RHEL 系统上。

步骤

1. 使用带有 **--remediate** 选项的 **oscap** 命令：

```
# oscap xccdf eval --profile hipaa --remediate /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

2. 重启您的系统。

验证

1. 使用 HIPAA 配置文件评估系统的合规性，并将扫描结果保存在 **hipaa_report.html** 文件中：

```
$ oscap xccdf eval --report hipaa_report.html --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

其他资源

- **scap-security-guide(8)** 和 **oscap(8)** 手册页

7.5. 使用 SSG ANSIBLE PLAYBOOK 修复系统以与特定基准保持一致

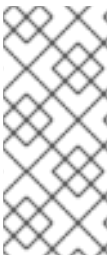
您可以使用 SCAP 安全指南项目中的 Ansible playbook 文件修复您的系统，使其与特定的基准保持一致。本例使用 Health Insurance Portability and Accountability Act (HIPAA)配置文件，但您可以进行修复，以与 SCAP 安全指南提供的任何其他配置集保持一致。有关列出可用配置文件的详情，请查看 查看配置合规的配置文件 部分。

警告

如果不小心使用，在启用了 **Remediate** 选项的情况下运行系统评估可能会导致系统无法正常工作。红帽不提供任何自动的方法来恢复由安全补救机制所做的更改。在 RHEL 系统上的默认配置中支持修复。如果在安装后更改了您的系统，运行补救可能无法使其与所需安全配置兼容。

先决条件

- **scap-security-guide** 软件包已安装。
- **ansible-core** 软件包已安装。如需更多信息，请参阅 Ansible 安装指南。



注意

在 RHEL 8.6 及更高版本中，Ansible Engine 被 **ansible-core** 软件包替代，该软件包仅包含内置模块。请注意，很多 Ansible 补救使用社区和可移植操作系统接口(POSIX)集合中的模块，它们没有包含在内置模块中。在这种情况下，您可以使用 Bash 补救来作为 Ansible 补救的替代。RHEL 9 中的 Red Hat Connector 包括必要的 Ansible 模块，以使修复 playbook 与 Ansible Core 一起工作。

流程

1. 使用 Ansible 修复您的系统，使其与 HIPAA 一致：

```
# ansible-playbook -i localhost, -c local /usr/share/scap-security-guide/ansible/rhel9-playbook-hipaa.yml
```

2. 重新启动系统。

验证

1. 使用 HIPAA 配置文件评估系统的合规性，并将扫描结果保存在 **hipaa_report.html** 文件中：

```
# oscap xccdf eval --profile hipaa --report hipaa_report.html /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

其他资源

- **scap-security-guide(8)** 和 **oscap(8)** 手册页
- Ansible 文档

7.6. 创建修复 ANSIBLE PLAYBOOK，使系统与特定的基准一致

您可以创建一个 Ansible playbook，其只包含使您的系统与特定基准保持一致所需的修正。这个示例使用了健康保险可移植性和责任法案(HIPAA)配置文件。通过这个过程，您可以创建一个较小的 playbook，其不包括已经满足的需求。按照以下步骤，您不需要以任何方式修改您的系统，您只需为后续应用程序准备一个文件。

注意

在 RHEL 9 中，Ansible Engine 被 **ansible-core** 软件包替代，该软件包只包含内置模块。请注意，很多 Ansible 补救使用社区和可端口操作系统接口(POSIX)集合中的模块，它们没有包含在内置模块中。在这种情况下，您可以使用 Bash 补救来作为 Ansible 补救的替代。RHEL 9.0 中的 Red Hat Connector 包括必要的 Ansible 模块，以使修复 playbook 与 Ansible Core 一起工作。

先决条件

- **scap-security-guide** 软件包已安装。

步骤

1. 扫描系统并保存结果：

```
# oscap xccdf eval --profile hipaa --results <hipaa-results.xml>
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

2. 使用结果在文件中找到结果 ID 的值：

```
# oscap info <hipaa-results.xml>
```

3. 根据在第 1 步中生成的文件生成 Ansible playbook：

```
# oscap xccdf generate fix --fix-type ansible --result-id <xccdf_org.open-
scap_testresult_xccdf_org.ssgproject.content_profile_hipaa> --output <hipaa-
remediations.yml> <hipaa-results.xml>
```

4. 查看生成的文件，其中包含在第 1 步中执行扫描过程中失败的规则的 Ansible 修复。查看生成的文件后，您可以使用 **ansible-playbook *<hipaa-remediations.yml>*** 命令应用它。

验证

- 在您选择的文本编辑器中，检查生成的 ***<hipaa-remediations.yml>*** 文件是否包含在第 1 步中执行的扫描中失败的规则。

其他资源

- **scap-security-guide(8)** 和 **oscap(8)** 手册页
- Ansible 文档

7.7. 为后续应用程序创建补救 BASH 脚本

使用此流程创建一个 Bash 脚本，其中包含使您的系统与 HIPAA 等安全配置文件一致的补救。通过以下步骤，您不需要对系统进行任何修改，您只需为后续应用准备一个文件。

先决条件

- **scap-security-guide** 软件包已安装在您的 RHEL 系统上。

流程

1. 使用 **oscap** 命令扫描系统，并将结果保存到 XML 文件中。在以下示例中，**oscap** 会根据 **hipaa** 配置文件评估系统：

```
# oscap xccdf eval --profile hipaa --results <hipaa-results.xml>
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

2. 使用结果在文件中找到结果 ID 的值：

```
# oscap info <hipaa-results.xml>
```

3. 根据在第 1 步中生成的结果文件生成 Bash 脚本：

```
# oscap xccdf generate fix --fix-type bash --result-id <xccdf_org.open-
scap_testresult_xccdf_org.ssgproject.content_profile_hipaa> --output <hipaa-
remediations.sh> <hipaa-results.xml>
```

4. ***<hipaa-remediations.sh>*** 文件包含在第 1 步中执行扫描过程中失败的规则的补救。查看生成的文件后，当您位于与此文件相同的目录中时，您可以使用 **./*<hipaa-remediations.sh>*** 命令应用它。

验证

- 在您选择的文本编辑器中，检查 ***<hipaa-remediations.sh>*** 文件是否包含在第 1 步中执行的扫描中失败的规则。

其他资源

- **scap-security-guide(8)**、**oscap(8)** 和 **bash(1)** 手册页

7.8. 使用 SCAP WORKBENCH 用自定义配置文件扫描系统

scap-workbench软件包中包含的**SCAP Workbench**是一个图形化的实用程序，用户可以在单个本地或远程系统上进行配置和漏洞扫描，对系统进行修复，并根据扫描评估结果生成报告。请注意，与 **oscap** 命令行工具相比，**SCAP Workbench** 的功能有限。**SCAP Workbench** 以数据流文件的形式处理安全内容。

7.8.1. 使用 SCAP Workbench 来扫描和修复系统

要根据所选的安全策略来评估您的系统，请使用以下流程。

先决条件

- **scap-workbench** 软件包已经安装在您的系统中。

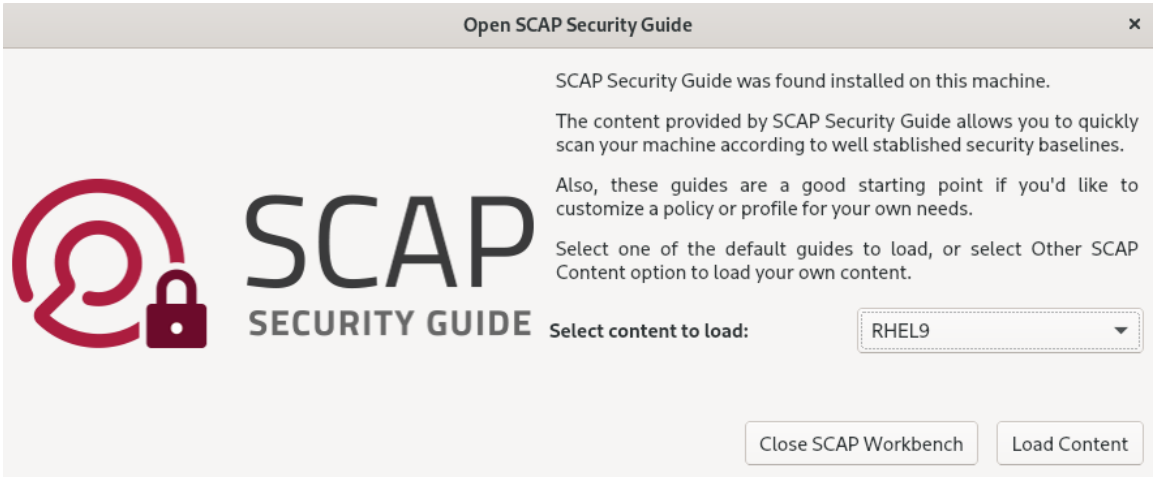
流程

1. 要从 **GNOME Classic** 桌面环境运行 **SCAP Workbench**，请按 **Super** 键进入 **Activities Overview**，输入 **scap-workbench**，然后按 **Enter**。或者，使用：

```
$ scap-workbench &
```

2. 使用以下其中一个选项来选择安全策略：

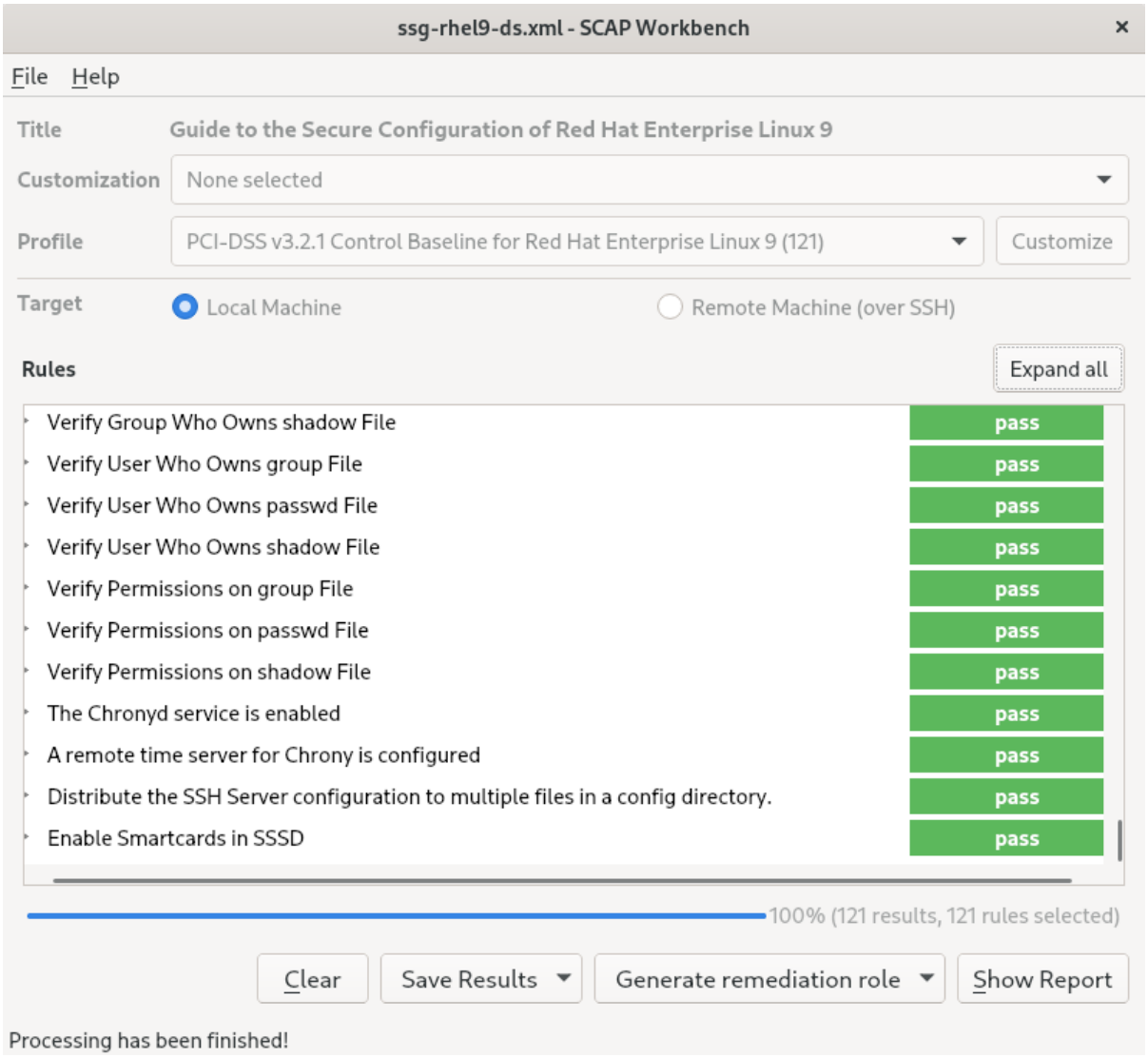
- 开始窗口中的 **Load Content** 按钮
- **打开 SCAP 安全指南中的内容**
- 在 **File** 中打开 **Other Content**，搜索相关的 XCCDF、SCAP RPM 或数据流文件。



3. 您可以选择 **Remediate** 复选框来允许自动修正系统配置。启用此选项后，**SCAP Workbench** 会尝试根据策略所应用的安全规则来修改系统配置。这个过程应该修复系统扫描过程中失败的相关检查。

> **警告**
>
> 如果不小心使用，在启用了 **Remediate** 选项的情况下运行系统评估可能会导致系统无法正常工作。红帽不提供任何自动的方法来恢复由安全补救机制所做的更改。在 RHEL 系统上的默认配置中支持修复。如果在安装后更改了您的系统，运行补救可能无法使其与所需安全配置兼容。

4. 单击**Scan**按钮，使用所选配置文件扫描您的系统。



5. 要以 XCCDF、ARF 或 HTML 文件的形式保存扫描结果，请点击 **Save Results** 组合框。选择 **HTML Report** 选项，以人类可读的格式生成扫描报告。XCCDF 和 ARF（数据流）格式适合进一步自动处理。您可以重复选择所有三个选项。

6. 要将基于结果的补救导出到文件，请使用 **Generate remediation role** 弹出菜单。

7.8.2. 使用 SCAP Workbench 自定义安全配置文件

您可以通过更改某些规则中的参数（如最小密码长度）、删除以不同方式涵盖的规则，并选择额外的规则来自定义安全配置文件，以实现内部策略。您不能通过自定义配置文件来定义新规则。

以下流程演示了如何使用 **SCAP Workbench** 来自定义（定制）配置文件。您还可以保存定制的配置文件，以便在 **oscap** 命令行工具中使用。。

先决条件

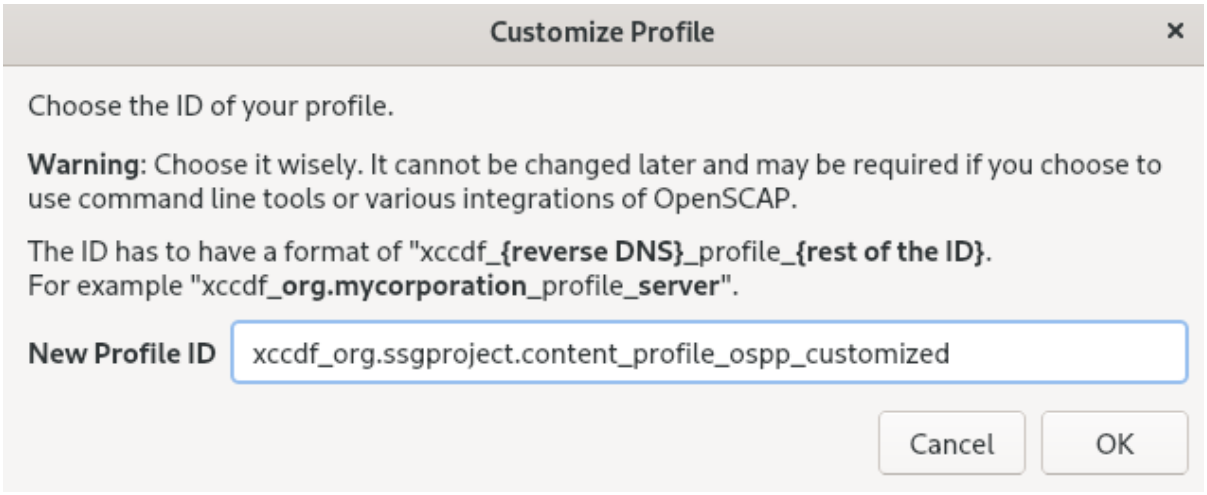
- **scap-workbench** 软件包已经安装在您的系统中。

流程

1. 运行 **SCAP Workbench**，选择要自定义的配置文件，方法是使用 **打开 SCAP 安全指南中的内容** 或者在 **File** 菜单中 **打开其他内容** 。

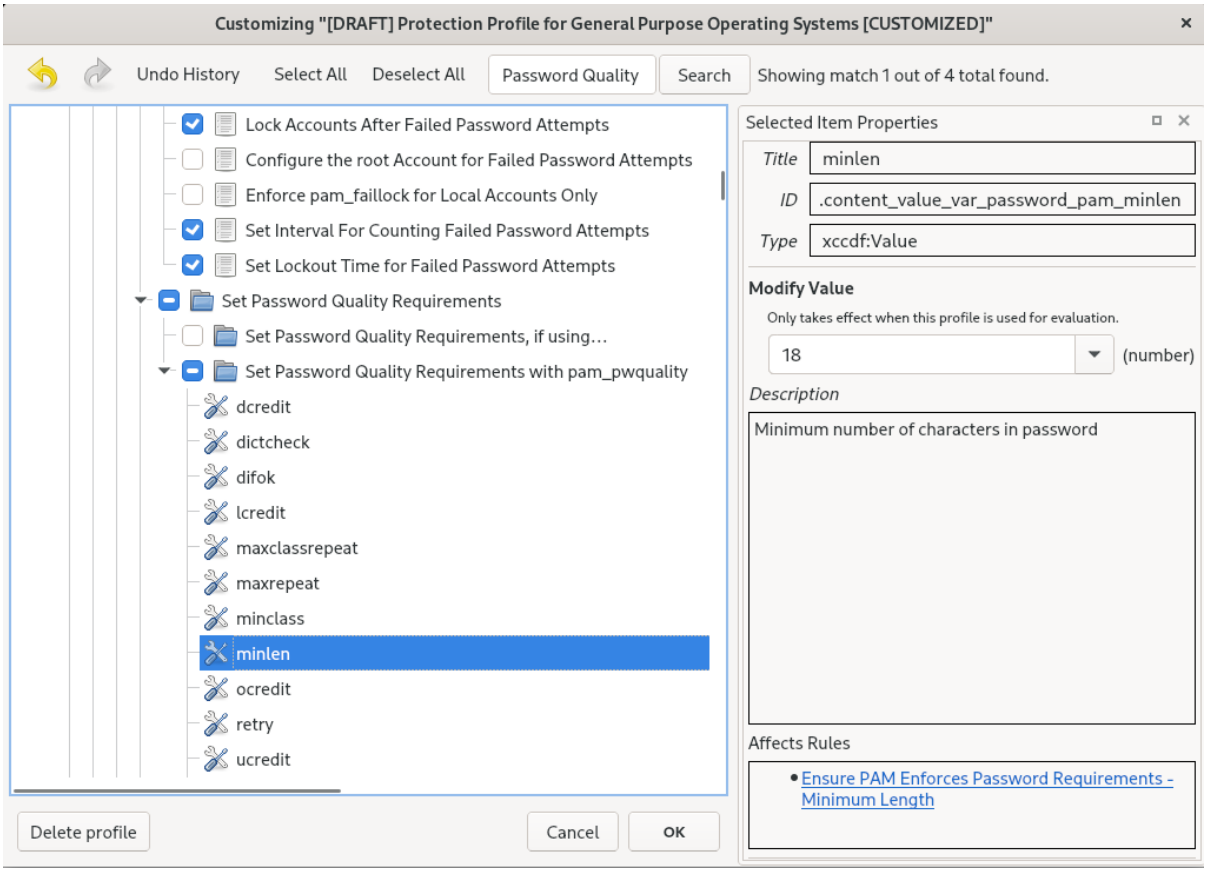
2. 要根据您的需要调整所选的安全配置文件，请点击 **Customize** 按钮。

这会打开新的 Customization 窗口，允许您在不修改原始数据流文件的情况下修改当前选择的配置文件。选择新的配置文件 ID。



3. 使用将规则组织成逻辑组的树结构或 **Search** 字段查找要修改的规则。

4. 使用树结构中的复选框来包含或排除规则，或者在适用情况下修改规则中的值。

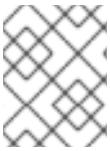


5. 点击 **OK** 按钮以确认修改。

6. 要永久存储您的修改，请使用以下选项之一：

   - 使用 **File** 菜单中的 **Save Customization Only** 分别保存自定义文件。
   - 通过在 **File** 菜单中的 **Save All** 来一次保存所有安全内容。
     如果您选择了 **Into a directory** 选项，**SCAP Workbench** 将数据流文件和自定义文件保存到指定的位置。您可以使用它作为备份解决方案。

通过选择 **As RPM** 选项，您可以指示 **SCAP Workbench** 创建包含数据流文件和自定义文件的 RPM 软件包。这对于将安全内容分发到无法远程扫描的系统以及交付内容以供进一步处理非常有用。



注意

因为 **SCAP Workbench** 不支持对定制配置文件的基于结果的补救，所以请使用 **oscap** 命令行工具导出的补救。

7.8.3. 其他资源

- **scap-workbench(8)** 手册页
- **scap-workbench** 软件包提供的 **/usr/share/doc/scap-workbench/user_manual.html** 文件
- 使用 Satellite 6.x 部署自定义 SCAP 策略 的 KCS 文章

7.9. 安装后立即部署符合安全配置文件的系统

您可以在安装过程后立即使用 OpenSCAP 套件来部署符合安全配置文件（如 OSPP、PCI-DSS 和 HIPAA 配置文件）的 RHEL 系统，。使用此部署方法，您可以使用修复脚本（例如密码强度和分区的规则）应用之后无法应用的特定规则。

7.9.1. 配置文件与 Server with GUI 不兼容

作为 **SCAP 安全指南** 的一部分提供的某些安全配置文件与 Server with GUI 基本环境中包含的扩展软件包集不兼容。因此，在安装与以下配置文件兼容的系统时，请不要选择 **Server with GUI** ：

表 7.1. 配置文件与 Server with GUI 不兼容

配置文件名称	配置文件 ID	原因	备注
[DRAFT] CIS Red Hat Enterprise Linux 9 基准（第 2 级 - 服务器）	**xccdf_org.ssgproject.content_profile_cis**	软件包 **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils** 和 **xorg-x11-server-Xwayland** 是 Server with GUI 软件包集的一部分，但该策略需要删除它们。	
[DRAFT] CIS Red Hat Enterprise Linux 9 基准（第 1 级 - 服务器）	**xccdf_org.ssgproject.content_profile_cis_server_l1**	软件包 **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils** 和 **xorg-x11-server-Xwayland** 是 Server with GUI 软件包集的一部分，但该策略需要删除它们。	

配置文件名称	配置文件 ID	原因	备注
DISA STIG for Red Hat Enterprise Linux 9	**xccdf_org.ssgproject.content_profile_stig**	软件包 **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils** 和 **xorg-x11-server-Xwayland** 是 Server with GUI 软件包集的一部分，但该策略需要删除它们。	要将 RHEL 系统安装为 Server with GUI 以与 DISA STIG 一致，您可以使用 DISA STIG with GUI profile BZ#1648162

7.9.2. 使用图形安装部署基本兼容 RHEL 系统

使用此流程部署与特定基准兼容的 RHEL 系统。这个示例为常规目的操作系统(OSPP)使用保护配置集。

警告

作为 **SCAP 安全指南** 的一部分提供的某些安全配置文件与 **Server with GUI** 基本环境中包含的扩展软件包集不兼容。如需了解更多详细信息，请参阅 与 GUI 服务器不兼容的配置文件。

前提条件

- 您已引导到 **图形** 安装程序。请注意，OSCAP Anaconda Add-on 不支持交互式文本安装。
- 您已访问 **安装概述** 窗口。

流程

1. 在 **安装概述** 窗口中点击 **软件选择**。此时会打开 **软件选择窗口**。
2. 在 **Base Environment** 窗格中选择 **服务器** 环境。您只能选择一个基本环境。
3. 点击 **完成** 应用设置并返回 **安装概述** 窗口。
4. 由于 OSPP 有必须满足的严格的分区要求，所以为 **/boot**、**/home**、**/var**、**/tmp**、**/var/log**、**/var/tmp** 和 **/var/log/audit** 创建单独的分区。
5. 点击 **安全策略**。此时会打开 **Security Policy** 窗口。
6. 要在系统中启用安全策略，将**Apply security policy** 切换为 **ON**。
7. 从配置集栏中选择 **Protection Profile for General Purpose Operating Systems**.
8. 点 **Select Profile** 来确认选择。
9. 确认在窗口底部显示 **Changes that were done or need to be done**。完成所有剩余的手动更改。

10. 完成图形安装过程。

注意

图形安装程序在安装成功后自动创建对应的 Kickstart 文件。您可以使用 **/root/anaconda-ks.cfg** 文件自动安装兼容 OSPP 的系统。

验证

- 要在安装完成后检查系统当前的状态,请重启系统并启动新的扫描：

```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

其他资源

- 配置手动分区

7.9.3. 使用 Kickstart 部署符合基线的 RHEL 系统

使用此流程部署符合特定基线的 RHEL 系统。这个示例为常规目的操作系统(OSPP)使用保护配置集。

前提条件

- **scap-security-guide** 软件包会在 RHEL 9 系统中安装。

流程

1. 在您选择的编辑器中打开 **/usr/share/scap-security-guide/kickstart/ssg-rhel9-ospp-ks.cfg** Kickstart 文件。
2. 更新分区方案以符合您的配置要求。为了实现 OSPP 合规性，必须保留 **/boot**、**/home**、**/var**、**/tmp**、**/var/log**、**/var/tmp** 和 **/var/log/audit** 的独立分区，您只能更改分区的大小。
3. 按照 使用 Kickstart 执行自动安装 中所述来开始 Kickstart 安装。

重要

对于 OSPP 的要求，不检查 Kickstart 文件中的密码。

验证

1. 要在安装完成后检查系统当前的状态,请重启系统并启动新的扫描：

```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

其他资源

- OSCAP Anaconda 附加组件

7.10. 扫描容器和容器镜像以查找漏洞

使用这个流程查找容器或容器镜像中的安全漏洞。

前提条件

- **openscap-utils** 和 **bzip2** 软件包已安装。

流程

1. 下载系统的最新 RHSA OVAL 定义：

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL9/rhel-9.oval.xml.bz2 | bzip2 --decompress > rhel-9.oval.xml
```

2. 获取容器或容器镜像的 ID，例如：

```
# podman images
REPOSITORY                          TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi9/ubi   latest   096cae65a207   7 weeks ago   239 MB
```

3. 扫描容器或容器镜像的漏洞，并将结果保存到 *vulnerability.html* 文件中：

```
# oscap-podman 096cae65a207 oval eval --report vulnerability.html rhel-9.oval.xml
```

请注意，**oscap-podman** 命令需要 root 特权，容器的 ID 是第一个参数。

验证

- 在您选择的浏览器中检查结果，例如：

```
$ firefox vulnerability.html &
```

其他资源

- 如需更多信息，请参阅 **oscap-podman(8)** 和 **oscap(8)** 手册页。

7.11. 使用特定基准评估容器或容器镜像的安全性合规

按照以下步骤，使用特定的安全基准来评估容器或容器镜像的合规性，如操作系统保护配置文件(OSPP)、支付卡行业数据安全标准(PCI-DSS)和健康保险可移植性和责任法案(HIPAA)。

前提条件

- **openscap-utils** 和 **scap-security-guide** 软件包已安装。

流程

1. 获取容器或容器镜像的 ID，例如：

```
# podman images
REPOSITORY                         TAG      IMAGE ID      CREATED       SIZE
registry.access.redhat.com/ubi9/ubi   latest   096cae65a207   7 weeks ago   239 MB
```

2. 使用 HIPAA 配置文件评估容器镜像的合规性，并将扫描结果保存到 *report.html* HTML 文件中

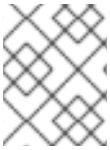
```
# oscap-podman 096cae65a207 xccdf eval --report report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

如果要评估符合 OSPP 或 PCI-DSS 基准的安全合规性，请用您容器镜像的 ID 替换 *096cae65a207*，用 *ospp* 或 *pci-dss* 替换 *hipaa* 。请注意，**oscap-podman** 命令需要 root 权限。

验证

- 在您选择的浏览器中检查结果，例如：

```
$ firefox report.html &
```



注意

标记为 *notapplicable* 的规则是不适用于容器化系统的规则。这些规则仅适用于裸机和虚拟化系统。

其他资源

- **oscap-podman(8)** 和 **scap-security-guide(8)** 手册页。
- **/usr/share/doc/scap-security-guide/** 目录。

7.12. RHEL 9 支持 SCAP 安全指南配置集

只使用 RHEL 的特定次要版本中提供的 SCAP 内容。这是因为，参与强化的组件有时会使用新功能进行更新。修改 SCAP 内容来反映这些更新，但并不总是向后兼容的。

在以下表格中，您可以找到 RHEL 9 中提供的配置集，以及与配置集匹配的策略版本。

表 7.2. RHEL 9.4 支持的 SCAP 安全指南配置文件。

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 增强级	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	2.0
法国信息系统安全部(ANSSI)BP-028 高级别	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	2.0
法国信息系统安全局(ANSSI)BP-028 中级	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	2.0

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 最低级	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	2.0
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	2022-10
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	2022-10
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis**	1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis_server_l1**	1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	1.0.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	1.0.0
[DRAFT] 在非保障信息系统和机构中未分类的信息(NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r2
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	未版本化
健康保险可移植性和责任法案 (HIPAA)	**xccdf_org.ssgproject.content_profile_hipaa**	未版本化
澳大利亚网络安全中心(ACSC)ISM 官方	**xccdf_org.ssgproject.content_profile_ism_o**	未版本化
常规目的操作系统的保护配置文件	**xccdf_org.ssgproject.content_profile_ospp**	4.3
适用于 Red Hat Enterprise Linux 9 的 PCI-DSS v3.2.1 控制基本行	**xccdf_org.ssgproject.content_profile_pci-dss**	4.0
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig**	RHEL 9.4.0:V1R2 RHEL 9.4.1 及更新版本：V1R3

配置文件名称	配置文件 ID	策略版本
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 9.4.0:V1R2 RHEL 9.4.1 及更新版本：V1R3
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	2022-10
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	2022-10

表 7.3. RHEL 9.3 中支持的 SCAP 安全指南配置文件。

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 增强级	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	2.0
法国信息系统安全部(ANSSI)BP-028 高级别	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	2.0
法国信息系统安全局(ANSSI)BP-028 中级	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	2.0
法国信息系统安全局(ANSSI)BP-028 最低级	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	2.0
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	2022-10
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	2022-10
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis**	1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis_server_l1**	1.0.0

配置文件名称	配置文件 ID	策略版本
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	1.0.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	1.0.0
[DRAFT] 在非保障信息系统和机构中未分类的信息(NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r2
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	未版本化
健康保险可移植性和责任法案 (HIPAA)	**xccdf_org.ssgproject.content_profile_hipaa**	未版本化
澳大利亚网络安全中心(ACSC)ISM 官方	**xccdf_org.ssgproject.content_profile_ism_o**	未版本化
常规目的操作系统的保护配置文件	**xccdf_org.ssgproject.content_profile_ospp**	4.3
适用于 Red Hat Enterprise Linux 9 的 PCI-DSS v3.2.1 控制基本行	**xccdf_org.ssgproject.content_profile_pci-dss**	RHEL 9.3.0 到 RHEL 9.3.2:3.2.1 RHEL 9.3.3:4.0
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig**	RHEL 9.3.0:草案[a] RHEL 9.3.2:V1R1 RHEL 9.3.3 及更新版本：V1R2
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 9.3.0:DRAFT[a] RHEL 9.3.2:V1R1 RHEL 9.3.3 及更新版本：V1R2
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	2022-10
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	2022-10

[a] DISA 尚未发布 RHEL 9 的官方基准

表 7.4. RHEL 9.2 中支持的 SCAP 安全指南配置文件

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 增强级	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	RHEL 9.2.0 到 RHEL 9.2.2:1.2 RHEL 9.2.3 及更新版本：2.0
法国信息系统安全部(ANSSI)BP-028 高级别	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	RHEL 9.2.0 到 RHEL 9.2.2:1.2 RHEL 9.2.3 及更新版本：2.0
法国信息系统安全局(ANSSI)BP-028 中级	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	RHEL 9.2.0 到 RHEL 9.2.2:1.2 RHEL 9.2.3 及更新版本：2.0
法国信息系统安全局(ANSSI)BP-028 最低级	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	RHEL 9.2.0 到 RHEL 9.2.2:1.2 RHEL 9.2.3 及更新版本：2.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis**	1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis_server_l1**	1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	1.0.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	1.0.0
[DRAFT] 在非保障信息系统和机构中未分类的信息(NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r2
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	未版本化
健康保险可移植性和责任法案 (HIPAA)	**xccdf_org.ssgproject.content_profile_hipaa**	未版本化
澳大利亚网络安全中心(ACSC)ISM 官方	**xccdf_org.ssgproject.content_profile_ism_o**	未版本化
常规目的操作系统的保护配置文件	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
适用于 Red Hat Enterprise Linux 9 的 PCI-DSS v3.2.1 控制基本行	**xccdf_org.ssgproject.content_profile_pci-dss**	RHEL 9.2.0 到 RHEL 9.2.5:3.2.1 RHEL 9.2.6 及更新版本：4.0

配置文件名称	配置文件 ID	策略版本
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig**	RHEL 9.2.0 到 RHEL 9.2.4:DRAFT[a] RHEL 9.2.5:V1R1 RHEL 9.2.6 到 RHEL 9.2.8:V1R2 RHEL 9.2.9 及更新版本：V1R3
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 9.2.0 到 9.2.4 ：DRAFT[a] RHEL 9.2.5:V1R1 RHEL 9.2.6 到 RHEL 9.2.8:V1R2 RHEL 9.2.9 及更新版本：V1R3
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	2022-10
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	2022-10

表 7.5. RHEL 9.1 支持 SCAP 安全指南配置集。

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 增强级	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	1.2
法国信息系统安全部(ANSSI)BP-028 高级别	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	1.2
法国信息系统安全局(ANSSI)BP-028 中级	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	1.2
法国信息系统安全局(ANSSI)BP-028 最低级	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	1.2
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis**	RHEL 9.1.0 和 RHEL 9.1.1:DRAFT[a] RHEL 9.1.2 及更新版本：1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 9.1.0 和 RHEL 9.1.1:DRAFT[a] RHEL 9.1.2 及更新版本：1.0.0

配置文件名称	配置文件 ID	策略版本
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 9.1.0 和 RHEL 9.1.1:DRAFT[a] RHEL 9.1.2 及更新版本：1.0.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 9.1.0 和 RHEL 9.1.1:DRAFT[a] RHEL 9.1.2 及更新版本：1.0.0
[DRAFT] 在非保障信息系统和机构中未分类的信息(NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r2
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	未版本化
健康保险可移植性和责任法案 (HIPAA)	**xccdf_org.ssgproject.content_profile_hipaa**	未版本化
澳大利亚网络安全中心(ACSC)ISM 官方	**xccdf_org.ssgproject.content_profile_ism_o**	未版本化
常规目的操作系统的保护配置文件	**xccdf_org.ssgproject.content_profile_ospp**	4.2.1
适用于 Red Hat Enterprise Linux 9 的 PCI-DSS v3.2.1 控制基本行	**xccdf_org.ssgproject.content_profile_pci-dss**	3.2.1
[草案]]用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig**	草案[a]
[草案] 用于 Red Hat Enterprise Linux 9 的带有 GUI 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig_gui**	草案[a]
[a] CIS 尚未发布 RHEL 9 的官方基准		

表 7.6. RHEL 9.0 支持的 SCAP 安全指南配置集

配置文件名称	配置文件 ID	策略版本
法国信息系统安全局(ANSSI)BP-028 增强级	**xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced**	RHEL 9.0.0 到 RHEL 9.0.10:1.2 RHEL 9.0.11 及更新版本：2.0

配置文件名称	配置文件 ID	策略版本
法国信息系统安全部(ANSSI)BP-028 高级别	**xccdf_org.ssgproject.content_profile_anssi_bp28_high**	RHEL 9.0.0 到 RHEL 9.0.10:1.2 RHEL 9.0.11 及更新版本：2.0
法国信息系统安全局(ANSSI)BP-028 中级	**xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary**	RHEL 9.0.0 到 RHEL 9.0.10:1.2 RHEL 9.0.11 及更新版本：2.0
法国信息系统安全局(ANSSI)BP-028 最低级	**xccdf_org.ssgproject.content_profile_anssi_bp28_minimal**	RHEL 9.0.0 到 RHEL 9.0.10:1.2 RHEL 9.0.11 及更新版本：2.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis**	RHEL 9.0.0 到 RHEL 9.0.6:DRAFT[a] RHEL 9.0.7 及更新版本：1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 服务器	**xccdf_org.ssgproject.content_profile_cis_server_l1**	RHEL 9.0.0 到 RHEL 9.0.6:DRAFT[a] RHEL 9.0.7 及更新版本：1.0.0
第 1 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l1**	RHEL 9.0.0 到 RHEL 9.0.6:DRAFT[a] RHEL 9.0.7 及更新版本：1.0.0
第 2 级 CIS Red Hat Enterprise Linux 9 基准 - 工作站	**xccdf_org.ssgproject.content_profile_cis_workstation_l2**	RHEL 9.0.0 到 RHEL 9.0.6:DRAFT[a] RHEL 9.0.7 及更新版本：1.0.0
[DRAFT] 在非保障信息系统和机构中未分类的信息(NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	r2
Australian Cyber Security Centre (ACSC) Essential Eight	**xccdf_org.ssgproject.content_profile_e8**	未版本化
健康保险可移植性和责任法案 (HIPAA)	**xccdf_org.ssgproject.content_profile_hipaa**	未版本化
澳大利亚网络安全中心(ACSC)ISM 官方	**xccdf_org.ssgproject.content_profile_ism_o**	未版本化
常规目的操作系统的保护配置文件	**xccdf_org.ssgproject.content_profile_ospp**	RHEL 9.0.0 到 RHEL 9.0.2:DRAFT RHEL 9.0.3 及更新版本：4.2.1
适用于 Red Hat Enterprise Linux 9 的 PCI-DSS v3.2.1 控制基本行	**xccdf_org.ssgproject.content_profile_pci-dss**	RHEL 9.0.0 到 RHEL 9.0.14:3.2.1 RHEL 9.0.15:4.0

配置文件名称	配置文件 ID	策略版本
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig**	RHEL 9.0.0 到 RHEL 9.0.13:DRAFT[a] RHEL 9.0.14:V1R1 RHEL 9.0.15 到 9.0.17:V1R2 RHEL 9.0.18 及更新版本：V1R3
用于 Red Hat Enterprise Linux 9 的国防信息系统局安全技术实施指南(DISA STIG)	**xccdf_org.ssgproject.content_profile_stig_gui**	RHEL 9.0.0 到 RHEL 9.0.13:DRAFT[a] RHEL 9.0.14:V1R1 RHEL 9.0.15 到 9.0.17:V1R2 RHEL 9.0.18 及更新版本：V1R3
CCN Red Hat Enterprise Linux 9 - 基本	**xccdf_org.ssgproject.content_profile_ccn_basic**	RHEL 9.0.11 及更新版本：2022-10
CCN Red Hat Enterprise Linux 9 - 中级	**xccdf_org.ssgproject.content_profile_ccn_intermediate**	RHEL 9.0.11 及更新版本：2022-10
CCN Red Hat Enterprise Linux 9 - 高级	**xccdf_org.ssgproject.content_profile_ccn_advanced**	RHEL 9.0.11 及更新版本：2022-10

7.13. 其他资源

- RHEL 中支持的 SCAP 安全指南版本
- OpenSCAP 项目页面 提供有关 **oscap** 工具以及与 SCAP 相关的其他组件和项目的详细信息。
- SCAP Workbench 项目页面 提供有关 **scap-workbench** 应用程序的详细信息。
- SCAP 安全指南(SSG)项目页面 为 Red Hat Enterprise Linux 提供最新的安全内容。
- 使用 OpenSCAP 进行安全合规和漏洞扫描 - 一个基于安全内容自动化协议(SCAP)标准运行工具的动手实验室，用于在 RHEL 中进行合规和漏洞扫描。
- 红帽安全演示：创建自定义安全策略内容以自动化安全合规 - 一个动手实验室，使用 RHEL 中包含的工具来获取自动化安全合规的初始经验，以符合行业标准安全策略以及自定义安全策略。如果您希望为您的团队提供培训或访问这些实验室练习，请联系您的红帽客户团队以了解更多详细信息。
- 红帽安全演示：使用 RHEL 安全技术保护自己 - 一个动手实验室，了解如何使用 RHEL 中可用的关键安全技术（包括 OpenSCAP）在所有 RHEL 系统层面上实现安全。如果您希望为您的团队提供培训或访问这些实验室练习，请联系您的红帽客户团队以了解更多详细信息。
- 美国国家标准与技术研究院(NIST) SCAP 页面 包含大量与 SCAP 相关的材料，包括 SCAP 出版物、规范和 SCAP 验证计划。
- 国家漏洞数据库(NVD) 有 SCAP 内容和其他基于 SCAP 标准的漏洞管理数据的最大存储库。
- 红帽 OVAL 内容存储库 包含对 RHEL 系统漏洞的 OVAL 定义。这是推荐的漏洞内容来源。

- MITRE CVE - 这是一个由 MITRE 公司提供的已知安全漏洞的数据库。对于 RHEL，建议您使用红帽提供的 OVAL CVE 内容。
- MITRE OVAL - 这是一个 MITRE 公司提供的与 OVAL 相关的项目。除了与 OVAL 相关的信息外，这些页面还包含 OVAL 语言和具有数千个 OVAL 定义 的 OVAL 内容存储库，。请注意，要扫描 RHEL，建议使用红帽提供的 OVAL CVE 内容。
- 在红帽 Satellite 中管理安全性合规 - 这组指南除了其他主题外，还描述了如何使用 OpenSCAP 来在多个系统上维护系统安全性。

第 8 章 使用 KEYLIME 确保系统完整性

借助 Keylime，您可以持续监控远程系统的完整性，并在引导时验证系统状态。您还可以将加密文件发送到被监控的系统，并指定监控系统失败时触发的自动化操作。

8.1. KEYLIME 的工作原理

您可以部署 Keylime 代理来执行以下操作中的一个或多个：

运行时完整性监控

Keylime 运行时完整性监控会持续监控在其上部署了代理的系统，并测量包含在 allowlist 中但不包含在 excludelist 中的文件的完整性。

测量的引导

Keylime 测量的引导会验证引导时的系统状态。

Keylime 的信任概念基于受信任的平台模块 (TPM) 技术。TPM 是一个带有集成加密密钥的硬件、固件或虚拟组件。通过轮询 TPM 参考并比较对象的哈希，Keylime 提供远程系统的初始和运行时监控。

重要

运行在虚拟机或使用虚拟 TPM 运行的 Keylime 取决于底层主机的完整性。在依赖虚拟环境中的 Keylime 测量前，请确保信任主机环境。

Keylime 由三个主要组件组成：

验证器

初始并持续验证运行代理的系统的完整性。

注册器

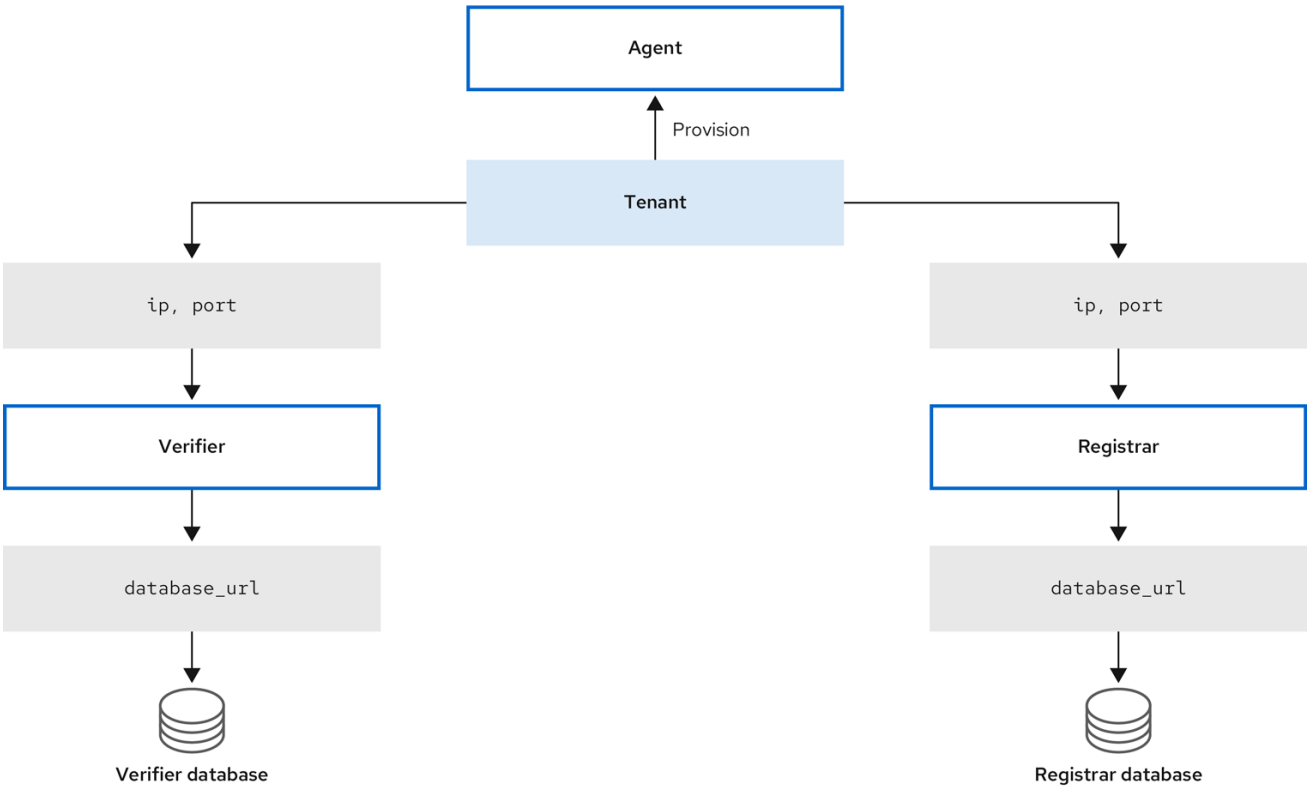
包含所有代理的数据库，它托管 TPM 供应商的公钥。

Agent

部署到由验证器测量的远程系统。

此外，Keylime 将 **keylime_tenant** 实用程序用于许多功能，包括在目标系统上调配代理。

图 8.1. 通过配置进行主精简组件之间的连接

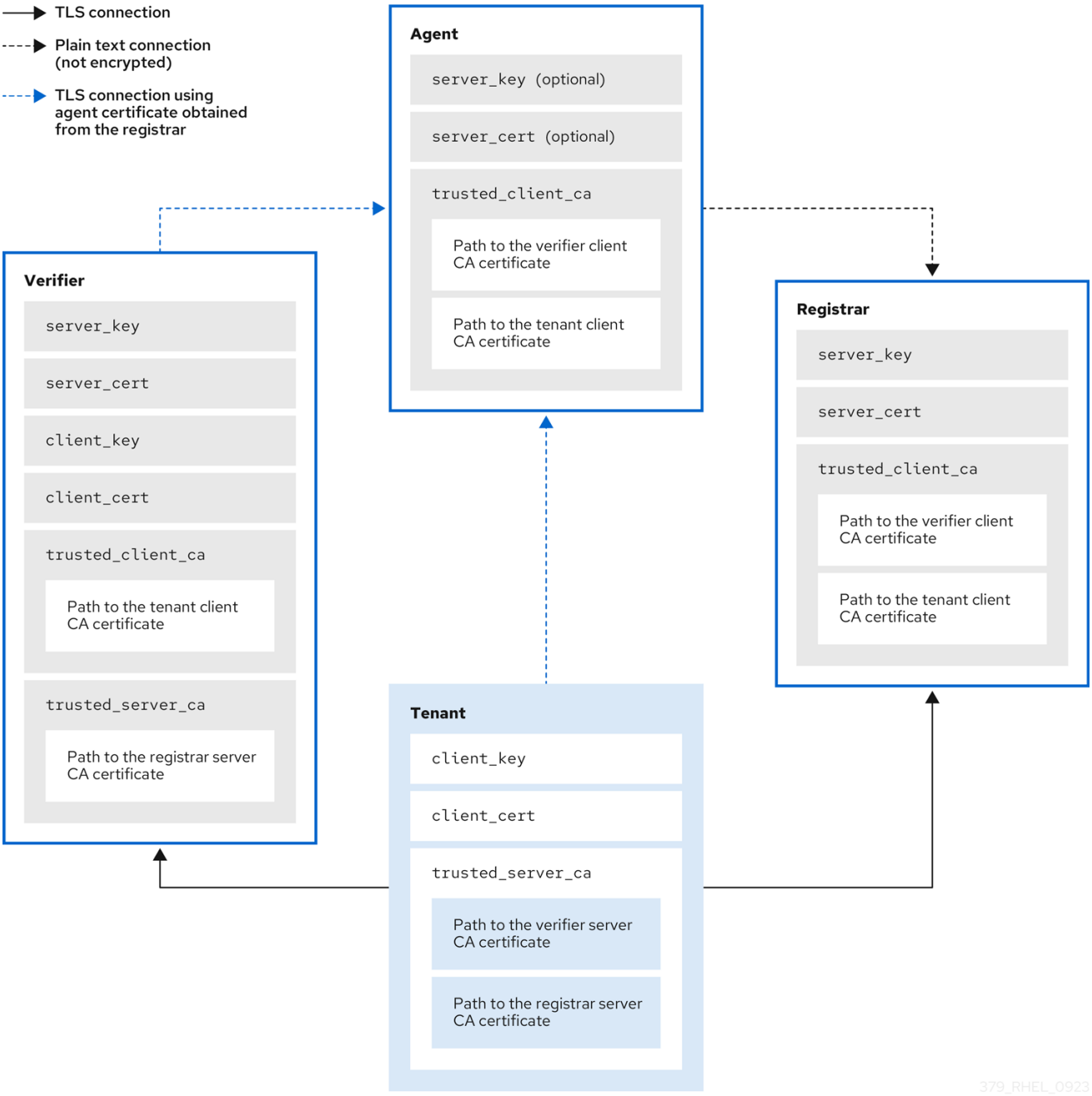


Keylime 通过使用组件与租户之间交换的密钥和证书，确保监控系统的完整性。对于此链的安全基础，请使用您可以信任的证书颁发机构(CA)。

注意

如果代理没有接收密钥和证书，它会生成一个密钥以及一个自签名证书，而无需 CA。

图 8.2. 关键精简组件证书和密钥之间的连接



8.2. 配置 KEYLIME 验证器

验证器是 Keylime 中最重要的组件。它执行系统完整性的初始和定期检查，并支持使用代理安全地引导加密密钥。验证器对其控制接口使用双向 TLS 加密。

重要

要维护信任链，请保持运行验证器的系统的安全，并在您的控制之下。

您可以根据您的要求，在单独的系统或作为 Keylime 注册器的同一系统上安装验证器。在单独的系统上运行验证器和注册器可提供更好的性能。

注意

要将配置文件保留在置入目录中，请使用具有两位数字前缀的文件名，例如 **/etc/keylime/verifier.conf.d/00-verifier-ip.conf**。配置处理以字典顺序读取置入目录中的文件，并对每个选项设置为它读取的最后一个值。

前提条件

- 您在要安装 Keylime 组件的系统上有 **root** 权限和网络连接。
- 您有来自您的证书颁发机构的有效密钥和证书。
- 可选：您可以访问 Keylime 保存来自验证器的数据的数据库。您可以使用以下数据库管理系统：
  - SQLite （默认）
  - PostgreSQL
  - MySQL
  - MariaDB

流程

1. 安装 Keylime 验证器：

```
# dnf install keylime-verifier
```

2. 通过在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件来定义验证器的 IP 地址和端口，例如：**/etc/keylime/verifier.conf.d/00-verifier-ip.conf**，其内容如下：

```
[verifier]
ip = <verifier_IP_address>
```

   - 将 ***<verifier_IP_address>*** 替换为验证器的 IP 地址。或者，使用 **ip = *** 或 **ip = 0.0.0.0** 将验证器绑定到所有可用 IP 地址。
   - 另外，您还可以使用 **port** 选项更改验证器的端口默认值 **8881** 。

3. 可选：为代理列表配置 verifier 的数据库。默认配置使用验证器的 **/var/lib/keylime/cv_data.sqlite/** 目录中的 SQLite 数据库。您可以通过在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件来定义一个不同的数据库，例如：**/etc/keylime/verifier.conf.d/00-db-url.conf**，其内容如下：

```
[verifier]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

   将 ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** 替换为数据库的 URL，如 **postgresql://verifier:UQ?nRNY9g7GZzN7@198.51.100.1/verifierdb**。

   确保您使用的凭证为 Keylime 提供了权限，来创建数据库结构。

4. 将证书和密钥添加到验证器（verifier）中。您可以让 Keylime 生成它们，或使用现有的密钥和证书生成它们：
   - 使用默认 **tls_dir = generate** 选项，Keylime 在 **/var/lib/keylime/cv_ca/** 目录中为验证器、注

册器和租户生成新证书。

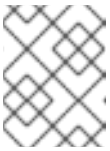
- 要在配置中加载现有的密钥和证书，请在验证器配置中定义其位置。

注意

证书必须可以被运行 Keylime 服务的 **keylime** 用户访问。

在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件，例如：**/etc/keylime/verifier.conf.d/00-keys-and-certs.conf**，其内容如下：

```
[verifier]
tls_dir = /var/lib/keylime/cv_ca
server_key = </path/to/server_key>
server_key_password = <passphrase1>
server_cert = </path/to/server_cert>
trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
client_key = </path/to/client_key>
client_key_password = <passphrase2>
client_cert = </path/to/client_cert>
trusted_server_ca = ['</path/to/ca/cert3>', '</path/to/ca/cert4>']
```



注意

使用绝对路径定义密钥和证书位置。或者，相对路径是从 **tls_dir** 选项中定义的目录解析的。

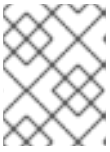
5. 在防火墙中打开端口：

```
# firewall-cmd --add-port 8881/tcp
# firewall-cmd --runtime-to-permanent
```

如果您使用其他端口，请将 **8881** 替换为 **.conf** 文件中定义的端口号。

6. 启动验证器（verifier）服务：

```
# systemctl enable --now keylime_verifier
```



注意

在默认配置中，在启动 **keylime_registrar** 服务前启动 **keylime_verifier**，因为验证器会为其他 Keylime 组件创建 CA 和证书。使用自定义证书时不需要这个顺序。

验证

- 检查 **keylime_verifier** 服务是否活跃且在运行：

```
# systemctl status keylime_verifier
● keylime_verifier.service - The Keylime verifier
    Loaded: loaded (/usr/lib/systemd/system/keylime_verifier.service; disabled; vendor preset:
disabled)
    Active: active (running) since Wed 2022-11-09 10:10:08 EST; 1min 45s ago
```

后续步骤

- 第 8.4 节 “配置 Keylime 注册器”.

8.3. 将 KEYLIME 验证器配置为容器

Keylime 验证器对系统完整性执行初始和定期检查，并支持使用代理安全地 bootstrap 加密密钥。您可以将 Keylime 验证器配置为容器，而不是 RPM 方法，而主机上无需有任何二进制文件或软件包。容器部署提供更好的隔离、模块化和 Keylime 组件的可重复性。

启动容器后，会使用默认的配置文件部署 Keylime 验证器。您可以使用一个或多个以下方法自定义配置：

- 将包含配置文件的目录挂载到容器上。这在所有 RHEL 9 版本中提供。
- 直接在容器上修改环境变量。这在 RHEL 9.3 及更新版本中提供。修改环境变量会覆盖配置文件中的值。

前提条件

- **podman** 软件包及其依赖项已安装在系统上。
- 可选：您可以访问 Keylime 保存验证器中数据的数据库。您可以使用以下数据库管理系统：
  - SQLite （默认）
  - PostgreSQL
  - MySQL
  - MariaDB
- 您有来自您的证书颁发机构的有效密钥和证书。

流程

1. 可选：安装 **keylime-verifier** 软件包，以访问配置文件。您可以配置没有此软件包的容器，但这可能更容易修改软件包提供的配置文件。

```
# dnf install keylime-verifier
```

2. 通过在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件，来将验证器绑定到所有可用 IP 地址上，例如：**/etc/keylime/verifier.conf.d/00-verifier-ip.conf**，其内容如下：

```
[verifier]
ip = *
```

   - 另外，您还可以使用 **port** 选项更改验证器的端口默认值 **8881** 。

3. 可选：为代理列表配置 verifier 的数据库。默认配置使用验证器的 **/var/lib/keylime/cv_data.sqlite/** 目录中的 SQLite 数据库。您可以通过在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件来定义一个不同的数据库，例如：**/etc/keylime/verifier.conf.d/00-db-url.conf**，其内容如下：

```
[verifier]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

将 ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** 替换为数据库的 URL，如 **postgresql://verifier:UQ?nRNY9g7GZzN7@198.51.100.1/verifierdb**。

确保您使用的凭证对 Keylime 有权限，以创建数据库结构。

4. 将证书和密钥添加到验证器（verifier）中。您可以让 Keylime 生成它们，或使用现有的密钥和证书生成它们：

   - 使用默认 **tls_dir = generate** 选项，Keylime 在 **/var/lib/keylime/cv_ca/** 目录中为验证器、注册器和租户生成新证书。
   - 要在配置中加载现有的密钥和证书，请在验证器配置中定义其位置。

   注意

   证书必须可以被运行 Keylime 进程的 **keylime** 用户访问。

   在 **/etc/keylime/verifier.conf.d/** 目录中创建一个新的 **.conf** 文件，例如：**/etc/keylime/verifier.conf.d/00-keys-and-certs.conf**，其内容如下：

   ```
   [verifier]
   tls_dir = /var/lib/keylime/cv_ca
   server_key = </path/to/server_key>
   server_cert = </path/to/server_cert>
   trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
   client_key = </path/to/client_key>
   client_cert = </path/to/client_cert>
   trusted_server_ca = ['</path/to/ca/cert3>', '</path/to/ca/cert4>']
   ```

   注意

   使用绝对路径定义密钥和证书位置。或者，相对路径是从 **tls_dir** 选项中定义的目录解析的。

5. 在防火墙中打开端口：

   ```
   # firewall-cmd --add-port 8881/tcp
   # firewall-cmd --runtime-to-permanent
   ```

   如果您使用其他端口，请将 **8881** 替换为 **.conf** 文件中定义的端口号。

6. 运行容器：

   ```
   $ podman run --name keylime-verifier \
     -p 8881:8881 \
     -v /etc/keylime/verifier.conf.d:/etc/keylime/verifier.conf.d:Z \
     -v /var/lib/keylime/cv_ca:/var/lib/keylime/cv_ca:Z \
     -d \
     -e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=<passphrase1> \
     -e KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD=<passphrase2> \
     registry.access.redhat.com/rhel9/keylime-verifier
   ```

   - **-p** 选项在主机和容器上打开默认端口 **8881**。

- **-v** 选项为目录创建到容器的绑定挂载。
  - 使用 **Z** 选项，Podman 使用私有未共享标签标记内容。这意味着只有当前容器可以使用私有卷。
- **-d** 选项在后台运行分离的容器。
- 选项 **-e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=*<passphrase1>*** 定义服务器密钥密码短语。
- 选项 **-e KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD=*<passphrase2>*** 定义客户端密钥密码短语。
- 您可以使用选项 **-e KEYLIME_VERIFIER_*<ENVIRONMENT_VARIABLE>*=*<value>***，使用环境变量覆盖配置选项。要修改附加选项，请为每个环境变量单独插入 **-e** 选项。有关环境变量及其默认值的完整列表，请参阅 第 8.12 节 “Keylime 环境变量”。

验证

- 检查容器是否正在运行：

```
$ podman ps -a
CONTAINER ID  IMAGE                                                COMMAND         CREATED
STATUS        PORTS                 NAMES
80b6b9dbf57c  registry.access.redhat.com/rhel9/keylime-verifier:latest  keylime_verifier  14
seconds ago  Up 14 seconds  0.0.0.0:8881->8881/tcp  keylime-verifier
```

后续步骤

- 第 8.5 节 “将 Keylime 注册器配置为容器”.

其他资源

- 有关 Keylime 组件的更多信息，请参阅 第 8.1 节 “Keylime 的工作原理”。
- 有关配置 Keylime 验证器的更多信息，请参阅 第 8.2 节 “配置 Keylime 验证器”。
- 有关 **podman run** 命令的更多信息，请参阅 **podman-run (1)** 手册页。

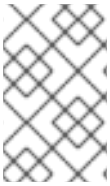
8.4. 配置 KEYLIME 注册器

注册器是包含所有代理的数据库的 Keylime 组件，它托管 TPM 供应商的公钥。在注册器的 HTTPS 服务接受可信平台模块(TPM)公钥后，它提供了一个接口来获取这些公钥以进行检查配额。

重要

要维护信任链，请保持运行注册器的系统的安全，并在您的控制之下。

您可以根据您的要求，在单独的系统或作为 Keylime 按整齐的同一系统上安装注册器。在单独的系统上运行验证器和注册器可提供更好的性能。



注意

要将配置文件保留在置入目录中，请使用具有两位数字前缀的文件名，例如 **/etc/keylime/registrar.conf.d/00-registrar-ip.conf**。配置处理以字典顺序读取置入目录中的文件，并对每个选项设置为它读取的最后一个值。

前提条件

- 您有访问安装和运行 Keylime 验证器的系统的网络权限。更多信息请参阅 第 8.2 节 “配置 Keylime 验证器”。
- 您在要安装 Keylime 组件的系统上有 **root** 权限和网络连接。
- 您可以访问 Keylime 保存来自注册中心的数据的数据库。您可以使用以下数据库管理系统：
  - SQLite （默认）
  - PostgreSQL
  - MySQL
  - MariaDB
- 您有来自您的证书颁发机构的有效密钥和证书。

流程

1. 安装 Keylime 注册器：

```
# dnf install keylime-registrar
```

2. 通过在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件来定义注册中心的 IP 地址和端口，例如：**/etc/keylime/registrar.conf.d/00-registrar-ip.conf**，其内容如下：

```
[registrar]
ip = <registrar_IP_address>
```

   - 将 ***<registrar_IP_address>*** 替换为注册中心的 IP 地址。或者，使用 **ip = *** 或 **ip = 0.0.0.0** 将注册器绑定到所有可用 IP 地址。
   - 可选，使用 **port** 选项更改 Keylime 代理连接的端口。默认值为 **8890**。
   - 可选，使用 **tls_port** 选项更改 Keylime 验证器和租户连接的 TLS 端口。默认值为 **8891**。

3. 可选：为代理列表配置注册数据库。默认配置使用 registrar 的 **/var/lib/keylime/reg_data.sqlite** 目录中的 SQLite 数据库。您可以在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/registrar.conf.d/00-db-url.conf**，其内容如下：

```
[registrar]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

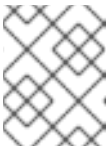
   将 ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** 替换为数据库的 URL，如 **postgresql://registrar:EKYX-bqY2?#raXm@198.51.100.1/registrardb**。

   确保您使用的凭证对 Keylime 有权限，以创建数据库结构。

4. 在注册器中添加证书和密钥：

   - 您可以使用默认配置，并将密钥和证书加载到 **/var/lib/keylime/reg_ca/** 目录中。
   - 或者，您可以在配置中定义密钥和证书的位置。在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/registrar.conf.d/00-keys-and-certs.conf**，其内容如下：

     ```
     [registrar]
     tls_dir = /var/lib/keylime/reg_ca
     server_key = </path/to/server_key>
     server_key_password = <passphrase1>
     server_cert = </path/to/server_cert>
     trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
     ```

     注意

     使用绝对路径定义密钥和证书位置。或者，您可以在 **tls_dir** 选项中定义目录，并使用相对于该目录的路径。

5. 在防火墙中打开端口：

   ```
   # firewall-cmd --add-port 8890/tcp --add-port 8891/tcp
   # firewall-cmd --runtime-to-permanent
   ```

   如果您使用其他端口，请将 **8890** 或 **8891** 替换为 **.conf** 文件中定义的端口号。

6. 启动 **keylime_registrar** 服务：

   ```
   # systemctl enable --now keylime_registrar
   ```

   注意

   在默认配置中，在启动 **keylime_registrar** 服务前启动 **keylime_verifier**，因为验证器会为其他 Keylime 组件创建 CA 和证书。使用自定义证书时不需要这个顺序。

验证

- 检查 **keylime_registrar** 服务是否活跃且在运行：

  ```
  # systemctl status keylime_registrar
  ● keylime_registrar.service - The Keylime registrar service
      Loaded: loaded (/usr/lib/systemd/system/keylime_registrar.service; disabled; vendor
  preset: disabled)
      Active: active (running) since Wed 2022-11-09 10:10:17 EST; 1min 42s ago
  ...
  ```

后续步骤

- 第 8.8 节 “配置 Keylime 租户”

8.5. 将 KEYLIME 注册器配置为容器

注册器是包含一个所有代理的数据库的 Keylime 组件，它托管可信平台模块(TPM)提供商的公钥。在注册器的 HTTPS 服务接受 TPM 公钥后，它提供一个接口来获取这些公钥，以检查引用。您可以将 Keylime 注册器配置为一个容器，而不是 RPM 方法，在主机上无需任何二进制文件或软件包。容器部署提供更好的隔离、模块化和 Keylime 组件的可重复性。

启动容器后，会使用默认配置文件部署 Keylime 注册器。您可以使用一个或多个以下方法自定义配置：

- 将包含配置文件的目录挂载到容器上。这在所有 RHEL 9 版本中提供。
- 直接在容器上修改环境变量。这在 RHEL 9.3 及更新版本中提供。修改环境变量会覆盖配置文件中的值。

前提条件

- **podman** 软件包及其依赖项已安装在系统上。
- 可选：您可以访问 Keylime 保存注册器中数据的数据库。您可以使用以下数据库管理系统：
  - SQLite （默认）
  - PostgreSQL
  - MySQL
  - MariaDB
- 您有来自您的证书颁发机构的有效密钥和证书。

流程

1. 可选：安装 **keylime-registrar** 软件包，以访问配置文件。您可以配置没有此软件包的容器，但这可能更容易修改软件包提供的配置文件。

```
# dnf install keylime-registrar
```

2. 通过在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件，来将注册器绑定到所有可用的 IP 地址，例如 **/etc/keylime/registrar.conf.d/00-registrar-ip.conf**，其内容如下：

```
[registrar]
ip = *
```

   - 可选，使用 **port** 选项更改 Keylime 代理连接的端口。默认值为 **8890**。
   - 可选，使用 **tls_port** 选项更改 Keylime 租户连接的 TLS 端口。默认值为 **8891**。

3. 可选：为代理列表配置注册数据库。默认配置使用 registrar 的 **/var/lib/keylime/reg_data.sqlite** 目录中的 SQLite 数据库。您可以在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/registrar.conf.d/00-db-url.conf**，其内容如下：

```
[registrar]
database_url =
&lt;protocol&gt;://&lt;name&gt;:&lt;password&gt;@&lt;ip_address_or_hostname&gt;/&lt;properties&gt;
```

   将 ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** 替换为数据库的 URL，如 **postgresql://registrar:EKYX-bqY2?#raXm@198.51.100.1/registrardb**。

确保您使用的凭证对 Keylime 有权限，以创建数据库结构。

4. 在注册器中添加证书和密钥：

   - 您可以使用默认配置，并将密钥和证书加载到 **/var/lib/keylime/reg_ca/** 目录中。
   - 或者，您可以在配置中定义密钥和证书的位置。在 **/etc/keylime/registrar.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/registrar.conf.d/00-keys-and-certs.conf**，其内容如下：

     ```
     [registrar]
     tls_dir = /var/lib/keylime/reg_ca
     server_key = </path/to/server_key>
     server_cert = </path/to/server_cert>
     trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
     ```

     注意

     使用绝对路径定义密钥和证书位置。或者，您可以在 **tls_dir** 选项中定义目录，并使用相对于该目录的路径。

5. 在防火墙中打开端口：

   ```
   # firewall-cmd --add-port 8890/tcp --add-port 8891/tcp
   # firewall-cmd --runtime-to-permanent
   ```

   如果您使用其他端口，请将 **8890** 或 **8891** 替换为 **.conf** 文件中定义的端口号。

6. 运行容器：

   ```
   $ podman run --name keylime-registrar \
     -p 8890:8890 \
     -p 8891:8891 \
     -v /etc/keylime/registrar.conf.d:/etc/keylime/registrar.conf.d:Z \
     -v /var/lib/keylime/reg_ca:/var/lib/keylime/reg_ca:Z \
     -d \
     -e KEYLIME_REGISTRAR_SERVER_KEY_PASSWORD=&lt;passphrase1&gt; \
     registry.access.redhat.com/rhel9/keylime-registrar
   ```

   - **-p** 选项打开主机和容器上的默认端口 **8890** 和 **8881**。
   - **-v** 选项为目录创建到容器的绑定挂载。
     - 使用 **Z** 选项，Podman 使用私有未共享标签标记内容。这意味着只有当前容器可以使用私有卷。
   - **-d** 选项在后台运行分离的容器。
   - 选项 **-e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=*<passphrase1>*** 定义服务器密钥密码短语。
   - 您可以使用选项 **-e KEYLIME_REGISTRAR_*<ENVIRONMENT_VARIABLE>*=*<value>***，使用环境变量覆盖配置选项。要修改附加选项，请为每个环境变量单独插入 **-e** 选项。有关环境变量及其默认值的完整列表，请参阅 第 8.12 节 “Keylime 环境变量”。

验证

- 检查容器是否正在运行：

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                                NAMES
07d4b4bff1b6  localhost/keylime-registrar:latest  keylime_registrar  12 seconds ago      Up 12
seconds      0.0.0.0:8881->8881/tcp, 0.0.0.0:8891->8891/tcp  keylime-registrar
```

后续步骤

- 第 8.8 节 “配置 Keylime 租户”.

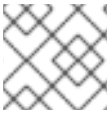
其他资源

- 有关 Keylime 组件的更多信息，请参阅 第 8.1 节 “Keylime 的工作原理”。
- 有关配置 Keylime 注册器的更多信息，请参阅 第 8.4 节 “配置 Keylime 注册器”。
- 有关 **podman run** 命令的更多信息，请参阅 **podman-run (1)** 手册页。

8.6. 使用系统角色设置 KEYLIME 服务器

您可以使用 **keylime_server** RHEL 系统角色设置验证器和注册器，它们是 Keylime 服务器组件。**keylime_server** 角色在每个节点上安装和配置 verifier 和 registrar 组件。

在 Ansible 控制节点上执行此步骤。



注意

有关 Keylime 的更多信息，请参阅 8.1。Keylime 的工作原理

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

流程

1. 创建定义所需角色的 playbook:

   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：

   ```
   # vi keylime-playbook.yml
   ```

   b. 插入以下内容：

   ```
   ---
   ```

```
- name: Manage keylime servers
  hosts: all
  vars:
    keylime_server_verifier_ip: "{{ ansible_host }}"
    keylime_server_registrar_ip: "{{ ansible_host }}"
    keylime_server_verifier_tls_dir: <ver_tls_directory>
    keylime_server_verifier_server_cert: <ver_server_certfile>
    keylime_server_verifier_server_key: <ver_server_key>
    keylime_server_verifier_server_key_passphrase: <ver_server_key_passphrase>
    keylime_server_verifier_trusted_client_ca: <ver_trusted_client_ca_list>
    keylime_server_verifier_client_cert: <ver_client_certfile>
    keylime_server_verifier_client_key: <ver_client_key>
    keylime_server_verifier_client_key_passphrase: <ver_client_key_passphrase>
    keylime_server_verifier_trusted_server_ca: <ver_trusted_server_ca_list>
    keylime_server_registrar_tls_dir: <reg_tls_directory>
    keylime_server_registrar_server_cert: <reg_server_certfile>
    keylime_server_registrar_server_key: <reg_server_key>
    keylime_server_registrar_server_key_passphrase: <reg_server_key_passphrase>
    keylime_server_registrar_trusted_client_ca: <reg_trusted_client_ca_list>
  roles:
    - rhel-system-roles.keylime_server
```

您可以在 keylime_server RHEL 系统角色的变量 中找到更多有关变量的信息。

2. 运行 playbook：

```
$ ansible-playbook <keylime-playbook.yml>
```

验证

1. 检查 **keylime_verifier** 服务是否活跃并在受管主机上运行：

```
# systemctl status keylime_verifier
● keylime_verifier.service - The Keylime verifier
    Loaded: loaded (/usr/lib/systemd/system/keylime_verifier.service; disabled; vendor preset:
disabled)
    Active: active (running) since Wed 2022-11-09 10:10:08 EST; 1min 45s ago
```

2. 检查 **keylime_registrar** 服务是否活跃且在运行：

```
# systemctl status keylime_registrar
● keylime_registrar.service - The Keylime registrar service
    Loaded: loaded (/usr/lib/systemd/system/keylime_registrar.service; disabled; vendor
preset: disabled)
    Active: active (running) since Wed 2022-11-09 10:10:17 EST; 1min 42s ago
...
```

后续步骤

第 8.8 节 “配置 Keylime 租户”

8.7. KEYLIME_SERVER RHEL 系统角色的变量

当使用 **keylime_server** RHEL 系统角色设置 Keylime 服务器时，您可以为注册器和验证器自定义以下变量。

用于配置 Keylime 验证器的 keylime_server RHEL 系统角色变量的列表

keylime_server_verifier_ip

定义 verifier 的 IP 地址。

keylime_server_verifier_tls_dir

指定存储密钥和证书的目录。如果设置为 default，verifier 将使用 **/var/lib/keylime/cv_ca** 目录。

keylime_server_verifier_server_key_passphrase

指定解密服务器私钥的密码短语。如果值为空，则私钥没有加密。

keylime_server_verifier_server_cert:指定 Keylime verifier 服务器证书文件。

keylime_server_verifier_trusted_client_ca

定义可信客户端 CA 证书的列表。您必须将文件存储在 **keylime_server_verifier_tls_dir** 选项中设置的目录中。

keylime_server_verifier_client_key

定义包含 Keylime verifier 私钥的文件。

keylime_server_verifier_client_key_passphrase

定义解密客户端私钥文件的密码短语。如果值为空，则私钥没有加密。

keylime_server_verifier_client_cert

定义 Keylime verifier 客户端证书文件。

keylime_server_verifier_trusted_server_ca

定义可信服务器 CA 证书的列表。您必须将文件存储在 **keylime_server_verifier_tls_dir** 选项中设置的目录中。

用于设置 keylime_server RHEL 系统角色的注册器变量的列表

keylime_server_registrar_ip

定义 registrar 的 IP 地址。

keylime_server_registrar_tls_dir

指定存储 registrar 密钥和证书的目录。如果将其设置为 default，registrar 将使用 **/var/lib/keylime/reg_ca** 目录。

keylime_server_registrar_server_key

定义 Keylime registrar 私有服务器密钥文件。

keylime_server_registrar_server_key_passphrase

指定解密 registrar 的服务器私钥的密码短语。如果值为空，则私钥没有加密。

keylime_server_registrar_server_cert

指定 Keylime registrar 服务器证书文件。

keylime_server_registrar_trusted_client_ca

定义可信客户端 CA 证书的列表。您必须将文件存储在 **keylime_server_registrar_tls_dir** 选项中设置的目录中。

8.8. 配置 KEYLIME 租户

Keylime 对许多功能使用 **keylime_tenant** 工具，包括在目标系统上置备代理。您可以在任何系统上安装 **keylime_tenant**，包括运行其他 Keylime 组件的系统，或者根据要求在单独的系统上安装。

前提条件

- 您在要安装 Keylime 组件的系统上有 **root** 权限和网络连接。
- 您对配置了其他 Keylime 组件的系统有网络访问权限：

  验证器

  更多信息请参阅 第 8.2 节 "配置 Keylime 验证器"。

  注册器

  更多信息请参阅 第 8.4 节 "配置 Keylime 注册器"。

流程

1. 安装 Keylime 租户：

```
# dnf install keylime-tenant
```

2. 通过编辑 **/etc/keylime/tenant.conf.d/00-verifier-ip.conf** 文件将租户的连接定义到 Keylime 验证器的连接：

```
[tenant]
verifier_ip = <verifier_ip>
```

   - 将 ***<verifier_ip>*** 替换为验证器系统的 IP 地址。
   - 如果验证器使用与默认值 **8881** 不同的端口，请添加 **verifier_port = *<verifier_port>*** 设置。

3. 通过编辑 **/etc/keylime/tenant.conf.d/00-registrar-ip.conf** 文件将租户的连接定义到 Keylime 注册器的连接：

```
[tenant]
registrar_ip = <registrar_ip>
```

   - 将 ***<registrar_ip>*** 替换为注册器系统的 IP 地址。
   - 如果注册器使用与默认值 **8891** 不同的端口，请添加 **registrar_port = *<registrar_port>*** 设置。

4. 在租户中添加证书和密钥：

   a. 您可以使用默认配置，并将密钥和证书加载到 **/var/lib/keylime/cv_ca** 目录。

   b. 或者，您可以在配置中定义密钥和证书的位置。在 **/etc/keylime/tenant.conf.d/** 目录中创建一个新的 **.conf** 文件，例如：**/etc/keylime/tenant.conf.d/00-keys-and-certs.conf**，其内容如下：

```
[tenant]
tls_dir = /var/lib/keylime/cv_ca
client_key = tenant-key.pem
```

```
client_key_password = <passphrase1>
client_cert = tenant-cert.pem
trusted_server_ca = ['</path/to/ca/cert>']
```

trusted_server_ca 参数接受到验证器和注册器服务器 CA 证书的路径。您可以提供多个以逗号分隔的路径，例如，验证器和注册器使用不同的 CA。

注意

使用绝对路径定义密钥和证书位置。或者，您可以在 **tls_dir** 选项中定义目录，并使用相对于该目录的路径。

5. 可选：如果无法使用 **/var/lib/keylime/tpm_cert_store** 目录中的证书验证可信平台模块(TPM)签署密钥(EK)，请将证书添加到该目录。这在使用具有模拟 TPM 的虚拟机时会出现这种情况。

验证

1. 检查 verifier 的状态：

```
# keylime_tenant -c cvstatus
Reading configuration from ['/etc/keylime/logging.conf']
2022-10-14 12:56:08.155 - keylime.tpm - INFO - TPM2-TOOLS Version: 5.2
Reading configuration from ['/etc/keylime/tenant.conf']
2022-10-14 12:56:08.157 - keylime.tenant - INFO - Setting up client TLS...
2022-10-14 12:56:08.158 - keylime.tenant - INFO - Using default client_cert option for tenant
2022-10-14 12:56:08.158 - keylime.tenant - INFO - Using default client_key option for tenant
2022-10-14 12:56:08.178 - keylime.tenant - INFO - TLS is enabled.
2022-10-14 12:56:08.178 - keylime.tenant - WARNING - Using default UUID d432fbb3-d2f1-4a97-9ef7-75bd81c00000
2022-10-14 12:56:08.221 - keylime.tenant - INFO - Verifier at 127.0.0.1 with Port 8881 does not have agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000.
```

如果正确设置了，且没有配置代理，验证器会响应它无法识别的默认代理 UUID。

2. 检查 registrar 的状态：

```
# keylime_tenant -c regstatus
Reading configuration from ['/etc/keylime/logging.conf']
2022-10-14 12:56:02.114 - keylime.tpm - INFO - TPM2-TOOLS Version: 5.2
Reading configuration from ['/etc/keylime/tenant.conf']
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Setting up client TLS...
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Using default client_cert option for tenant
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Using default client_key option for tenant
2022-10-14 12:56:02.137 - keylime.tenant - INFO - TLS is enabled.
2022-10-14 12:56:02.137 - keylime.tenant - WARNING - Using default UUID d432fbb3-d2f1-4a97-9ef7-75bd81c00000
2022-10-14 12:56:02.171 - keylime.registrar_client - CRITICAL - Error: could not get agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000 data from Registrar Server: 404
2022-10-14 12:56:02.172 - keylime.registrar_client - CRITICAL - Response code 404: agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000 not found
2022-10-14 12:56:02.172 - keylime.tenant - INFO - Agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000 does not exist on the registrar. Please register the agent with the registrar.
2022-10-14 12:56:02.172 - keylime.tenant - INFO - {"code": 404, "status": "Agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000 does not exist on registrar 127.0.0.1 port 8891.", "results": {}}
```

如果正确设置了，且没有配置代理，注册器会响应它无法识别的默认代理 UUID。

其他资源

- 对于 **keylime_tenant** 工具程序的额外高级选项，请输入 **keylime_tenant -h** 命令。

8.9. 配置 KEYLIME 代理

Keylime 代理是部署到被 Keylime 监控的所有系统的组件。

默认情况下，Keylime 代理将其所有数据保存在被监控系统的 **/var/lib/keylime/** 目录中。

注意

要将配置文件保留在置入目录中，请使用具有两位数字前缀的文件名，例如 **/etc/keylime/agent.conf.d/00-registrar-ip.conf**。配置处理以字典顺序读取置入目录中的文件，并对每个选项设置为它读取的最后一个值。

前提条件

- 您有对被监控系统的 **root** 权限。
- 监控的系统有一个受信任的平台模块(TPM)。

注意

要验证，请输入 **tpm2_pcrread** 命令。如果输出返回多个哈希，则代表 TPM 可用。

- 您对配置了其他 Keylime 组件的系统有网络访问权限：

  验证器

  更多信息请参阅 第 8.2 节 “配置 Keylime 验证器”。

  注册器

  更多信息请参阅 第 8.4 节 “配置 Keylime 注册器”。

  租户

  更多信息请参阅 第 8.8 节 “配置 Keylime 租户”。

- 在监控的系统上启用了完整性测量架构(IMA)。如需更多信息，请参阅 启用完整性测量架构和扩展验证模块。

流程

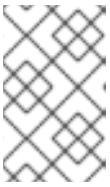
1. 安装 Keylime 代理：

```
# dnf install keylime-agent
```

   这个命令会安装 **keylime-agent-rust** 软件包。

2. 在配置文件中定义代理的 IP 地址和端口。在 **/etc/keylime/agent.conf.d/** 目录中创建一个新的 **.conf** 文件，例如：**/etc/keylime/agent.conf.d/00-agent-ip.conf**，其内容如下：

```
[agent]
ip = '<agent_ip>'
```



注意

Keylime 代理配置使用 TOML 格式，它与用于其他组件配置的 INI 格式不同。因此，使用有效 TOML 语法输入值，例如，在带单引号的路径以及带方括号的多个路径的数组。

- 将 ***<agent_IP_address>*** 替换为代理的 IP 地址。或者，使用 **ip = '*'** 或 **ip = '0.0.0.0'** 将代理绑定到所有可用 IP 地址。
- 另外，您还可以使用 **port = '*<agent_port>*'** 选项更改代理端口的默认值 **9002**。

3. 在配置文件中定义注册的 IP 地址和端口。在 **/etc/keylime/agent.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/agent.conf.d/00-registrar-ip.conf**，其内容如下：

```
[agent]
registrar_ip = '<registrar_IP_address>'
```

- 将 ***<registrar_IP_address>*** 替换为注册中心的 IP 地址。
- 另外，您还可以使用 **registrar_port = '*<registrar_port>*'** 选项更改注册器端口的默认值 **8890**。

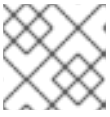
4. 可选：定义代理的通用唯一标识符(UUID)。如果没有定义，则使用默认 UUID。在 **/etc/keylime/agent.conf.d/** 目录中创建一个新的 **.conf** 文件，例如：**/etc/keylime/agent.conf.d/00-agent-uuid.conf**，其内容如下：

```
[agent]
uuid = '<agent_UUID>'
```

- 将 ***<agent_UUID>*** 替换为代理的 UUID，如 **d432fbb3-d2f1-4a97-9ef7-abcdef012345**。您可以使用 **uuidgen** 工具生成 UUID。

5. 可选：加载代理的现有密钥和证书。如果代理没有接收 **server_key** 和 **server_cert**，它会生成自己的密钥和自签名证书。
在配置中定义密钥和证书的位置。在 **/etc/keylime/agent.conf.d/** 目录中创建一个新的 **.conf** 文件，例如： **/etc/keylime/agent.conf.d/00-keys-and-certs.conf**，其内容如下：

```
[agent]
server_key = '</path/to/server_key>'
server_key_password = '<passphrase1>'
server_cert = '</path/to/server_cert>'
trusted_client_ca = '[</path/to/ca/cert3>, </path/to/ca/cert4>]'
```



注意

使用绝对路径定义密钥和证书位置。Keylime 代理不接受相对路径。

6. 在防火墙中打开端口：

```
# firewall-cmd --add-port 9002/tcp
# firewall-cmd --runtime-to-permanent
```

如果您使用不同的端口，请将 **9002** 替换为 **.conf** 文件中定义的端口号。

7. 启用并启动 **keylime_agent** 服务：

```
# systemctl enable --now keylime_agent
```

8. 可选：在配置了 Keylime 租户的系统中，验证代理是否已正确配置，并可以连接到注册器。

```
# keylime_tenant -c regstatus --uuid <agent_uuid>
Reading configuration from ['/etc/keylime/logging.conf']
...
==\n-----END CERTIFICATE-----\n", "ip": "127.0.0.1", "port": 9002, "regcount": 1,
"operational_state": "Registered"}}}
```

- 将 ***<agent_uuid>*** 替换为代理的 UUID。
  如果正确配置了注册器和代理，输出会显示代理的 IP 地址和端口，后跟 **"operational_state" : "Registered"**。

9. 通过在 **/etc/ima/ima-policy** 文件中输入以下内容来创建新的 IMA 策略：

```
# PROC_SUPER_MAGIC = 0x9fa0
dont_measure fsmagic=0x9fa0
# SYSFS_MAGIC = 0x62656572
dont_measure fsmagic=0x62656572
# DEBUGFS_MAGIC = 0x64626720
dont_measure fsmagic=0x64626720
# TMPFS_MAGIC = 0x01021994
dont_measure fsmagic=0x1021994
# RAMFS_MAGIC
dont_measure fsmagic=0x858458f6
# DEVPTS_SUPER_MAGIC=0x1cd1
dont_measure fsmagic=0x1cd1
# BINFMTFS_MAGIC=0x42494e4d
dont_measure fsmagic=0x42494e4d
# SECURITYFS_MAGIC=0x73636673
dont_measure fsmagic=0x73636673
# SELINUX_MAGIC=0xf97cff8c
dont_measure fsmagic=0xf97cff8c
# SMACK_MAGIC=0x43415d53
dont_measure fsmagic=0x43415d53
# NSFS_MAGIC=0x6e736673
dont_measure fsmagic=0x6e736673
# EFIVARFS_MAGIC
dont_measure fsmagic=0xde5e81e4
# CGROUP_SUPER_MAGIC=0x27e0eb
dont_measure fsmagic=0x27e0eb
# CGROUP2_SUPER_MAGIC=0x63677270
dont_measure fsmagic=0x63677270
# OVERLAYFS_MAGIC
# when containers are used we almost always want to ignore them
dont_measure fsmagic=0x794c7630
# MEASUREMENTS
```

```
measure func=BPRM_CHECK
measure func=FILE_MMAP mask=MAY_EXEC
measure func=MODULE_CHECK uid=0
```

此策略针对已执行的应用程序的运行时监控。您可以根据您的场景调整此策略。您可以在 **statfs (2)** 手册页中找到 MAGIC 常量。

10. 更新内核参数：

```
# grubby --update-kernel DEFAULT --args 'ima_appraise=fix ima_canonical_fmt
ima_policy=tcb ima_template=ima-ng'
```

11. 重启系统以应用新的 IMA 策略。

验证

1. 验证代理是否正在运行：

```
# systemctl status keylime_agent
● keylime_agent.service - The Keylime compute agent
     Loaded: loaded (/usr/lib/systemd/system/keylime_agent.service; enabled; preset:
disabled)
     Active: active (running) since ...
```

后续步骤

在您要监控的所有系统上配置了代理后，您可以部署 Keylime 来执行以下一个或多个功能：

- 第 8.10 节 “部署对运行时监控的 Keylime”
- 第 8.11 节 “在测试时为测量的引导部署主精简”

其他资源

- 完整性测量架构(IMA) Wiki

8.10. 部署对运行时监控的 KEYLIME

要验证受监控系统状态是否正确，Keylime 代理必须运行在监控的系统上。

重要

因为 Keylime 运行时监控使用完整性测量架构(IMA)来测量大量文件，所以可能会对系统性能产生重大影响。

在置备代理时，您还可以定义 Keylime 发送到被监控系统的文件。Keylime 对发送到代理的文件进行加密，只有在代理系统符合 TPM 策略并使用 IMA allowlist 时才解密。

您可以通过配置 Keylime excludelist 使 Keylime 忽略对特定文件或特定目录中文件的更改。排除的文件仍可由 IMA 进行测量。

从 RHEL 9.3 中提供的 Keylime 版本 7.3.0 开始，allowlis 和 excludelist 合并到 Keylime 运行时策略中。

前提条件

- 您有对配置了 Keylime 组件的系统的网络访问权限：

  验证器

  更多信息请参阅 第 8.2 节 “配置 Keylime 验证器”。

  注册器

  更多信息请参阅 第 8.4 节 “配置 Keylime 注册器”。

  租户

  更多信息请参阅 第 8.8 节 “配置 Keylime 租户”。

  Agent

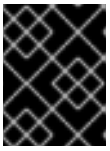
  更多信息请参阅 第 8.9 节 “配置 Keylime 代理”。

流程

1. 在配置并运行 Keylime 代理的受监控系统上，从系统的当前状态生成一个 allowlist：

```
# /usr/share/keylime/scripts/create_allowlist.sh -o <allowlist.txt> -h sha256sum
```

   将 ***<allowlist.txt>*** 替换为 allowlist 的文件名。



   重要

   使用 SHA-256 哈希功能。SHA-1 不安全，在 RHEL 9 中已弃用。如需更多信息，请参阅 Red Hat Enterprise Linux 9 中的 SHA-1 弃用 。

2. 将生成的 allowlist 复制到配置了 **keylime_tenant** 工具的系统中，例如：

```
# scp <allowlist.txt> root@<tenant.ip>:/root/<allowlist.txt>
```

3. 可选：您可以通过在租户系统上创建文件并输入要排除的文件和目录路径，来定义从 Keylime 测量中排除的文件或目录的列表。excludelist 接受 Python 正则表达式，每行一个正则表达式。有关特殊字符的完整列表，请参阅 docs.python.org 中的正则表达式操作。在租户系统上保存 excludelist。

4. 将 allowlist 和 excludelist 合并到 Keylime 运行时策略中：

```
# keylime_create_policy -a <allowlist.txt> -e <excludelist.txt> -o <policy.json>
```

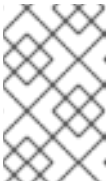
5. 在配置了 Keylime 租户的系统上，使用 **keylime_tenant** 工具置备代理：

```
# keylime_tenant -c add -t <agent_ip> -u <agent_uuid> --runtime-policy <policy.json> --cert
default
```

   - 将 ***<agent_ip>*** 替换为代理的 IP 地址。
   - 将 ***<agent_uuid>*** 替换为代理的 UUID。
   - 将 ***<policy.json>*** 替换为 Keylime 运行时策略文件的路径。
   - 使用 **--cert** 选项时，租户通过使用指定目录或默认的 **/var/lib/keylime/ca/** 目录中的 CA 证书和密钥为代理生成和签名证书。如果目录不包含 CA 证书和密钥，则租户将根据 **/etc/keylime/ca.conf** 文件中的配置自动生成它们，并将其保存到指定的目录中。然后，租户将这些密钥和证书发送给代理。

在生成 CA 证书或签名代理证书时，可能会提示您输入访问 CA 私钥的密码：**Please enter the password to decrypt your keystore:**

如果您不想使用证书，请使用 **-f** 选项，而不向代理发送文件。置备代理需要发送任何文件，即使是空文件。

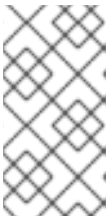


注意

Keylime 对发送到代理的文件进行加密，只有在代理系统符合 TPM 策略和 IMA allowlist 时才解密该文件。默认情况下，Keylime 解压缩发送的 **.zip** 文件。

例如，使用以下命令，**keylime_tenant** 在 **127.0.0.1** 处添加一个新的 UUID 为 **d432fbb3-d2f1-4a97-9ef7-75bd81c00000** 的 Keylime 代理，并加载运行时策略 **policy.json**。它还在默认目录中生成一个证书，并将证书文件发送给代理。只有满足 **/etc/keylime/verifier.conf** 中的 TPM 策略才对文件进行解密：

```
# keylime_tenant -c add -t 127.0.0.1 -u d432fbb3-d2f1-4a97-9ef7-75bd81c00000 --runtime-policy policy.json --cert default
```



注意

您可以使用 **# keylime_tenant -c delete -u *<agent_uuid>*** 命令停止 Keylime 监控节点。

您可以使用 **keylime_tenant -c update** 命令修改已注册的代理的配置。

验证

1. 可选：重启被监控的系统，以验证设置是否持久。
2. 验证代理是否成功：

```
# keylime_tenant -c cvstatus -u <agent.uuid>
...
{"<agent.uuid>": {"operational_state": "Get Quote"..."attestation_count": 5
...
```

将 ***<agent.uuid>*** 替换为代理的 UUID。

如果 **operational_state** 的值为 **Get Quote**，并且 **attestation_count** 为非零，则此代理的证明成功。

如果 **operational_state** 的值为 **Invalid Quote** 或 **Failed**，则验证失败，命令会显示类似如下的输出：

```
{"<agent.uuid>": {"operational_state": "Invalid Quote", ... "ima.validation.ima-ng.not_in_allowlist", "attestation_count": 5, "last_received_quote": 1684150329, "last_successful_attestation": 1684150327}}
```

3. 如果验证失败，请在验证器日志中显示更多详细信息：

```
# journalctl -u keylime_verifier
```

```
keylime.tpm - INFO - Checking IMA measurement list...
keylime.ima - WARNING - File not found in allowlist: /root/bad-script.sh
keylime.ima - ERROR - IMA ERRORS: template-hash 0 fnf 1 hash 0 good 781
keylime.cloudverifier - WARNING - agent D432FBB3-D2F1-4A97-9EF7-75BD81C00000
failed, stopping polling
```

其他资源

- 有关 IMA 的更多信息，请参阅使用内核完整性子系统增强安全性。

8.11. 在测试时为测量的引导部署主精简

当您为测量的引导验证配置 Keylime 时，Keylime 会检查测量系统上的引导过程是否与您定义的状态相符。

前提条件

- 您有对配置了 Keylime 组件的系统的网络访问权限：

  验证器

  更多信息请参阅 第 8.2 节 “配置 Keylime 验证器”。

  注册器

  更多信息请参阅 第 8.4 节 “配置 Keylime 注册器”。

  租户

  更多信息请参阅 第 8.8 节 “配置 Keylime 租户”。

  Agent

  更多信息请参阅 第 8.9 节 “配置 Keylime 代理”。

- 在代理系统上启用了统一可扩展固件接口(UEFI)。

流程

1. 在配置并运行 Keylime 代理的监控系统上，安装 **python3-keylime** 软件包，其包含 **create_mb_refstate** 脚本：

   ```
   # dnf -y install python3-keylime
   ```

2. 在被监控的系统上，使用 **create_mb_refstate** 脚本从系统当前状态的测量引导日志中生成一个策略：

   ```
   # /usr/share/keylime/scripts/create_mb_refstate
   /sys/kernel/security/tpm0/binary_bios_measurements
   <./measured_boot_reference_state.json>
   ```

   - 将 ***<./measured_boot_reference_state.json>*** 替换为保存所生成的策略的路径。
   - 如果您的 UEFI 系统没有启用安全引导，请传递 **--without-secureboot** 参数。

重要

使用 **create_mb_refstate** 脚本生成的策略基于系统的当前状态，非常严格。任何系统修改，包括内核更新和系统更新都会更改引导过程，系统就会失败。

3. 将生成的策略复制到配置了 **keylime_tenant** 工具的系统上，例如：

```
# scp root@<agent_ip>:<./measured_boot_reference_state.json>
<./measured_boot_reference_state.json>
```

4. 在配置了 Keylime 租户的系统上，使用 **keylime_tenant** 工具置备代理：

```
# keylime_tenant -c add -t <agent_ip> -u <agent_uuid> --mb_refstate
<./measured_boot_reference_state.json> --cert default
```

- 将 ***<agent_ip>*** 替换为代理的 IP 地址。
- 将 ***<agent_uuid>*** 替换为代理的 UUID。
- 将 ***<./measured_boot_reference_state.json>*** 替换为计算引导策略的路径。

如果将测量引导与运行时监控结合起来进行配置，请在输入 **keylime_tenant -c add** 命令时提供这两个用例中的所有选项。

注意

您可以使用 **# keylime_tenant -c delete -t *<agent_ip>* -u *<agent_uuid>*** 命令停止 Keylime 监控节点。

您可以使用 **keylime_tenant -c update** 命令修改已注册的代理的配置。

验证

1. 重启被监控的系统，并验证代理是否成功：

```
# keylime_tenant -c cvstatus -u <agent_uuid>
...
{"<agent.uuid>": {"operational_state": "Get Quote"..."attestation_count": 5
...
```

将 ***<agent_uuid>*** 替换为代理的 UUID。

如果 **operational_state** 的值为 **Get Quote**，并且 **attestation_count** 为非零，则此代理的证明成功。

如果 **operational_state** 的值为 **Invalid Quote** 或 **Failed**，则验证失败，命令会显示类似如下的输出：

```
{"<agent.uuid>": {"operational_state": "Invalid Quote", ... "ima.validation.ima-
ng.not_in_allowlist", "attestation_count": 5, "last_received_quote": 1684150329,
"last_successful_attestation": 1684150327}}
```

2. 如果验证失败，请在验证器日志中显示更多详细信息：

```
# journalctl -u keylime_verifier
{"d432fbb3-d2f1-4a97-9ef7-75bd81c00000": {"operational_state": "Tenant Quote Failed", ...
"last_event_id": "measured_boot.invalid_pcr_0", "attestation_count": 0,
"last_received_quote": 1684487093, "last_successful_attestation": 0}}
```

8.12. KEYLIME 环境变量

您可以设置 Keylime 环境变量来覆盖配置文件中的值，例如，当使用 **-e** 选项，使用 **podman run** 命令启动容器时。

环境变量使用以下语法：

```
KEYLIME_<SECTION>_<ENVIRONMENT_VARIABLE>=<value>
```

其中：

- ***<SECTION>*** 是 Keylime 配置文件部分。
- ***<ENVIRONMENT_VARIABLE>*** 是环境变量。
- ***<value>*** 是您要设置环境变量的值。

例如，**-e KEYLIME_VERIFIER_MAX_RETRIES=6** 将 **[verifier]** 部分中的 **max_retries** 配置选项设置为 **6**。

验证**器配置**

表 8.1. [verifier] 部分

配置选项	环境变量	默认值
auto_migrate_db	**KEYLIME_VERIFIER_AUTO_MIGRATE_DB**	**True**
client_cert	**KEYLIME_VERIFIER_CLIENT_CERT**	**default**
client_key_password	**KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD**	
client_key	**KEYLIME_VERIFIER_CLIENT_KEY**	**default**
database_pool_sz_ovfl	**KEYLIME_VERIFIER_DATABASE_POOL_SZ_OVFL**	**5,10**
database_url	**KEYLIME_VERIFIER_DATABASE_URL**	**SQLite**
durable_attestation_import	**KEYLIME_VERIFIER_DURABLE_ATTESTATION_IMPORT**	

配置选项	环境变量	默认值
enable_agent_mtls	KEYLIME_VERIFIER_ENABLE_AGENT_MTLS	True
exponential_backoff	KEYLIME_VERIFIER_EXPONENTIAL_BACKOFF	True
ignore_tomtou_errors	KEYLIME_VERIFIER_IGNORE_TOMTOU_ERRORS	False
ip	KEYLIME_VERIFIER_IP	127.0.0.1
max_retries	KEYLIME_VERIFIER_MAX_RETRIES	5
max_upload_size	KEYLIME_VERIFIER_MAX_UPLOAD_SIZE	104857600
measured_boot_evaluate	KEYLIME_VERIFIER_MEASURED_BOOT_EVALUATE	once
measured_boot_imports	KEYLIME_VERIFIER_MEASURED_BOOT_IMPORTS	[]
measured_boot_policy_name	KEYLIME_VERIFIER_MEASURED_BOOT_POLICY_NAME	accept-all
num_workers	KEYLIME_VERIFIER_NUM_WORKERS	0
persistent_store_encoding	KEYLIME_VERIFIER_PERSISTENT_STORE_ENCODING	
persistent_store_format	KEYLIME_VERIFIER_PERSISTENT_STORE_FORMAT	json
persistent_store_url	KEYLIME_VERIFIER_PERSISTENT_STORE_URL	
port	KEYLIME_VERIFIER_PORT	8881
quote_interval	KEYLIME_VERIFIER_QUOTE_INTERVAL	2
registrar_ip	KEYLIME_VERIFIER_REGISTRAR_IP	127.0.0.1

配置选项	环境变量	默认值
registrar_port	KEYLIME_VERIFIER_REGISTRAR_PORT	8891
request_timeout	KEYLIME_VERIFIER_REQUEST_TIMEOUT	60.0
require_allow_list_signatures	KEYLIME_VERIFIER_REQUIRE_ALLOW_LIST_SIGNATURES	True
retry_interval	KEYLIME_VERIFIER_RETRY_INTERVAL	2
server_cert	KEYLIME_VERIFIER_SERVER_CERT	default
server_key_password	KEYLIME_VERIFIER_SERVER_KEY_PASSWORD	default
server_key	KEYLIME_VERIFIER_SERVER_KEY	default
severity_labels	KEYLIME_VERIFIER_SEVERITY_LABELS	["info", "notice", "warning", "error", "critical", "alert", "emergency"]
severity_policy	KEYLIME_VERIFIER_SEVERITY_POLICY	[{"event_id": ".*", "severity_label" : "emergency"}]
signed_attributes	KEYLIME_VERIFIER_SIGNED_ATTRIBUTES	
time_stamp_authority_certs_path	KEYLIME_VERIFIER_TIME_STAMP_AUTHORITY_CERTS_PATH	
time_stamp_authority_url	KEYLIME_VERIFIER_TIME_STAMP_AUTHORITY_URL	
tls_dir	KEYLIME_VERIFIER_TLS_DIR	generate
transparency_log_sign_algo	KEYLIME_VERIFIER_TRANSPARENCY_LOG_SIGN_ALGO	sha256

配置选项	环境变量	默认值
transparency_log_url	**KEYLIME_VERIFIER_TRANSPARENCY_LOG_URL**	
trusted_client_ca	**KEYLIME_VERIFIER_TRUSTED_CLIENT_CA**	**default**
trusted_server_ca	**KEYLIME_VERIFIER_TRUSTED_SERVER_CA**	**default**
uuid	**KEYLIME_VERIFIER_UUID**	**default**
version	**KEYLIME_VERIFIER_VERSION**	**2.0**

表 8.2. [revocations] 部分

配置选项	环境变量	默认值
enabled_revocation_notifications	**KEYLIME_VERIFIER_REVOCATIONS_ENABLED_REVOCATION_NOTIFICATIONS**	**[agent]**
webhook_url	**KEYLIME_VERIFIER_REVOCATIONS_WEBHOOK_URL**	

注册器配置

表 8.3. [registrar] 部分

配置选项	环境变量	默认值
auto_migrate_db	**KEYLIME_REGISTRAR_AUTO_MIGRATE_DB**	**True**
database_pool_sz_ovfl	**KEYLIME_REGISTRAR_DATABASE_POOL_SZ_OVFL**	**5,10**
database_url	**KEYLIME_REGISTRAR_DATABASE_URL**	**SQLite**
durable_attestation_import	**KEYLIME_REGISTRAR_DURABLE_ATTESTATION_IMPORT**	
ip	**KEYLIME_REGISTRAR_IP**	**127.0.0.1**

persistent_store_encoding	**KEYLIME_REGISTRAR_PERSISTENT_STORE_ENCODING**	
persistent_store_format	**KEYLIME_REGISTRAR_PERSISTENT_STORE_FORMAT**	**json**
persistent_store_url	**KEYLIME_REGISTRAR_PERSISTENT_STORE_URL**	
port	**KEYLIME_REGISTRAR_PORT**	**8890**
prov_db_filename	**KEYLIME_REGISTRAR_PROV_DB_FILENAME**	**provider_reg_data.sqlite**
server_cert	**KEYLIME_REGISTRAR_SERVER_CERT**	**default**
server_key_password	**KEYLIME_REGISTRAR_SERVER_KEY_PASSWORD**	**default**
server_key	**KEYLIME_REGISTRAR_SERVER_KEY**	**default**
signed_attributes	**KEYLIME_REGISTRAR_SIGNED_ATTRIBUTES**	**ek_tpm,aik_tpm,ekcert**
time_stamp_authority_certs_path	**KEYLIME_REGISTRAR_TIME_STAMP_AUTHORITY_CERTS_PATH**	
time_stamp_authority_url	**KEYLIME_REGISTRAR_TIME_STAMP_AUTHORITY_URL**	
tls_dir	**KEYLIME_REGISTRAR_TLS_DIR**	**default**
tls_port	**KEYLIME_REGISTRAR_TLS_PORT**	**8891**
transparency_log_sign_algo	**KEYLIME_REGISTRAR_TRANSPARENCY_LOG_SIGN_ALGO**	**sha256**
transparency_log_url	**KEYLIME_REGISTRAR_TRANSPARENCY_LOG_URL**	

trusted_client_ca	**KEYLIME_REGISTRAR_TRUSTED_CLIENT_CA**	**default**
version	**KEYLIME_REGISTRAR_VERSION**	**2.0**

租户配置

表 8.4. [tenant] 部分

配置选项	环境变量	默认值
accept_tpm_encryption_algs	**KEYLIME_TENANT_ACCEPT_TPM_ENCRYPTION_ALGS**	**ecc, rsa**
accept_tpm_hash_algs	**KEYLIME_TENANT_ACCEPT_TPM_HASH_ALGS**	**sha512, sha384, sha256**
accept_tpm_signing_algs	**KEYLIME_TENANT_ACCEPT_TPM_SIGNING_ALGS**	**ecschnorr, rsassa**
client_cert	**KEYLIME_TENANT_CLIENT_CERT**	**default**
client_key_password	**KEYLIME_TENANT_CLIENT_KEY_PASSWORD**	
client_key	**KEYLIME_TENANT_CLIENT_KEY**	**default**
ek_check_script	**KEYLIME_TENANT_EK_CHECK_SCRIPT**	
enable_agent_mtls	**KEYLIME_TENANT_ENABLE_AGENT_MTLS**	**True**
exponential_backoff	**KEYLIME_TENANT_EXPONENTIAL_BACKOFF**	**True**
max_payload_size	**KEYLIME_TENANT_MAX_PAYLOAD_SIZE**	**1048576**
max_retries	**KEYLIME_TENANT_MAX_RETRIES**	**5**
mb_refstate	**KEYLIME_TENANT_MB_REFSTATE**	

registrar_ip	**KEYLIME_TENANT_REGISTRAR_IP**	**127.0.0.1**
registrar_port	**KEYLIME_TENANT_REGISTRAR_PORT**	**8891**
request_timeout	**KEYLIME_TENANT_REQUEST_TIMEOUT**	**60**
require_ek_cert	**KEYLIME_TENANT_REQUIRE_EK_CERT**	**True**
retry_interval	**KEYLIME_TENANT_RETRY_INTERVAL**	**2**
tls_dir	**KEYLIME_TENANT_TLS_DIR**	**default**
tpm_cert_store	**KEYLIME_TENANT_TPM_CERT_STORE**	**/var/lib/keylime/tpm_cert_store**
trusted_server_ca	**KEYLIME_TENANT_TRUSTED_SERVER_CA**	**default**
verifier_ip	**KEYLIME_TENANT_VERIFIER_IP**	**127.0.0.1**
verifier_port	**KEYLIME_TENANT_VERIFIER_PORT**	**8881**
version	**KEYLIME_TENANT_VERSION**	**2.0**

CA 配置

表 8.5. [CA] 部分

配置选项	环境变量	默认值
cert_bits	**KEYLIME_CA_CERT_BITS**	**2048**
cert_ca_lifetime	**KEYLIME_CA_CERT_CA_LIFETIME**	**3650**
cert_ca_name	**KEYLIME_CA_CERT_CA_NAME**	**Keylime Certificate Authority**
cert_country	**KEYLIME_CA_CERT_COUNTRY**	**US**

cert_crl_dist	KEYLIME_CA_CERT_CRL_DIST	http://localhost:38080/crl
cert_lifetime	KEYLIME_CA_CERT_LIFETIME	365
cert_locality	KEYLIME_CA_CERT_LOCALITY	Lexington
cert_org_unit	KEYLIME_CA_CERT_ORG_UNIT	53
cert_organization	KEYLIME_CA_CERT_ORGANIZATION	MITLL
cert_state	KEYLIME_CA_CERT_STATE	MA
password	KEYLIME_CA_PASSWORD	default
version	KEYLIME_CA_VERSION	2.0

代理配置

表 8.6. [agent] 部分

配置选项	环境变量	默认值
contact_ip	KEYLIME_AGENT_CONTACT_IP	127.0.0.1
contact_port	KEYLIME_AGENT_CONTACT_PORT	9002
dec_payload_file	KEYLIME_AGENT_DEC_PAYLOAD_FILE	decrypted_payload
ek_handle	KEYLIME_AGENT_EK_HANDLE	generate
enable_agent_mtls	KEYLIME_AGENT_ENABLE_AGENT_MTLS	true
enable_insecure_payload	KEYLIME_AGENT_ENABLE_INSECURE_PAYLOAD	false
enable_revocation_notifications	KEYLIME_AGENT_ENABLE_REVOCATION_NOTIFICATIONS	true

配置选项	环境变量	默认值
enc_keyname	KEYLIME_AGENT_ENC_KEYNAME	derived_tci_key
exponential_backoff	KEYLIME_AGENT_EXPONENTIAL_BACKOFF	true
extract_payload_zip	KEYLIME_AGENT_EXTRACT_PAYLOAD_ZIP	true
ip	KEYLIME_AGENT_IP	127.0.0.1
max_retries	KEYLIME_AGENT_MAX_RETRIES	4
measure_payload_pcr	KEYLIME_AGENT_MEASURE_PAYLOAD_PCR	-1
payload_script	KEYLIME_AGENT_PAYLOAD_SCRIPT	autorun.sh
port	KEYLIME_AGENT_PORT	9002
registrar_ip	KEYLIME_AGENT_REGISTRAR_IP	127.0.0.1
registrar_port	KEYLIME_AGENT_REGISTRAR_PORT	8890
retry_interval	KEYLIME_AGENT_RETRY_INTERVAL	2
revocation_actions	KEYLIME_AGENT_REVOCATION_ACTIONS	[]
revocation_cert	KEYLIME_AGENT_REVOCATION_CERT	default
revocation_notification_ip	KEYLIME_AGENT_REVOCATION_NOTIFICATION_IP	127.0.0.1
revocation_notification_port	KEYLIME_AGENT_REVOCATION_NOTIFICATION_PORT	8992
run_as	KEYLIME_AGENT_RUN_AS	keylime:tss

配置选项	环境变量	默认值
secure_size	**KEYLIME_AGENT_SECURE_SIZE**	**1m**
server_cert	**KEYLIME_AGENT_SERVER_CERT**	**default**
server_key_password	**KEYLIME_AGENT_SERVER_KEY_PASSWORD**	
server_key	**KEYLIME_AGENT_SERVER_KEY**	**default**
tls_dir	**KEYLIME_AGENT_TLS_DIR**	**default**
tpm_encryption_alg	**KEYLIME_AGENT_TPM_ENCRYPTION_ALG**	**rsa**
tpm_hash_alg	**KEYLIME_AGENT_TPM_HASH_ALG**	**sha256**
tpm_ownerpassword	**KEYLIME_AGENT_TPM_OWNERPASSWORD**	
tpm_signing_alg	**KEYLIME_AGENT_TPM_SIGNING_ALG**	**rsassa**
trusted_client_ca	**KEYLIME_AGENT_TRUSTED_CLIENT_CA**	**default**
uuid	**KEYLIME_AGENT_UUID**	**d432fbb3-d2f1-4a97-9ef7-75bd81c00000**
version	**KEYLIME_AGENT_VERSION**	**2.0**

日志记录配置

表 8.7. [logging] 部分

配置选项	环境变量	默认值
version	**KEYLIME_LOGGING_VERSION**	**2.0**

表 8.8. [loggers] 部分

配置选项	环境变量	默认值

keys	**KEYLIME_LOGGING_LOGGERS_KEYS**	**root,keylime**

表 8.9. **[handlers] 部分**

配置选项	环境变量	默认值
keys	**KEYLIME_LOGGING_HANDLERS_KEYS**	**consoleHandler**

表 8.10. **[formatters] 部分**

配置选项	环境变量	默认值
keys	**KEYLIME_LOGGING_FORMATTERS_KEYS**	**formatter**

表 8.11. **[formatter_formatter] 部分**

配置选项	环境变量	默认值
datefmt	**KEYLIME_LOGGING_FORMATTER_FORMATTER_DATEFMT**	**%Y-%m-%d %H:%M:%S**
格式	**KEYLIME_LOGGING_FORMATTER_FORMATTER_FORMAT**	**%(asctime)s.%(msecs)03d - %(name)s - %(levelname)s - %(message)s**

表 8.12. **[logger_root] 部分**

配置选项	环境变量	默认值
handlers	**KEYLIME_LOGGING_LOGGER_ROOT_HANDLERS**	**consoleHandler**
level	**KEYLIME_LOGGING_LOGGER_ROOT_LEVEL**	**INFO**

表 8.13. **[handler_consoleHandler] 部分**

配置选项	环境变量	默认值
args	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_ARGS**	**(sys.stdout,)**

class	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_CLASS**	**StreamHandler**
formatter	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_FORMATTER**	**formatter**
level	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_LEVEL**	**INFO**

表 8.14. [logger_keylime] 部分

配置选项	环境变量	默认值
handlers	**KEYLIME_LOGGING_LOGGER_KEYLIME_HANDLERS**	
level	**KEYLIME_LOGGING_LOGGER_KEYLIME_LEVEL**	**INFO**
qualname	**KEYLIME_LOGGING_LOGGER_KEYLIME_QUALNAME**	**keylime**

第 9 章 使用 AIDE 检查完整性

高级入侵检测环境(AIDE)是一个在系统上创建文件数据库的工具，然后使用该数据库来确保文件的完整性，并检测系统入侵。

9.1. 安装 AIDE

下列步骤是安装 AIDE 并启动其数据库所必需的。

前提条件

- **AppStream**存储库已启用。

流程

1. 安装 **aide** 软件包：

```
# dnf install aide
```

2. 生成一个初始数据库：

```
# aide --init
```

注意

在默认配置中，**aide --init** 命令只检查 **/etc/aide.conf** 文件中定义的一组目录和文件。要在 AIDE 数据库中包含其他目录或文件，并更改其监视的参数，请相应地编辑 **/etc/aide.conf**。

3. 要开始使用数据库，请从初始数据库文件名中删除 **.new** 子字符串：

```
# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

4. 要更改 AIDE 数据库的位置，请编辑 **/etc/aide.conf** 文件，并修改 **DBDIR** 值。要获得额外的安全性，请将数据库、配置和 **/usr/sbin/aide** 二进制文件存储在安全的位置，如只读介质。

9.2. 使用 AIDE 执行完整性检查

前提条件

- AIDE 已正确安装，其数据库已初始化。请参阅 安装 AIDE

流程

1. 启动手动检查：

```
# aide --check
Start timestamp: 2018-07-11 12:41:20 +0200 (AIDE 0.16)
AIDE found differences between database and filesystem!!
...
[trimmed for clarity]
```

2. 至少，将系统配置为每周运行 AIDE。最好每天运行 AIDE。例如，要使用 **cron** 命令计划在每天 *04:05* a.m. 执行一次 AIDE，请在 **/etc/crontab** 文件中添加以下行：

```
05 4 * * * root /usr/sbin/aide --check
```

9.3. 更新 AIDE 数据库

验证系统更改后，如软件包更新或配置文件调整，红帽建议更新您的基准 AIDE 数据库。

前提条件

- AIDE 已正确安装，其数据库已初始化。请参阅 安装 AIDE

步骤

1. 更新您的基准 AIDE 数据库：

```
# aide --update
```

aide --update 命令创建 **/var/lib/aide/aide.db.new.gz** 数据库文件。

2. 若要开始使用更新的数据库进行完整性检查，请从文件名中删除 **.new** 子字符串。

9.4. 文件完整性工具：AIDE 和 IMA

Red Hat Enterprise Linux 提供多个用于检查和维护系统上文件和目录完整性的工具。下表可帮助您决定哪个工具更适合您的场景。

表 9.1. AIDE 和 IMA 之间的比较

问题	高级入侵检测环境(AIDE)	完整性测量架构 (IMA)
什么	AIDE 是一个在系统上创建文件和目录数据库的工具。此数据库用于检查文件完整性及检测入侵检测。	IMA 通过检查与之前存储的扩展属性相比的文件度量（哈希值）来检查文件是否被修改了。
如何	AIDE 使用规则来比较文件和目录的完整性状态。	IMA 使用文件哈希值来检测入侵。
为什么	检测 - AIDE 通过验证规则来检测文件是否被修改。	检测和防止 - IMA 通过替换文件的扩展属性来检测和防止攻击。
使用	当文件或目录被修改了，AIDE 会检测到威胁。	当有人试图更改整个文件时，IMA 会检测到威胁。
扩展	AIDE 检查本地系统上文件和目录的完整性。	IMA 确保本地和远程系统的安全性。

9.5. 其他资源

- **aide(1)** 手册页

- 内核完整性子系统

第 10 章 使用 LUKS 加密块设备

通过使用磁盘加密，您可以通过对其进行加密来保护块设备上的数据。要访问设备的解密内容，请输入密码短语或密钥作为身份验证。这对移动计算机和可移动介质非常重要，因为它有助于保护设备的内容，即使它已从系统物理上移除。LUKS 格式是 Red Hat Enterprise Linux 中块设备加密的默认实现。

10.1. LUKS 磁盘加密

Linux Unified Key Setup-on-disk-format (LUKS)提供了一组简化管理加密设备的工具。使用 LUKS，您可以加密块设备，并启用多个用户密钥来解密主密钥。要批量加密分区，请使用这个主密钥。

Red Hat Enterprise Linux 使用 LUKS 执行块设备加密。默认情况下，在安装过程中不选中加密块设备的选项。如果您选择加密磁盘的选项，则系统会在每次引导计算机时都提示您输入密码短语。这个密码短语解锁解密分区的批量加密密钥。如果要修改默认分区表，您可以选择要加密的分区。这是在分区表设置中设定的。

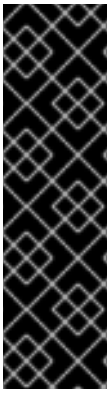
加密系统

LUKS 使用的默认密码是 **aes-xts-plain64**。LUKS 的默认密钥大小为 512 字节。 **Anaconda** XTS 模式的 LUKS 的默认密钥大小为 512 位。以下是可用的密码：

- 高级加密标准(AES)
- Twofish
- Serpent

LUKS 执行的操作

- LUKS 对整个块设备进行加密，因此非常适合保护移动设备的内容，如可移动存储介质或笔记本电脑磁盘驱动器。
- 加密块设备的底层内容是任意的，这有助于加密交换设备。对于将特殊格式化块设备用于数据存储的某些数据库，这也很有用。
- LUKS 使用现有的设备映射器内核子系统。
- LUKS 增强了密码短语，防止字典攻击。
- LUKS 设备包含多个密钥槽，这意味着您可以添加备份密钥或密码短语。



重要

在以下情况下不建议使用 LUKS：

- LUKS 等磁盘加密解决方案仅在您的系统关闭时保护数据。在系统启动并且 LUKS 已解密磁盘后，该磁盘上的文件可供有权访问它们的用户使用。
- 需要多个用户对同一设备具有不同的访问密钥的情况。LUKS1 格式提供八个密钥插槽，LUKS2 提供最多 32 个密钥插槽。
- 需要文件级加密的应用程序。

其他资源

- LUKS 项目主页

- LUKS 磁盘格式规范
- FIPS 197:高级加密标准(AES)

10.2. RHEL 中的 LUKS 版本

在 Red Hat Enterprise Linux 中，LUKS 加密的默认格式为 LUKS2。旧的 LUKS1 格式仍被完全支持，并提供与早期 Red Hat Enterprise Linux 版本兼容的格式。与 LUKS1 重新加密相比，LUKS2 重新加密被视为更加强大且更安全。

LUKS2 格式允许将来对各个部分的更新，而无需修改二进制结构。它在内部对元数据使用 JSON 文本格式，提供元数据冗余，检测元数据损坏，并从元数据副本自动修复。

重要

不要在只支持 LUKS1 的系统中使用 LUKS2。

从 Red Hat Enterprise Linux 9.2 开始，您可以使用 LUKS 版本的 **cryptsetup reencrypt** 命令加密磁盘。

在线重新加密

LUKS2 格式支持在设备正在使用时重新加密加密设备。例如：您不必卸载该设备中的文件系统来执行以下任务：

- 更改卷密钥
- 更改加密算法
  加密未加密的设备时，您仍然必须卸载文件系统。您可以在简短初始化加密后重新挂载文件系统。

  LUKS1 格式不支持在线重新加密。

转换

在某些情况下，您可以将 LUKS1 转换为 LUKS2。在以下情况下无法进行转换：

- LUKS1 设备被标记为由基于策略的解密(PBD) Clevis 解决方案使用。当检测到某些 **luksmeta** 元数据时，**cryptsetup** 工具不会转换设备。
- 设备正在活跃。在任何转换前，该设备必须处于不活跃状态。

10.3. LUKS2 重新加密过程中数据保护选项

LUKS2 提供了几个选项，在重新加密过程中优先选择性能或数据保护。它为 **resilience** 选项提供以下模式，您可以使用 **cryptsetup reencrypt --resilience Resilient-mode *dev/sdx*** 命令选择任何一种模式：

checksum

默认模式。它在数据保护和性能之间保持平衡。
这个模式将扇区的单个校验和存储在重新加密区域中，恢复进程可以检测 LUKS2 重新加密的扇区。模式要求块设备扇区写入具有"原子"性。

journal

最安全的模式，但也是最慢的模式。由于此模式将重新加密区域记录在二进制区域中，所以 LUKS2 将数据写入两次。

none

none 模式优先选择性能，不提供数据保护。它只保护数据免受安全进程终止，如 **SIGTERM** 信号或用户按 **Ctrl+C** 键。任何意外的系统故障或应用程序失败都可能会导致数据损坏。

如果 LUKS2 重新加密进程意外被强行终止，LUKU2 可通过以下方法执行恢复：

自动

在下一个 LUKS2 设备打开操作过程中，执行以下操作之一会触发自动恢复操作：

- 执行 **cryptsetup open** 命令。
- 使用 **systemd-cryptsetup** 命令附加设备。

手动

通过在 LUKS2 设备上使用 **cryptsetup repair** ***/dev/sdx*** 命令。

其他资源

- **cryptsetup-reencrypt (8)** 和 **cryptsetup-repair (8)** 手册页

10.4. 使用 LUKS2 加密块设备上的现有数据

您可以使用 LUKS2 格式加密尚未加密设备上的现有数据。新的 LUKS 标头保存在设备的标头中。

先决条件

- 块设备有一个文件系统。
- 已备份了数据。

警告

由于硬件、内核或人为故障，您可能会在加密过程中丢失数据。在开始加密数据之前，请确保您有可靠的备份。

步骤

1. 卸载您要加密的设备上的所有文件系统，例如：

```
# umount /dev/mapper/vg00-lv00
```

2. 为存储 LUKS 标头腾出空间。使用以下适合您场景的选项之一：
   - 如果是加密逻辑卷，您可以扩展逻辑卷而无需调整文件系统的大小。例如：

```
# lvextend -L+32M /dev/mapper/vg00-lv00
```

   - 使用分区管理工具（如 **parted** ）扩展分区。

- 缩小该设备的文件系统。您可以对 ext2、ext3 或 ext4 文件系统使用 **resize2fs** 工具。请注意，您无法缩小 XFS 文件系统。

3. 初始化加密：

```
# cryptsetup reencrypt --encrypt --init-only --reduce-device-size 32M /dev/mapper/vg00-lv00 lv00_encrypted

/dev/mapper/lv00_encrypted is now active and ready for online encryption.
```

4. 挂载该设备：

```
# mount /dev/mapper/lv00_encrypted /mnt/lv00_encrypted
```

5. 为到 **/etc/crypttab** 文件的持久映射添加一个条目：

   a. 查找 **luksUUID** ：

```
# cryptsetup luksUUID /dev/mapper/vg00-lv00

a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
```

   b. 在您选择的文本编辑器中打开 **/etc/crypttab**，并在此文件中添加一个设备：

```
$ vi /etc/crypttab

lv00_encrypted UUID=a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325 none
```

   将 *a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325* 替换为您设备的 **luksUUID**。

   c. 使用 **dracut** 刷新 initramfs：

```
$ dracut -f --regenerate-all
```

6. 向 **/etc/fstab** 文件中添加一个永久挂载条目：

   a. 查找活跃的 LUKS 块设备的文件系统的 UUID：

```
$ blkid -p /dev/mapper/lv00_encrypted

/dev/mapper/lv00-encrypted: UUID="37bc2492-d8fa-4969-9d9b-bb64d3685aa9" BLOCK_SIZE="4096" TYPE="xfs" USAGE="filesystem"
```

   b. 在您选择的文本编辑器中打开 **/etc/fstab**，并在此文件中添加一个设备，例如：

```
$ vi /etc/fstab

UUID=37bc2492-d8fa-4969-9d9b-bb64d3685aa9 /home auto rw,user,auto 0
```

   将 *37bc2492-d8fa-4969-9d9b-bb64d3685aa9* 替换为您文件系统的 UUID。

7. 恢复在线加密：

```
# cryptsetup reencrypt --resume-only /dev/mapper/vg00-lv00
```

```
Enter passphrase for /dev/mapper/vg00-lv00:
Auto-detected active dm device 'lv00_encrypted' for data device /dev/mapper/vg00-lv00.
Finished, time 00:31.130, 10272 MiB written, speed 330.0 MiB/s
```

验证

1. 验证现有数据是否已加密：

```
# cryptsetup luksDump /dev/mapper/vg00-lv00

LUKS header information
Version: 2
Epoch: 4
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID: a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
Label: (no label)
Subsystem: (no subsystem)
Flags: (no flags)

Data segments:
  0: crypt
 offset: 33554432 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
[...]
```

2. 查看加密的空白块设备的状态：

```
# cryptsetup status lv00_encrypted

/dev/mapper/lv00_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/mapper/vg00-lv00
```

其它资源

- **cryptsetup (8)**, **cryptsetup-reencrypt (8)**, **lvextend (8)**, **resize2fs (8)**, 和 **parted (8)** 手册页

10.5. 使用带有分离标头的 LUKS2 在块设备上加密现有数据

您可以在块设备上加密现有的数据，而无需为存储 LUKS 标头创建可用空间。标头存储在分离的位置，它也充当额外的安全层。该流程使用 LUKS2 加密格式。

先决条件

- 块设备有一个文件系统。
- 已备份了数据。

警告

由于硬件、内核或人为故障，您可能会在加密过程中丢失数据。在开始加密数据之前，请确保您有可靠的备份。

步骤

1. 卸载设备上的所有文件系统，例如：

```
# umount /dev/nvme0n1p1
```

2. 初始化加密：

```
# cryptsetup reencrypt --encrypt --init-only --header /home/header /dev/nvme0n1p1
nvme_encrypted

WARNING!
========
Header file does not exist, do you want to create it?

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /home/header:
Verify passphrase:
/dev/mapper/nvme_encrypted is now active and ready for online encryption.
```

将 */home/header* 替换为具有分离的 LUKS 标头的文件的路径。必须可以访问分离的 LUKS 标头，以便稍后解锁加密设备。

3. 挂载该设备：

```
# mount /dev/mapper/nvme_encrypted /mnt/nvme_encrypted
```

4. 恢复在线加密：

```
# cryptsetup reencrypt --resume-only --header /home/header /dev/nvme0n1p1

Enter passphrase for /dev/nvme0n1p1:
Auto-detected active dm device 'nvme_encrypted' for data device /dev/nvme0n1p1.
Finished, time 00m51s,   10 GiB written, speed 198.2 MiB/s
```

验证

1. 验证使用具有分离标头的 LUKS2 块设备上的现有数据是否已加密：

```
# cryptsetup luksDump /home/header

LUKS header information
Version:        2
Epoch:          88
Metadata area:  16384 [bytes]
```

```
Keyslots area:  16744448 [bytes]
UUID:           c4f5d274-f4c0-41e3-ac36-22a917ab0386
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
 offset: 0 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
 sector: 512 [bytes]
[...]
```

2. 查看加密的空白块设备的状态：

```
# cryptsetup status nvme_encrypted

/dev/mapper/nvme_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/nvme0n1p1
```

其他资源

- **cryptsetup (8)** 和 **cryptsetup-reencrypt (8)** 手册页

10.6. 使用 LUKS2 加密空白块设备

您可以加密空白块设备，您可以使用 LUKS2 格式将其用于加密存储。

先决条件

- 空白块设备。您可以使用 **lsblk** 等命令来查找设备上是否没有实际数据，例如，文件系统。

步骤

1. 将分区设置为加密的 LUKS 分区：

```
# cryptsetup luksFormat /dev/nvme0n1p1

WARNING!
========
This will overwrite data on /dev/nvme0n1p1 irrevocably.
Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/nvme0n1p1:
Verify passphrase:
```

2. 打开加密的 LUKS 分区：

```
# cryptsetup open /dev/nvme0n1p1 nvme0n1p1_encrypted

Enter passphrase for /dev/nvme0n1p1:
```

这会解锁分区，并使用设备映射器将其映射到新设备。为了不覆盖加密数据，这个命令会警告内核，该设备是一个加密设备，可以使用 **/dev/mapper/*device_mapped_name*** 路径通过 LUKS 解决。

3. 创建一个文件系统来将加密的数据写入分区，该分区必须可通过设备映射名称访问：

```
# mkfs -t ext4 /dev/mapper/nvme0n1p1_encrypted
```

4. 挂载该设备：

```
# mount /dev/mapper/nvme0n1p1_encrypted mount-point
```

验证

1. 验证空白块设备是否已加密：

```
# cryptsetup luksDump /dev/nvme0n1p1

LUKS header information
Version:        2
Epoch:          3
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           34ce4870-ffdf-467c-9a9e-345a53ed8a25
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
 offset: 16777216 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
 sector: 512 [bytes]
[...]
```

2. 查看加密的空白块设备的状态：

```
# cryptsetup status nvme0n1p1_encrypted

/dev/mapper/nvme0n1p1_encrypted is active and is in use.
  type:    LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/nvme0n1p1
  sector size:  512
  offset:  32768 sectors
  size:    20938752 sectors
  mode:    read/write
```

其他资源

- **cryptsetup (8)**, **cryptsetup-open (8)** 和 **cryptsetup-lusFormat (8)** 手册页

10.7. 使用 STORAGE RHEL 系统角色创建一个 LUKS2 加密的卷

您可以使用 **存储** 角色来通过运行 Ansible playbook 创建和配置使用 LUKS 加密的卷。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- name: Create and configure a volume encrypted with LUKS
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: <password>
```

   您还可以将其他加密参数（如 **encryption_key**,**encryption_cipher**,**encryption_key_size** 和 **encryption_luks**）添加到 playbook 文件中。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

   请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 查看加密状态：

```
# cryptsetup status sdb

/dev/mapper/sdb is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdb
...
```

2. 验证创建的 LUKS 加密的卷：

```
# cryptsetup luksDump /dev/sdb

Version:        2
Epoch:          6
Metadata area:  16384 [bytes]
Keyslots area:  33521664 [bytes]
UUID:           a4c6be82-7347-4a91-a8ad-9479b72c9426
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          allow-discards

Data segments:
  0: crypt
        offset: 33554432 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 4096 [bytes]
...
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录
- 使用 LUKS 加密块设备

第 11 章 使用基于策略的解密配置加密卷的自动解锁

基于策略的解密(PBD)是技术的一种集合，可在物理和虚拟上解锁加密的根和硬盘的辅助卷。PBD 使用各种解锁方法，如用户密码、受信任的平台模块(TPM)设备、连接到系统的 PKCS #11 设备，如智能卡或特殊的网络服务器。

PBD 允许将不同的解锁方法合并成一个策略，从而可以以不同的方式解锁同一个卷。RHEL 中 PBD 的当前实现由 Clevis 框架和称为 *pins* 的插件组成。每个 pin 都提供单独的解锁功能。目前，可提供以下 pins ：

tang

允许使用网络服务器解锁卷。

tpm2

允许使用 TPM2 策略解锁卷。

sss

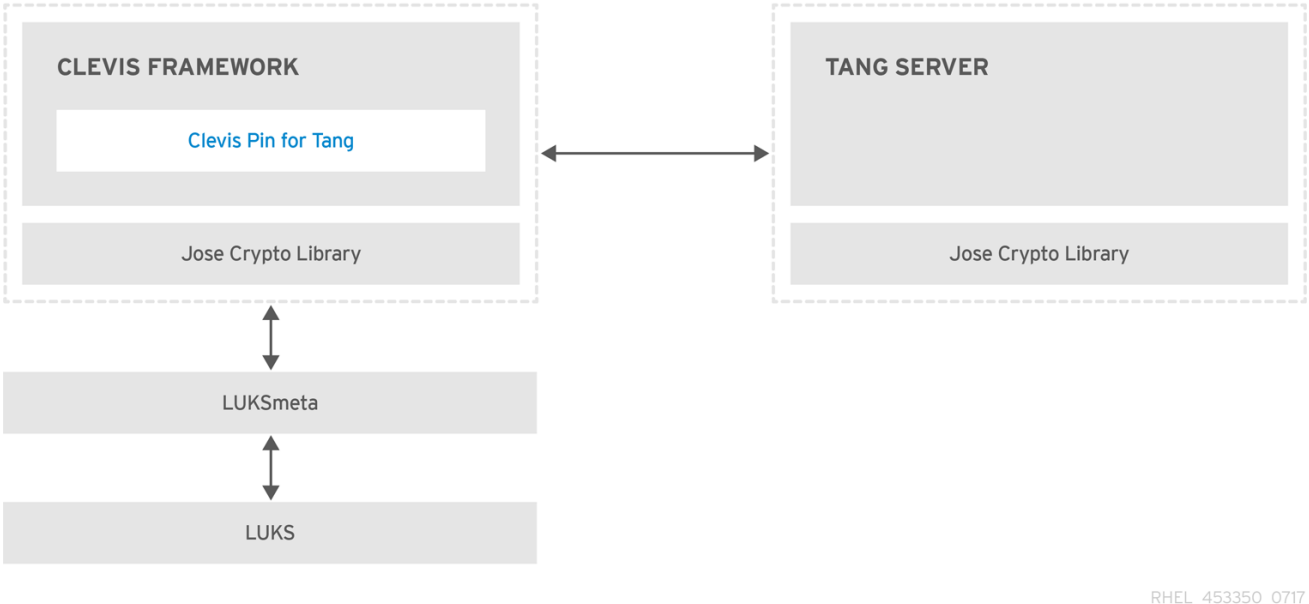
允许使用 Shamir 的 Secret Sharing (SSS)加密方案部署高可用性系统。

11.1. 网络绑定磁盘加密

网络绑定加密 (NBDE) 是基于策略的解密 (PBD) 的子类别，允许将加密的卷绑定到特殊的网络服务器。NBDE 的当前实现包括 Tang 服务器的 Clevis pin 和 Tang 服务器本身。

在 RHEL 中，NBDE 通过以下组件和技术实现：

图 11.1. 使用 LUKS1 加密的卷时的 NBDE 方案。luksmeta 软件包不用于 LUKS2 卷。



Tang 是一个将数据绑定到网络状态的服务器。当系统绑定到某个安全网络时，它会使包含数据的系统变得可用。Tang 是无状态的，不需要 TLS 或身份验证。与基于 escrow 的解决方案不同，服务器存储所有加密密钥并了解以前使用的每个密钥，Tang 从不与任何客户端密钥进行交互，因此不会从客户端获得任何识别信息。

Clevis 是一个自动化解密的可插拔框架。在 NBDE 中，Clevis 提供 LUKS 卷的自动解锁。 **clevis** 软件包提供了该功能的客户端。

Clevis pin 是 Clevis 框架的一个插件。其中一个 pins 是实现与 NBDE 服务器进行交互的插件 - Tang。

Clevis 和 Tang 是通用的客户端和服务器组件，提供网络绑定加密。在 RHEL 中，它们与 LUKS 一起使用，以加密和解密 root 和非 root 存储卷，以完成网络绑定磁盘加密。

客户端和服务器端组件都使用 *José* 库来执行加密和解密操作。

当您开始调配 NBDE 时，Tang 服务器的 Clevis pin 将获得 Tang 服务器发布的非对称密钥的列表。或者，由于密钥是非对称的，因此 Tang 的公钥列表可以分发到带外，以便客户端能够在不访问 Tang 服务器的情况下进行操作。此模式称为 *脱机调配*。

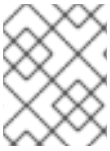
Tang 的 Clevis pin 使用其中一个公钥来生成唯一的强加密的加密密钥。使用此密钥加密数据后，密钥将被丢弃。Clevis 客户端应将此调配操作生成的状态存储在方便的位置。这种加密数据的过程就是 *调配步骤*。

LUKS 版本 2(LUKS2)是 RHEL 中的默认磁盘加密格式，因此 NBDE 的调配状态作为令牌存储在 LUKS2 标头中。**luksmeta** 软件包对 NBDE 的调配状态的利用仅用于使用 LUKS1 加密的卷。

Tang 的 Clevis pin 支持 LUKS1 和 LUKS2，不需要规范。Clevis 可以加密纯文本文件，但您必须使用 **cryptsetup** 工具来加密块设备。如需更多信息，请参阅 使用 LUKS 加密块设备。

当客户端准备好访问其数据时，它会加载再调配步骤中生成的元数据，并响应恢复加密密钥。此过程是 *恢复步骤*。

在 NBDE 中，Clevis 使用 pin 绑定 LUKS 卷，以便能自动解锁它。成功完成绑定流程后，可以使用提供的 Dracut 解锁程序解锁磁盘。



注意

如果将 **kdump** 内核崩溃转储机制设置为将系统内存的内容保存到 LUKS 加密的设备中，则会在第二次内核引导时提示您输入密码。

其他资源

- NBDE （网络绑定磁盘加密）技术知识库文章
- **tang(8)**、**clevis(1)**、**jose(1)** 和 **clevis-luks-unlockers(7)** 手册页
- 如何设置多个 LUKS 设备（Clevis + Tang 解锁）的网络绑定磁盘加密 知识库文章

11.2. 安装加密客户端 - CLEVIS

使用此流程可以在您的系统上部署并开始使用 Clevis 可插拔框架。

步骤

1. 在带有加密卷的系统上安装 Clevis 及其 pins：

```
# dnf install clevis
```

2. 要解密数据，请使用 **clevis decrypt** 命令，并提供 JSON Web 加密(JWE)格式的密码文本，例如：

```
$ clevis decrypt < secret.jwe
```

其他资源

- **cllevis(1)** 手册页
- 输入不带任何参数的 **clevis** 命令后，内置的 CLI 帮助信息：

```
$ clevis
Usage: clevis COMMAND [OPTIONS]

clevis decrypt      Decrypts using the policy defined at encryption time
clevis encrypt sss  Encrypts using a Shamir's Secret Sharing policy
clevis encrypt tang Encrypts using a Tang binding server policy
clevis encrypt tpm2 Encrypts using a TPM2.0 chip binding policy
clevis luks bind    Binds a LUKS device using the specified policy
clevis luks edit    Edit a binding from a clevis-bound slot in a LUKS device
clevis luks list    Lists pins bound to a LUKSv1 or LUKSv2 device
clevis luks pass    Returns the LUKS passphrase used for binding a particular slot.
clevis luks regen   Regenerate clevis binding
clevis luks report  Report tang keys' rotations
clevis luks unbind  Unbinds a pin bound to a LUKS volume
clevis luks unlock  Unlocks a LUKS volume
```

11.3. 部署 SELINUX 处于 ENFORCING 模式的 TANG 服务器

您可以使用 Tang 服务器在启用了 Clevis 的客户端中自动解锁 LUKS 加密卷。在最小的场景中，您可以通过安装 **tang** 软件包并输入 **systemctl enable tangd.socket --now** 命令在端口 80 上部署 Tang 服务器。以下示例流程演示了运行在自定义端口上的 Tang 服务器作为 SELinux enforcing 模式下受限制的服务的部署。

先决条件

- **policycoreutils-python-utils** 包及其依赖项已经安装。
- **firewalld** 服务正在运行。

步骤

1. 要安装 **tang** 软件包及其依赖项，请以 **root** 用户身份输入以下命令：

```
# dnf install tang
```

2. 选择一个未被占用的端口，例如 *7500/tcp*，并允许 **tangd** 服务绑定到该端口：

```
# semanage port -a -t tangd_port_t -p tcp 7500
```

请注意，一个端口只能供一个服务使用，因此试图使用一个已有的端口意味着 **ValueError：Port already defined** 错误信息。

3. 在防火墙中打开端口：

```
# firewall-cmd --add-port=7500/tcp
# firewall-cmd --runtime-to-permanent
```

4. 启用 **tangd** 服务：

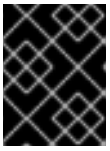
```
# systemctl enable tangd.socket
```

5. 创建覆盖文件：

```
# systemctl edit tangd.socket
```

6. 在以下编辑器屏幕中，其打开了位于 **/etc/systemd/system/tangd.socket.d/** 目录中的一个空 **override.conf** 文件，通过添加以下行将 Tang 服务器的默认端口从 80 改为之前选择的端口号：

```
[Socket]
ListenStream=
ListenStream=7500
```



重要

在以 **# Anything between here** 和 **# Lines below this** 开头的行之间插入之前的代码片段，否则系统会丢弃您的更改。

7. 按 **Ctrl+O** 并按 **Enter** 保存更改。按 **Ctrl+X** 退出编辑器。

8. 重新载入更改的配置：

```
# systemctl daemon-reload
```

9. 检查您的配置是否正常工作：

```
# systemctl show tangd.socket -p Listen
Listen=[::]:7500 (Stream)
```

10. 启动 **tangd** 服务：

```
# systemctl restart tangd.socket
```

由于 **tangd** 使用了 **systemd** 套接字激活机制，因此服务器会在第一次连接进来时就立即启动。在第一次启动时会自动生成一组新的加密密钥。要执行手动生成密钥等加密操作，请使用 **jose** 工具。

其他资源

- **tang (8)**, **semanage (8)**, **firewall-cmd (1)**, **jose (1)**, **systemd.unit (5)**, 和 **systemd.socket (5)** 书册页。

11.4. 轮转 TANG 服务器密钥并更新客户端上的绑定

使用以下步骤轮转 Tang 服务器密钥，并更新客户端上现有的绑定。轮转它们的确切间隔取决于您的应用程序、密钥大小以及机构策略。

或者，您可以使用 **nbde_server** RHEL 系统角色来轮转 Tang 密钥。如需更多信息，请参阅 使用 nbde_server 系统角色来设置多个 Tang 服务器。

先决条件

- Tang 服务器在运行。

- **clevis** 和 **clevis-luks** 软件包已安装在您的客户端上。

步骤

1. 重命名 **/var/db/tang** 密钥数据库目录中的所有密钥，使其前面有一个 **.**，将它们隐藏起来，以防被看到。请注意，以下示例中的文件名与 Tang 服务器的密钥数据库目录中的独特文件名不同：

```
# cd /var/db/tang
# ls -l
-rw-r--r--. 1 root root 349 Feb  7 14:55 UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
-rw-r--r--. 1 root root 354 Feb  7 14:55 y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
# mv UV6dqXSwe1bRKG3KbJmdiR020hY.jwk .UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
# mv y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk .y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
```

2. 检查是否重命名了，是否隐藏了 Tang 服务器中的所有密钥：

```
# ls -l
total 0
```

3. 使用 **/usr/libexec/tangd-keygen** 命令，在Tang 服务器上的 **/var/db/tang** 中生成新的密钥：

```
# /usr/libexec/tangd-keygen /var/db/tang
# ls /var/db/tang
3ZWS6-cDrCG61UPJS2BMmPU4I54.jwk zyLuX6hijUy_PSeUEFDi7hi38.jwk
```

4. 检查您的 Tang 服务器是否可以显示新密钥对的签名密钥，例如：

```
# tang-show-keys 7500
3ZWS6-cDrCG61UPJS2BMmPU4I54
```

5. 在 NBDE 客户端上，使用 **clevis luks report** 命令检查 Tang 服务器显示的密钥是否保持不变。您可以使用 **clevis luks list** 命令识别带有相关绑定的插槽，例如：

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv"}'
# clevis luks report -d /dev/sda2 -s 1
...
Report detected that some keys were rotated.
Do you want to regenerate luks metadata with "clevis luks regen -d /dev/sda2 -s 1"? [ynYN]
```

6. 要为新密钥重新生成 LUKS 元数据，在上一个命令提示时按 **y**，或使用 **clevis luks regen** 命令：

```
# clevis luks regen -d /dev/sda2 -s 1
```

7. 当您确定所有旧客户端都使用新密钥时，您可以从 Tang 服务器中删除旧密钥，例如：

```
# cd /var/db/tang
# rm .*.jwk
```

警告

在客户端仍在使用旧密钥时删除旧密钥可能会导致数据丢失。如果您意外删除了这些密钥，请在客户端上使用 **clevis luks regen** 命令，并手动提供您的 LUKS 密码。

其他资源

- **tang-show-keys(1)**、**clevis-luks-list(1)**、**clevis-luks-report(1)** 和 **clevis-luks-regen(1)** 手册页

11.5. 在 WEB 控制台中使用 TANG 密钥配置自动解锁

您可以使用 Tang 服务器提供的密钥配置 LUKS 加密存储设备的自动解锁。

先决条件

- 已安装 RHEL 9 web 控制台。详情请参阅 安装 web 控制台。
- **cockpit-storaged** 和 **clevis-luks** 软件包已安装在您的系统上。
- **cockpit.socket**服务运行在9090端口。
- Tang 服务器可用。详情请参阅 使用 SELinux enforcing 模式部署 Tang 服务器 。

步骤

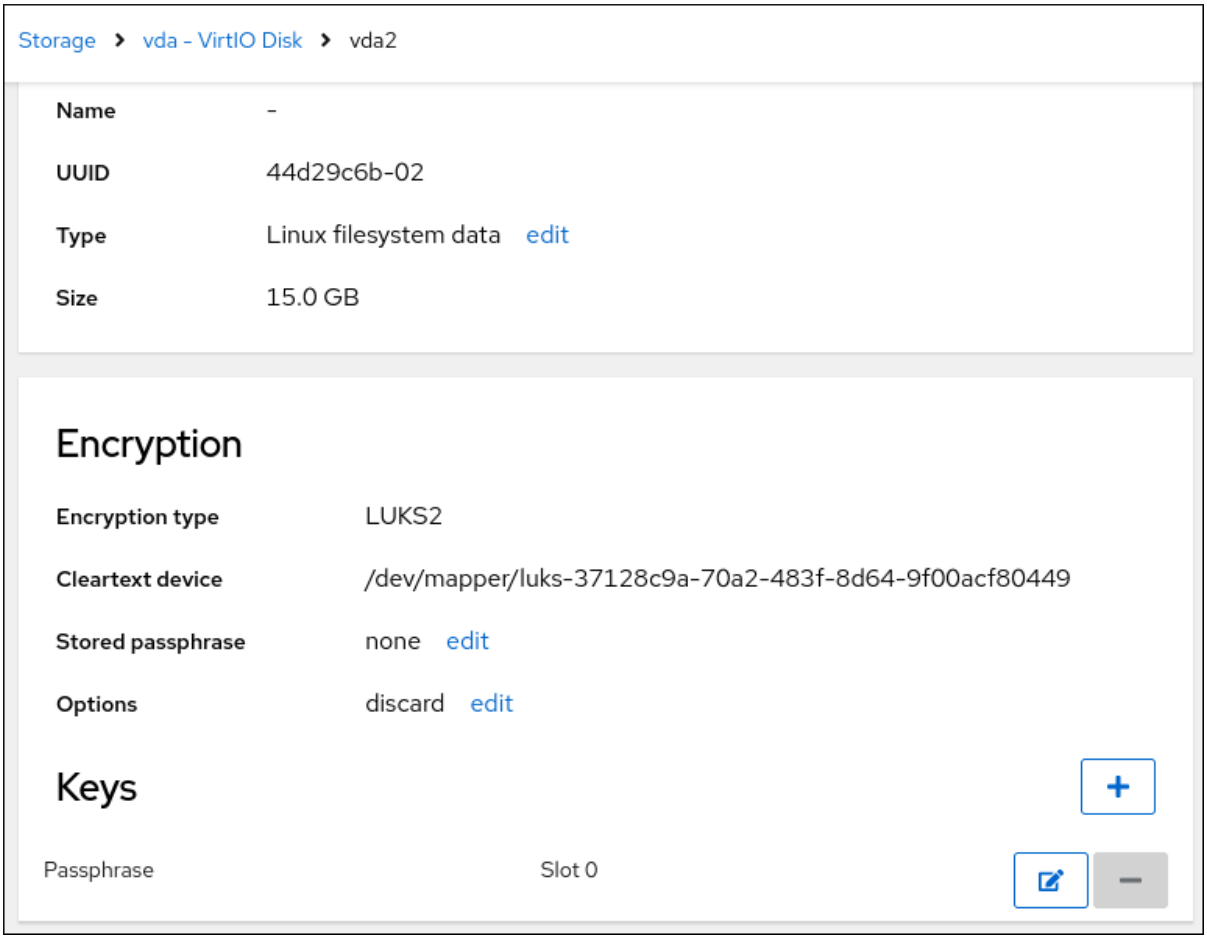
1. 在 web 浏览器中输入以下地址来打开 RHEL web 控制台：

   ```
   https://<localhost>:9090
   ```

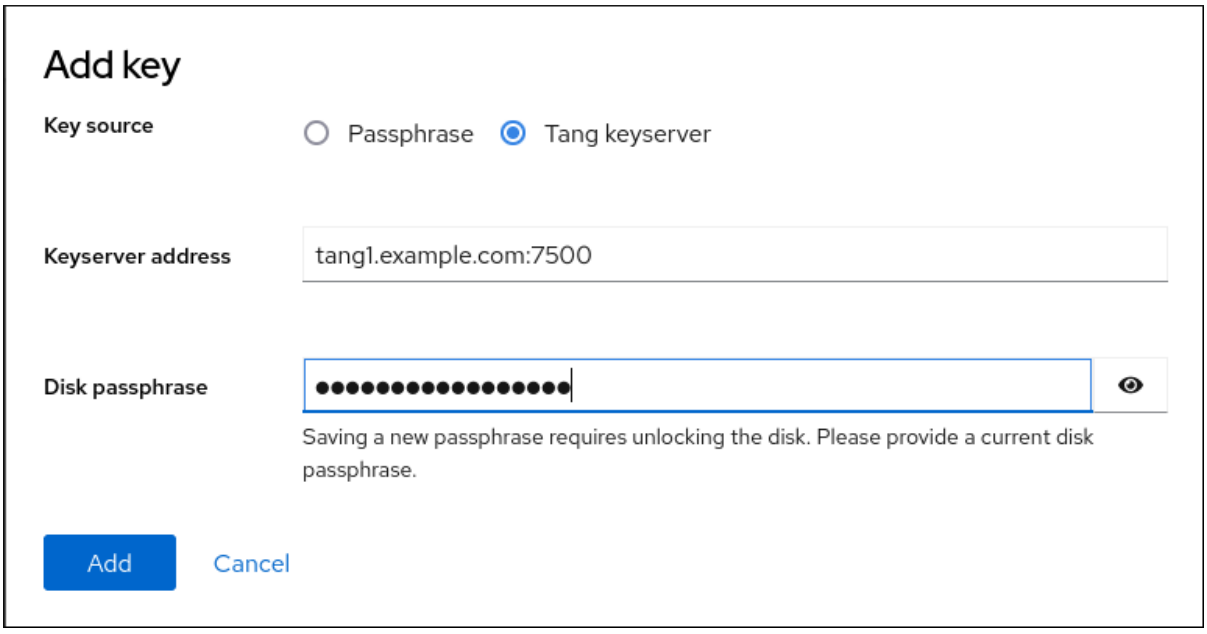
   连接到远程系统时，将 *<localhost>* 部分替换为远程服务器的主机名或 IP 地址。

2. 提供您的凭证并点击 **Storage**。在 **Storage** 表中，点包含您计划添加的加密卷的磁盘，来自动解锁。

3. 在以下巨有所选磁盘详情的页面中，点 **Keys** 部分中的 **+** 来添加 Tang 密钥：



4. 选择 **Tang keyserver** 作为 **Key source**，提供 Tang 服务器的地址，以及解锁 LUKS 加密设备的密码。点击 **Add** 确认：



以下对话框窗口提供了一个命令来验证密钥哈希是否匹配。

5. 在 Tang 服务器上的终端中，使用 **tang-show-keys** 命令来显示密钥哈希以进行比较。在本例中，Tang 服务器运行在端口 *7500* 上：

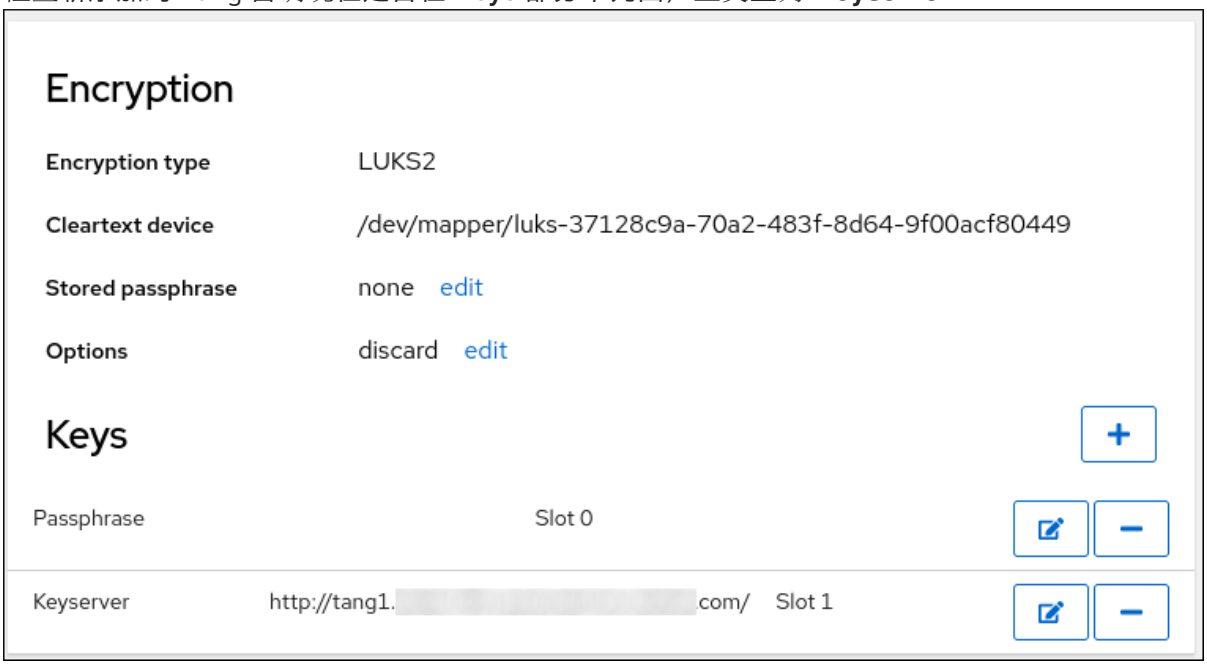
```
# tang-show-keys 7500
x100_1k6GPiDOaMlL3WbpCjHOy9ul1bSfdhI3M08wO0
```

6. 当 web 控制台中的密钥哈希与之前列出的命令的输出中的密钥哈希相同时，请点击 **Trust key**：

7. 在 RHEL 9.2 及更高版本中，选择了加密的根文件系统和 Tang 服务器后，您可以跳过向内核命令行中添加 **rd.neednet=1** 参数，安装 **clevis-dracut** 软件包，并重新生成一个初始 RAM 磁盘 (**initrd**)。对于非 root 文件系统，web 控制台现在启用 **remote-cryptsetup.target** 和 **clevis-luks-akspass.path systemd** 单元，安装 **clevis-systemd** 软件包，并将 **_netdev** 参数添加到 **fstab** 和 **crypttab** 配置文件中。

验证

1. 检查新添加的 Tang 密钥现在是否在 **Keys** 部分中列出，且类型为 **Keyserver** ：



2. 验证绑定是否在早期引导时可用，例如：

```
# lsinitrd | grep clevis-luks
lrwxrwxrwx   1 root     root          48 Jan  4 02:56
etc/systemd/system/cryptsetup.target.wants/clevis-luks-askpass.path ->
/usr/lib/systemd/system/clevis-luks-askpass.path
…
```

-

其他资源

- 使用 RHEL web 控制台入门

11.6. 基本的 NBDE 和 TPM2 加密客户端操作

Clevis 框架可以加密纯文本文件，并解密 JSON Web 加密(JWE)格式的密文和 LUKS 加密的块设备。Clevis 客户端可以使用 Tang 网络服务器或受信任的平台模块 2.0(TPM 2.0)芯片进行加密操作。

以下命令通过包含纯文本文件的示例演示了 Clevis 提供的基本功能。您还可以使用它们来对 NBDE 或 Clevis+TPM 部署进行故障排除。

绑定到 Tang 服务器的加密客户端

- 要检查 Clevis 加密客户端是否绑定到 Tang 服务器，请使用 **clevis encrypt tang** 子命令：

```
$ clevis encrypt tang '{"url":"http://tang.srv:port"}' < input-plain.txt > secret.jwe
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
```

更改上例中的 ***http://tang.srv:port*** URL，使其与安装了 **tang** 的服务器的 URL 匹配。 ***secret.jwe*** 输出文件包含您加密的密码文本，格式为 JWE。这个密码文本是从 ***input-plain.txt*** 输入文件中读取的。

另外，如果您的配置需要与 Tang 服务器进行非互动通信而无需 SSH 访问，您可以下载公告并将其保存到文件中：

```
$ curl -sfg http://tang.srv:port/adv -o adv.jws
```

对任何以下任务（如加密文件或消息）使用 ***adv.jws*** 文件中的公告：

```
$ echo 'hello' | clevis encrypt tang '{"url":"http://tang.srv:port","adv":"adv.jws"}'
```

- 要解密数据，请使用 **clevis decrypt** 命令，并提供密码文本(JWE)：

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

使用 TPM 2.0 加密客户端

- 要使用 TPM 2.0 芯片进行加密，请使用 **clevis encrypt tpm2** 子命令，唯一的参数是 JSON 配置对象：

```
$ clevis encrypt tpm2 '{}' < input-plain.txt > secret.jwe
```

要选择不同的层次结构、哈希和密钥算法，请指定配置属性，例如：

```
$ clevis encrypt tpm2 '{"hash":"sha256","key":"rsa"}' < input-plain.txt > secret.jwe
```

- 要解密数据，请提供 JSON Web 加密(JWE)格式的密码文本：

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

pin 还支持将数据封装到平台配置寄存器(PCR)状态。这样，只有 PCR 哈希值与密封时使用的策略匹配，数据才能被解封。

例如，对于 SHA-256 块要将数据密封到索引为 0 和 7 的 PCR：

```
$ clevis encrypt tpm2 '{"pcr_bank":"sha256","pcr_ids":"0,7"}' < input-plain.txt > secret.jwe
```

警告

PCR 中的哈希值可以重写，您无法再解锁加密的卷。因此，添加一个强大的密码短语，以便您手动解锁加密的卷，即使 PCR 中的值发生了变化。

如果在升级 **shim-x64** 软件包后系统无法自动解锁加密的卷，请遵照 重启后，Clevis TPM2 不再解密 LUKS 设备 KCS 文章中的步骤进行操作。

其他资源

- **clevis-encrypt-tang(1)**、**clevis-luks-unlockers(7)**、**clevis(1)** 和 **clevis-encrypt-tpm2(1)** 手册页
- 不带任何参数的 **clevis**、**clevis decrypt** 和 **clevis encrypt tang** 命令会显示内置 CLI 帮助信息，例如：

```
$ clevis encrypt tang
Usage: clevis encrypt tang CONFIG < PLAINTEXT > JWE
...
```

11.7. 配置 LUKS 加密卷的手动注册

使用 Clevis 框架，您可以在所选 Tang 服务器可用时配置客户端来自动解锁 LUKS 加密卷。这会创建一个 NBDE （网络绑定磁盘加密）部署。

先决条件

- Tang 服务器正在运行且可用。

步骤

1. 要自动解锁现有的 LUKS 加密卷，请安装 **clevis-luks** 子软件包：

```
# dnf install clevis-luks
```

2. 识别 PBD 的 LUKS 加密卷。在以下示例中，块设备是指 */dev/sda2* ：

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. 使用 **clevis luks bind** 命令将卷绑定到 Tang 服务器：

```
# clevis luks bind -d /dev/sda2 tang '{"url":"http://tang.srv"}'
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
You are about to initialize a LUKS device for metadata storage.
Attempting to initialize it may result in data loss if data was
already written into the LUKS header gap in a different format.
A backup is advised before initialization is performed.

Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

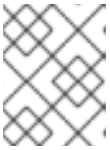
此命令执行四个步骤：

a. 使用与 LUKS 主密钥相同的无序状态测量法创建新的密钥。

b. 使用 Clevis 加密新密钥.

c. 将 Clevis JWE 对象存储在 LUKS2 标头令牌中，或者使用 LUKSMeta（如果使用非默认的 LUKS1 标头）。

d. 启用与 LUKS 一起使用的新密钥。



注意

绑定过程假定至少有一个可用的 LUKS 密码插槽。**clevis luks bind** 命令占用了其中一个插槽。

现在可以使用您的现有密码和 Clevis 策略来解锁卷。

4. 要启用早期引导系统来处理磁盘绑定，请在已安装的系统上使用 **dracut** 工具：

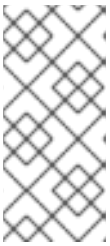
```
# dnf install clevis-dracut
```

在 RHEL 中，Clevis 生成没有特定于主机配置选项的通用 **initrd**（初始 RAM 磁盘），且不会自动将 **rd.neednet=1** 等参数添加到内核命令行。如果您的配置依赖于在早期引导期间需要网络的 Tang pin，请在检测到 Tang 绑定时使用 **--hostonly-cmdline** 参数和 **dracut** add **rd.neednet=1**：

```
# dracut -fv --regenerate-all --hostonly-cmdline
```

或者，在 **/etc/dracut.conf.d/ 中创建一个 .conf** 文件，并将 **hostonly_cmdline=yes** 选项添加到该文件中，例如：

```
# echo "hostonly_cmdline=yes" > /etc/dracut.conf.d/clevis.conf
```



注意

您还可以通过使用安装了 Clevis 的系统上的 **grubby** 工具，确保在早期引导时 Tang pin 的网络可用：

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

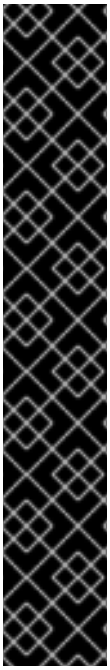
然后您可以使用不带 **--hostonly-cmdline** 的 **dracut** ：

```
# dracut -fv --regenerate-all
```

验证

1. 要验证 Clevis JWE 对象是否已成功放入 LUKS 标头中，请使用 **clevis luks list** 命令：

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv:port"}'
```



重要

要将 NBDE 用于使用静态 IP 配置（没有 DHCP）的客户端，请手动将网络配置传给 **dracut** 工具，例如：

```
# dracut -fv --regenerate-all --kernel-cmdline
"ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none nameserver=192.0.2.100"
```

或者，在 **/etc/dracut.conf.d/** 目录中创建 一个带有静态网络信息的.conf 文件。例如：

```
# cat /etc/dracut.conf.d/static_ip.conf
kernel_cmdline="ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none
nameserver=192.0.2.100"
```

重新生成初始 RAM 磁盘镜像：

```
# dracut -fv --regenerate-all
```

其他资源

- **clevis-luks-bind(1)** 和 **dracut.cmdline(7)** 手册页
- 网络引导选项
- 在初始 ramdisk (initrd)中查找 Linux 网络配置

11.8. 使用 TPM 2.0 策略配置 LUKS 加密的卷的手动注册

使用以下步骤，使用受信任的平台模块 2.0(TPM 2.0)策略来配置 LUKS 加密卷的解锁。

先决条件

- 一个可访问的 TPM 2.0 兼容设备。
- 具有 64 位 Intel 或 64 位 AMD 架构的系统。

步骤

1. 要自动解锁现有的 LUKS 加密卷，请安装 **clevis-luks** 子软件包：

```
# dnf install clevis-luks
```

2. 识别 PBD 的 LUKS 加密卷。在以下示例中，块设备是指 */dev/sda2* ：

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. 使用 **clevis luks bind** 命令将卷绑定到 TPM 2.0 设备，例如：

```
# clevis luks bind -d /dev/sda2 tpm2 '{"hash":"sha256","key":"rsa"}'
...
Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

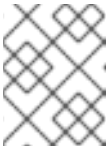
此命令执行四个步骤：

a. 使用与 LUKS 主密钥相同的无序状态测量法创建新的密钥。

b. 使用 Clevis 加密新密钥.

c. 将 Clevis JWE 对象存储在 LUKS2 标头令牌中，或者使用 LUKSMeta（如果使用非默认的 LUKS1 标头）。

d. 启用与 LUKS 一起使用的新密钥。



注意

绑定过程假定至少有一个可用的 LUKS 密码插槽。**clevis luks bind** 命令占用了其中一个插槽。

或者，如果您要将数据封装为特定的平台配置寄存器(PCR)状态，请在 **clevis luks bind** 命令中添加 **pcr_bank** 和 **pcr_ids** 值，例如：

```
# clevis luks bind -d /dev/sda2 tpm2
'{"hash":"sha256","key":"rsa","pcr_bank":"sha256","pcr_ids":"0,1"}'
```

警告

由于只有 PCR 哈希值与密封时使用的策略匹配，并且可以重写哈希时，数据才会被解封，因此添加一个强大的密码短语，以便您可以在 PCR 中的值变化时手动解锁加密的卷。

如果在升级 **shim-x64** 软件包后系统无法自动解锁加密的卷，请遵照 重启后，Clevis TPM2 不再解密 LUKS 设备 KCS 文章中的步骤进行操作。

4. 现在可以使用您的现有密码和 Clevis 策略来解锁卷。
5. 要启用早期引导系统来处理磁盘绑定，请在已安装的系统上使用 **dracut** 工具：

```
# dnf install clevis-dracut
# dracut -fv --regenerate-all
```

验证

1. 要验证 Clevis JWE 对象是否已成功放入 LUKS 标头中，请使用 **clevis luks list** 命令：

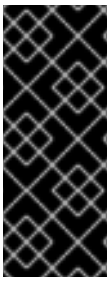
```
# clevis luks list -d /dev/sda2
1: tpm2 '{"hash":"sha256","key":"rsa"}'
```

其他资源

- **clevis-luks-bind(1)**、**clevis-encrypt-tpm2(1)** 和 **dracut.cmdline(7)** 手册页

11.9. 手动从 LUKS 加密卷中删除 CLEVIS PIN

使用以下步骤手动删除 **clevis luks bind** 命令创建的元数据，以及擦除包含 Clevis 添加的密码短语的密钥插槽。



重要

从 LUKS 加密卷中删除 Clevis pin 的建议方法是通过 **clevis luks unbind** 命令。使用 **clevis luks unbind** 的删除过程只包含一个步骤，适用于 LUKS1 和 LUKS2 卷。以下示例命令删除绑定步骤创建的元数据，并擦除 ***/dev/sda2*** 设备上的密钥槽 ***1***：

```
# clevis luks unbind -d /dev/sda2 -s 1
```

先决条件

- 具有 Clevis 绑定的 LUKS 加密卷。

步骤

1. 检查卷（如 ***/dev/sda2***）是由哪个 LUKS 版本加密的，并识别绑定到 Clevis 的槽和令牌：

```
# cryptsetup luksDump /dev/sda2
LUKS header information
Version:        2
...
Keyslots:
  0: luks2
...
1: luks2
      Key:        512 bits
      Priority:   normal
      Cipher:     aes-xts-plain64
...
      Tokens:
        0: clevis
              Keyslot:  1
...
```

在上例中，Clevis 令牌标识为 ***0***，关联的密钥槽是 ***1***。

2. 如果是 LUKS2 加密，请删除令牌：

```
# cryptsetup token remove --token-id 0 /dev/sda2
```

3. 如果您的设备由 LUKS1 加密，由 **Version 表示：1** string 在 **cryptsetup luksDump** 命令的输出中，使用 **luksmeta** flush 命令执行这个额外步骤：

```
# luksmeta wipe -d /dev/sda2 -s 1
```

4. 擦除包含 Clevis 密码短语的密钥插槽：

```
# cryptsetup luksKillSlot /dev/sda2 1
```

其他资源

- **clevis-luks-unbind(1)**、**cryptsetup(8)** 和 **luksmeta(8)** 手册页

11.10. 使用 KICKSTART 配置 LUKS 加密的卷的自动注册

按照此流程中的步骤配置使用 Clevis 注册 LUKS 加密卷的自动安装过程。

步骤

1. 指示 Kickstart 对磁盘进行分区，以便使用临时密码为所有挂载点（除 **/boot** ）启用了 LUKS 加密。注册过程的这一步中的密码是临时密码。

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --grow --encrypted --passphrase=temppass
```

请注意，兼容 OSPP 的系统需要更复杂的配置，例如：

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --size=2048 --encrypted --passphrase=temppass
part /var --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
```

```
part /tmp --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /home --fstype="xfs" --ondisk=vda --size=2048 --grow --encrypted --
passphrase=temppass
part /var/log --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /var/log/audit --fstype="xfs" --ondisk=vda --size=1024 --encrypted --
passphrase=temppass
```

2. 通过在 **%packages** 部分中列出它们来安装相关的 Clevis 软件包：

```
%packages
clevis-dracut
clevis-luks
clevis-systemd
%end
```

3. （可选）要确保您可以在需要时手动解锁加密的卷，请在删除临时密码短语前添加更强的密码短语。如需更多信息，请参阅 如何给现有 LUKS 设备添加密语、密钥或 keyfile 的文章。

4. 在 **%post** 部分中调用 **clevis luks bind** 来执行绑定。之后，删除临时密码：

```
%post
clevis luks bind -y -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv"}' <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

如果您的配置依赖于在早期引导过程中需要网络的 Tang pin，或者使用带有静态 IP 配置的 NBDE 客户端，那么您必须修改**dracut** 命令，如 配置 LUKS 加密卷的手动注册 中所述。

请注意，RHEL 8.3 提供了 **clevis luks bind** 命令的 **-y** 选项。在 RHEL 8.2 及更旧版本中，在 **clevis luks bind** 命令中将 **-y** 替换为 **-f**，并从 Tang 服务器下载公告：

```
%post
curl -sfg http://tang.srv/adv -o adv.jws
clevis luks bind -f -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv","adv":"adv.jws"}' <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

警告

cryptsetup luksRemoveKey 命令可以防止对应用该命令的 LUKS2 设备进行任何进一步的管理。您只能对 LUKS1 设备使用 **dmsetup** 命令恢复删除的主密钥。

在使用 TPM 2.0 策略而不是 Tang 服务器时，您可以使用类似的流程。

其他资源

- **clevis(1)**、**clevis-luks-bind(1)**、**cryptsetup(8)** 和 **dmsetup(8)** 手册页
- 使用 Kickstart 安装 Red Hat Enterprise Linux 9

11.11. 配置 LUKS 加密的可移动存储设备的自动解锁

使用这个流程来设置 LUKS 加密的 USB 存储设备的自动解锁过程。

步骤

1. 要自动解锁 LUKS 加密的可移动存储设备，如 USB 驱动器，请安装 **clevis-udisks2** 软件包：

```
# dnf install clevis-udisks2
```

2. 重启系统，然后使用 **clevis luks bind** 命令执行绑定步骤，如 配置 LUKS 加密卷的手动注册 中所述，例如：

```
# clevis luks bind -d /dev/sdb1 tang '{"url":"http://tang.srv"}'
```

3. 现在，可以在 GNOME 桌面会话中自动解锁 LUKS 加密的可移动设备。绑定到 Clevis 策略的设备也可以通过 **clevis luks unlock** 命令解锁：

```
# clevis luks unlock -d /dev/sdb1
```

在使用 TPM 2.0 策略而不是 Tang 服务器时，您可以使用类似的流程。

其他资源

- **clevis-luks-unlockers(7)** 手册页

11.12. 部署高可用性 NBDE 系统

Tang 提供两种构建高可用性部署的方法：

客户端冗余（推荐）

客户端应配置成能够绑定到多个 Tang 服务器。在此设置中，每个 Tang 服务器都有自己的密钥，客户端可以通过联系这些服务器的子集来进行解密。Clevis 已通过其 **sss** 插件支持此工作流。红帽建议对高可用性部署使用这个方法。

密钥共享

出于冗余的目的，可以部署多个 Tang 实例。要设置第二个或后续的实例，请安装 **tang** 软件包，并使用 **rsync**，通过 **SSH** 将密钥目录复制到新主机上。请注意，红帽不推荐此方法，因为共享密钥会增加密钥的风险，需要额外的自动化基础设施。

使用 Shamir 的 Secret 共享的高可用性 NBDE

Shamir 的 Secret 共享(SSS)是一种加密方案，可将Secret 分成多个独特的部分。要重建 secret，需要几个部分。数字称为阈值，SSS 也被称为阈值方案。

Clevis 提供 SSS 的实施。它创建一个密钥，并将其分为若干个片。每片都使用另一个 pin 进行加密，甚至包括递归 SSS。另外，您可以定义阈值 **t**。如果 NBDE 部署至少解密了 **t** 片，那么它将恢复加密密钥，并且解密过程会成功。当 Clevis 检测到比阈值中指定的部分少时，它会打印错误消息。

示例 1：带两个 Tang 服务器的冗余
当两个 Tang 服务器中至少有一个可用时，以下命令会解密 LUKS 加密设备：

```
# clevis luks bind -d /dev/sda1 sss '{"t":1,"pins":{"tang":[{"url":"http://tang1.srv"},
{"url":"http://tang2.srv"}]}}'
```

上一命令使用以下配置方案：

```
{
   "t":1,
   "pins":{
      "tang":[
         {
            "url":"http://tang1.srv"
         },
         {
            "url":"http://tang2.srv"
         }
      ]
   }
}
```

在此配置中，SSS 阈值 **t** 设置为 **1**，如果列出的两个 **tang** 服务器中至少有一台可用，则 **clevis luks bind** 命令可以成功重建 secret。

示例 2：Tang 服务器和 TPM 设备的共享 secret
当 **tang** 服务器和 **tpm2** 设备都可用时，以下命令可成功解密 LUKS 加密设备：

```
# clevis luks bind -d /dev/sda1 sss '{"t":2,"pins":{"tang":[{"url":"http://tang1.srv"}], "tpm2":
{"pcr_ids":"0,7"}}}'
```

现在 SSS 阀值 't' 设置为 '2' 的配置方案是：

```
{
   "t":2,
   "pins":{
      "tang":[
         {
            "url":"http://tang1.srv"
         }
      ],
      "tpm2":{
         "pcr_ids":"0,7"
      }
   }
}
```

其他资源

- **Tang(8)**（**高可用性** 部分）、**clevis(1)**（**Shamir 的 Secret 共享** 部分）和 **clevis-encrypt-sss(1)** 手册页

11.13. NBDE 网络中虚拟机的部署

clevis luks bind 命令不会改变 LUKS 主密钥。这意味着，如果您创建了一个在虚拟机或云环境中使用的 LUKS 加密镜像，则所有运行此镜像的实例都会共享一个主密钥。这极其不安全，应始终避免。

这不是 Clevis 的一个限制，而是 LUKS 的设计原则。如果您的场景需要在云中有加密的根卷，请也对云中 Red Hat Enterprise Linux 的每个实例执行安装过程（通常使用 Kickstart）。如果没有共享 LUKS 主密钥，就无法共享镜像。

要在虚拟环境中部署自动解锁，请将诸如 **lorax** 或 **virt-install** 的系统与 Kickstart 文件一起使用（请参阅使用 Kickstart 配置 LUKS 加密卷的自动注册）或其它自动配置工具来确保每个加密的虚拟机都有一个唯一的主密钥。

其他资源

- **clevis-luks-bind(1)** 手册页

11.14. 使用 NBDE 为云环境构建可自动注册的虚拟机镜像

在云环境中部署可自动注册的加密镜像会带来一系列独特的挑战。与其他虚拟化环境一样，建议减少从一个镜像启动的实例数量，以避免共享 LUKS 主密钥。

因此，最佳实践是创建自定义映像，这些映像不在任何公共存储库中共享，为部署有限数量的实例提供了基础。要创建的实例的确切数量应当由部署的安全策略定义，并且基于与 LUKS 主密钥攻击向量相关联的风险容忍度。

要构建启用 LUKS 的自动化部署，应当使用 Lorax 或 virt-install 等系统以及一个 Kickstart 文件，来确保镜像构建过程中主密钥的唯一性。

云环境支持我们在这里考虑的两种 Tang 服务器部署选项。首先，Tang 服务器可以在云环境本身中部署。其次，Tang 服务器可以部署在云外的独立的基础架构上，并且这两个基础架构之间有 VPN 连接。

在云中原生部署 Tang 可以轻松部署。但是，考虑到它与其他系统的密文数据持久性层共享基础设施，因此 Tang 服务器的私钥和 Clevis 元数据可以存储在同一个物理磁盘上。对这个物理磁盘的访问允许密文数据的完全泄露。

重要

因此，红帽强烈建议在存储数据的位置和运行 Tang 的系统之间保持物理隔离。在云和 Tang 服务器之间的这种隔离可确保 Tang 服务器的私钥不会被意外与 Clevis 元数据组合。如果云基础设施面临风险，它还提供了对 Tang 服务器的本地控制。

11.15. 将 TANG 部署为容器

tang 容器镜像为在 OpenShift Container Platform(OCP)集群中或独立的虚拟机中运行的 Clevis 客户端提供 Tang-server 解密功能。

先决条件

- **podman** 软件包及其依赖项已安装在系统上。
- 你可以使用 **podman login registry.redhat.io** 命令登录到 **registry.redhat.io** 容器目录。如需更多信息，请参阅 红帽容器注册表身份验证。
- Clevis 客户端安装在包含 LUKS 加密卷的系统上，您希望使用 Tang 服务器自动解锁这些卷。

步骤

1. 从 **registry.redhat.io** 注册中心中拉取 **tang** 容器镜像：

```
# podman pull registry.redhat.io/rhel9/tang
```

2. 运行容器，指定其端口，并指定到 Tang 密钥的路径。前面的示例运行 **tang** 容器，指定端口 *7500*，并指示到 **/var/db/tang** 目录的 Tang 密钥的路径：

```
# podman run -d -p 7500:7500 -v tang-keys:/var/db/tang --name tang
registry.redhat.io/rhel9/tang
```

请注意，Tang 默认使用端口 80，但这可能与其他服务冲突，如 Apache HTTP 服务器。

3. [可选] 为提高安全性，定期轮转 Tang 密钥。您可以使用 **tangd-rotate-keys** 脚本，例如：

```
# podman run --rm -v tang-keys:/var/db/tang registry.redhat.io/rhel9/tang tangd-rotate-keys -
v -d /var/db/tang
Rotated key 'rZAMKAseaXBe0rcKXL1hCCIq-DY.jwk' -> .'rZAMKAseaXBe0rcKXL1hCCIq-
DY.jwk'
Rotated key 'x1AIpc6WmnCU-CabD8_4q18vDuw.jwk' -> .'x1AIpc6WmnCU-
CabD8_4q18vDuw.jwk'
Created new key GrMMX_WfdqomIU_4RyjpcdlXb0E.jwk
Created new key _dTTfn17sZZqVAp80u3ygFDHtjk.jwk
Keys rotated successfully.
```

验证

- 在包含 LUKS 加密卷的系统上，通过 Tang 服务器自动解锁，检查 Clevis 客户端是否可以使用 Tang 加密和解密纯文本消息：

```
# echo test | clevis encrypt tang '{"url":"http://localhost:7500"}' | clevis decrypt
The advertisement contains the following signing keys:

x1AIpc6WmnCU-CabD8_4q18vDuw

Do you wish to trust these keys? [ynYN] y
test
```

在 *localhost* URL 上的 Tang 服务器可用并通过端口 *7500* 进行通信时，上一示例命令在其输出的末尾显示 **test** 字符串。

其他资源

- **podman(1)**、**clevis(1)** 和 **tang(8)**手册页

11.16. NBDE_CLIENT 和 NBDE_SERVER RHEL 系统角色(CLEVIS 和 TANG)简介

RHEL 系统角色是 Ansible 角色和模块的一个集合，为远程管理多个 RHEL 系统提供一致的配置接口。

您可以使用 Ansible 角色使用 Clevis 和 Tang 自动部署基于策略的解密(PBD)解决方案。**rhel-system-roles** 包中包含了这些系统角色、相关的例子以及参考文档。

nbde_client 系统角色使您能够以自动化的方式部署多个Clevis客户端。请注意，**nbde_client** 角色只支持 Tang 绑定，您目前无法将其用于 TPM2 绑定。

nbde_client 角色需要已经使用 LUKS 加密的卷。此角色支持将 LUKS 加密卷绑定到一个或多个网络绑定 (NBDE)服务器 - Tang 服务器。您可以使用密码短语保留现有的卷加密，或者将其删除。删除密码短语后，您只能使用 NBDE 解锁卷。当卷最初是使用在置备系统后会删除的临时密钥或密码进行加密时，这非常有用，

如果您同时提供密语和密钥文件，角色将使用您首先提供的那一个。如果找不到任何有效密语或密码，它将尝试从现有的绑定中检索密码短语。

PBD 将绑定定义为设备到插槽的映射。这意味着对同一个设备你可以有多个绑定。默认插槽是插槽 1。

nbde_client 角色也提供了 **state** 变量。使用 **present** 值来创建新绑定或更新现有绑定。与 **clevis luks bind** 命令不同，您可以使用 **state: present** 来覆盖其设备插槽中的现有绑定。**absent** 的值会删除指定的绑定。

使用 **nbde_client** 系统角色，您可以部署和管理 Tang 服务器，来作为自动磁盘加密解决方案的一部分。此角色支持以下功能：

- 轮转 Tang 密钥
- 部署和备份 Tang 密钥

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.nbde_server/README.md** file
- **/usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md** 文件
- **/usr/share/doc/rhel-system-roles/nbde_server/** directory
- **/usr/share/doc/rhel-system-roles/nbde_client/** directory

11.17. 使用 NBDE_SERVER RHEL 系统角色设置多个 TANG 服务器

按照以下步骤准备和应用包含您的 Tang 服务器设置的 Ansible playbook。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_server
  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true
```

此 playbook 示例确保部署 Tang 服务器和密钥轮转。

当 **nbde_server_manage_firewall** 和 **nbde_server_manage_selinux** 都被设置为 **true** 时，**nbde_server** 角色使用 **firewall** 和 **selinux** 角色来管理 **nbde_server** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 要在安装了 Clevis 的系统上使用 **grubby** 工具来确保 Tang pin 的网络可用，请输入：

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.nbde_server/README.md** file
- **/usr/share/doc/rhel-system-roles/nbde_server/** directory

11.18. 使用 NBDE_CLIENT RHEL 系统角色设置多个 CLEVIS 客户端

使用 **nbde_client** RHEL 系统角色，您可以在多个系统上准备并应用包含 Clevis 客户端设置的 Ansible playbook。

注意

nbde_client 系统角色只支持 Tang 绑定。因此，您无法将其用于 TPM2 绑定。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  vars:
    nbde_client_bindings:
```

```
    - device: /dev/rhel/root
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com
    - device: /dev/rhel/swap
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com
```

这个示例 playbook 配置 Clevis 客户端，以便在两个 Tang 服务器中至少有一个可用时自动解锁两个 LUKS 加密的卷

nbde_client 系统角色只支持使用动态主机配置协议(DHCP)的场景。要将 NBDE 用于具有静态 IP 配置的客户端，请使用以下 playbook：

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
      - device: /dev/rhel/swap
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
  tasks:
    - name: Configure a client with a static IP address during early boot
      ansible.builtin.command:
        cmd: grubby --update-kernel=ALL --args='GRUB_CMDLINE_LINUX_DEFAULT="ip={{
<ansible_default_ipv4.address> }}::{{ <ansible_default_ipv4.gateway> }}:{{
<ansible_default_ipv4.netmask> }}::{{ <ansible_default_ipv4.alias> }}:none"'
```

在这个 playbook 中，将 ***<ansible_default_ipv4.*>*** 字符串替换为您网络的 IP 地址，例如：**ip={{ 192.0.2.10 }}::{{ 192.0.2.1 }}:{{ 255.255.255.0 }}::{{ ens3 }}:none**。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md** 文件

- **/usr/share/doc/rhel-system-roles/nbde_client/** directory
- 在初始 ramdisk (initrd)中查找 Linux 网络配置 文章

第 12 章 审计系统

审计不会为您的系统提供额外的安全，而是用于发现系统上使用的安全策略的违规。可以通过其他安全措施(如 SELinux)进一步防止这些违规。

12.1. LINUX 审计

Linux 审计系统提供了一种跟踪与您的系统相关的安全信息的方法。根据预配置的规则，审计会生成日志条目，来尽可能多地记录系统上所发生的事件的相关信息。对于关键任务环境而言至关重要，可用来确定安全策略的违反者及其所执行的操作。

以下列表总结了审计可以在其日志文件中记录的一些信息：

- 事件的日期、时间、类型和结果
- 主题和对象的敏感度标签
- 事件与触发事件的用户的身份的关联
- 所有对审计配置的修改，并尝试访问审计日志文件
- 所有身份验证机制的使用，如 SSH 和 Kerberos 等
- 对任何可信数据库的更改，如 **/etc/passwd**
- 尝试向或从系统导入或导出信息
- 根据用户身份、主题和对象标签以及其他属性包含或排除事件

审计系统的使用也是许多安全相关认证的一项要求。审计旨在满足或超出以下认证或合规指南的要求：

- 受控访问保护配置文件(CAPP)
- 标记的安全保护配置文件(LSPP)
- 规则集基本访问控制(RSBAC)
- 国家工业安全计划操作手册(NISPOM)
- 联邦信息安全管理法案(FISMA)
- 支付卡行业 - 数据安全标准(PCI-DSS)
- 安全技术实施指南(STIG)

审计还包括：

- 由国家信息保障合作伙伴(NIAP)和最佳安全行业(BSI)进行了评估.
- 在 Red Hat Enterprise Linux 5 上认证为 LSPP/CAPP/RSBAC/EAL4+
- 在 Red Hat Enterprise Linux 6 上认证为操作系统保护配置文件/评估保证级别 4+(OSPP/EAL4+)

使用案例

监视文件访问

审计可以跟踪文件或目录是否已被访问、修改、执行或文件属性是否已被改变。例如，这有助于检测对重要文件的访问，并在其中一个文件损坏时提供审计跟踪。

监控系统调用

可将审计配置为在每次使用特定系统调用时生成日志条目。例如，这可用于通过监控 **settimeofday**、**clock_adjtime** 和其他与时间相关的系统调用来跟踪对系统时间的修改。

记录用户运行的命令

审计可以跟踪文件是否已被执行，因此可以定义一个规则以记录每次特定命令的执行。例如，可以对 **/bin** 目录中的每个可执行文件定义一个规则。然后，可以按用户 ID 搜索生成的日志条目，以生成每个用户所执行的命令的审计跟踪。

记录系统路径名称的执行

除了观察在规则调用时将路径转换为 inode 的文件访问之外，审计现在还可以观察路径的执行，即使路径在规则调用中不存在，或者在规则调用后文件被替换了。这允许规则在升级程序可执行文件后或甚至在其安装之前继续运行。

记录安全事件

pam_faillock 认证模块能够记录失败的登录尝试。也可以将审计设置为记录失败的登录尝试，并提供试图登录的用户的额外信息。

搜索事件

审计提供了 **ausearch** 工具，可用于过滤日志条目，并根据多个条件提供完整的审计跟踪。

运行总结报告

aureport 实用程序可用于生成记录事件的日常报告等。然后，系统管理员可以分析这些报告，并进一步调查可疑的活动。

监控网络访问

nftables、**iptables** 和 **ebtables** 工具可以配置为触发审计事件，使系统管理员能够监控网络访问。

注意

系统性能可能会受到影响，具体取决于审计所收集的信息量。

12.2. 审计系统架构

审计系统由两个主要部分组成：用户空间应用程序和工具，以及内核端系统调用处理。内核组件接收用户空间应用程序的系统调用，并通过以下过滤器对其进行过滤：**user**、**task**、**fstype** 或 **exit**。

系统调用传递 **exclude** 过滤器后，它将通过上述中的一个过滤器发送，该过滤器根据审计规则配置，将其发送到审计守护进程以进行进一步处理。

用户空间审计守护进程从内核收集信息，并在日志文件中创建条目。其他审计用户空间工具与审计守护进程、内核审计组件或审计日志文件进行交互：

- **auditctl** Audit 控制工具与内核审计组件进行交互，来管理规则并控制事件生成进程的许多设置和参数。
- 其余的审计工具将审计日志文件的内容作为输入，并根据用户的要求生成输出。例如，**aureport** 工具生成所有记录的事件的报告。

在 RHEL 9 中，Audit 分配程序守护进程(**audisp**)功能集成到 Audit 守护进程中(**auditd**)。用于实时分析程序与审计事件交互的插件配置文件默认位于 **/etc/audit/plugins.d/** 目录中。

12.3. 为安全环境配置 AUDITD

默认的 **auditd** 配置应该适合于大多数环境。但是，如果您的环境必须满足严格的安全策略，您可以在 **/etc/audit/auditd.conf** 文件中更改审计守护进程配置的以下设置：

log_file

包含审计日志文件的目录（通常为 **/var/log/audit/**）应位于单独的挂载点上。这可以防止其他进程消耗此目录的空间，并为审计守护进程提供准确的剩余空间检测。

max_log_file

指定单个审计日志文件的最大大小，必须设置为充分利用保存审计日志文件的分区上的可用空间。**max_log_file'** 参数指定最大文件大小（以 MB 为单位）。给出的值必须是数字。

max_log_file_action

一旦达到 **max_log_file** 中设置的限制，决定要采取什么行动，应将其设置为 **keep_logs**，以防止审计日志文件被覆盖。

space_left

指定磁盘上剩余的可用空间量，其是**space_left_action**参数中设置的触发时所采取的操作。必须设置一个数字，让管理员有足够的时间来响应，并释放磁盘空间。**space_left** 的值取决于审计日志文件的生成速度。如果 space_left 的值被指定为整数，它将被解释为绝对大小(MiB)。如果值被指定为 1 到 99 之间的数字，后跟一个百分比符号（例如 5%），则审计守护进程会根据包含 **log_file** 的文件系统的大小来计算绝对大小（以 MB 为单位）。

space_left_action

建议将 **space_left_action** 参数设置为 **email** 或 使用适当通知方法的 **exec**。

admin_space_left

指定绝对最小可用空间量，其是 **admin_space_left_action** 参数中设置的触发时所采取的操作，必须设置一个值，为记录管理员所执行的操作保留足够的空间。此参数的数字值应小于 space_left 的数。您还可以在数字后面附加一个百分比符号（例如 1%），以便审计守护进程根据磁盘分区计算数值。

admin_space_left_action

应设置为 **single** 来将系统置于单用户模式，并允许管理员释放一些磁盘空间。

disk_full_action

指定当保存审计日志文件的分区上没有可用空间时触发的操作，必须设置为 **halt** 或 **single**。当审计无法记录事件时，这可确保系统关闭或以单用户模式运行。

disk_error_action

指定当在包含审计日志文件的分区上检测到错误时触发的操作，必须设置为 **syslog**、**single** 或**halt**，具体取决于您处理硬件故障的本地安全策略。

flush

应设置为 **incremental_async**。它与 **freq** 参数相结合，该参数决定了在强制与硬盘进行硬盘同步前可以将多少条记录发送到磁盘。**freq** 参数应设置为**100**。这些参数可确保审计事件数据与磁盘上的日志文件同步，同时保持良好的活动性能。

其余配置选项应根据您的本地安全策略来设置。

12.4. 启动和控制 AUDITD

配置了 **auditd** 后，启动服务以收集 审计信息，并将它存储在日志文件中。以 root 用户身份运行以下命令来启动 **auditd** ：

```
# service auditd start
```

将 **auditd** 配置为在引导时启动：

```
# systemctl enable auditd
```

您可以使用 **# auditctl -e 0** 命令临时禁用 **auditd**，并使用 **# auditctl -e 1** 重新启用它。

您可以使用 **service auditd *<action>*** 命令对 **auditd** 执行其他操作，其中 *<action>* 可以是以下之一：

stop

停止 **auditd**。

restart

重新启动 **auditd**。

reload 或 force-reload

重新加载 **/etc/audit/auditd.conf** 文件中 **auditd** 的配置。

rotate

轮转 **/var/log/audit/** 目录中的日志文件。

resume

在其之前被暂停后重新恢复审计事件记录，例如，当保存审计日志文件的磁盘分区中没有足够的可用空间时。

condrestart 或 try-restart

只有当 **auditd** 运行时才重新启动它。

status

显示 **auditd** 的运行状态。

注意

service命令是与 **auditd** 守护进程正确交互的唯一方法。您需要使用 **service** 命令，以便正确记录 **auid** 值。您只将 **systemctl** 命令用于两个操作： **enable** 和 **status**。

12.5. 了解审计日志文件

默认情况下，审计系统将日志条目存储在 **/var/log/audit/audit.log** 文件中；如果启用了日志轮转，则轮转的 **audit.log** 文件也在存储同一个目录中。

添加以下审计规则，来记录读取或修改 **/etc/ssh/sshd_config** 文件的每次尝试：

```
# auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

如果 **auditd** 守护进程正在运行，使用以下命令在审计日志文件中创建新事件，例如：

```
$ cat /etc/ssh/sshd_config
```

audit.log 文件中的该事件如下。

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no exit=-13
a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538 auid=1000 uid=1000
gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1
comm="cat" exe="/bin/cat" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="sshd_config"
type=CWD msg=audit(1364481363.243:24287):  cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0 name="/etc/ssh/sshd_config" inode=409248
```

```
dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0
type=PROCTITLE msg=audit(1364481363.243:24287) :
proctitle=636174002F6574632F7373682F737368645F636F6E666967
```

以上事件由四个记录组成，它们共享相同的时间戳和序列号。记录始终以 **type=** 关键字开头。每个记录由多个 ***name=value*** 对组成，它们之间由空格或逗号分开。对上述事件的详细分析如下：

第一条记录

type=SYSCALL

type 字段包含记录的类型。在本例中，**SYSCALL** 值指定此记录是由对内核的系统调用触发的。

msg=audit(1364481363.243:24287):

msg 字段记录：

- **audit (*time_stamp*:*ID*)** 格式的记录的时间戳和唯一ID。如果多个记录是作为同一审计事件的一部分而产生的，则它们共享相同的时间戳和 ID。时间戳使用 Unix 时间格式 - 自 1970 年 1 月 1 日 00:00:00 UTC 以来的秒数。
- 由内核或用户空间应用程序提供的各种特定于事件的 ***name=value***对。

arch=c000003e

arch 字段包含系统的 CPU 架构信息。该值 **c000003e** 以十六进制表示法编码。当使用 **ausearch** 命令搜索审计记录时，请使用 **-i** 或 **--interpret** 选项来自动将十六进制值转换成人类可读的等效值。**c000003e** 值被解释为 **x86_64**。

syscall=2

syscall字段记录了发送到内核的系统调用的类型。值 **2** 可以与 **/usr/include/asm/unistd_64.h** 文件中人类可读的等效值匹配。在本例中，**2** 是 **打开** 系统调用。请注意，**ausyscall** 工具允许您将系统调用号转换为人类可读的等效值。使用 **ausyscall --dump** 命令显示所有系统调用及其编号的列表。如需更多信息，请参阅 **ausyscall**(8)手册页。

success=no

success 字段记录了该特定事件中记录的系统调用是成功还是失败。在这种情况下，调用不成功。

exit=-13

exit 字段包含一个值，指定系统调用返回的退出码。此值因不同的系统调用而不同。您可以使用以下命令将值解释成人类可读的等效值：

```
# ausearch --interpret --exit -13
```

请注意，上例假定您的审计日志包含一个失败的事件，其退出码为 **-13** 。

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

a0至**a3**字段记录了该事件中系统调用的前四个参数，用十六进制符号编码。这些参数取决于使用的系统调用，可以通过 **ausearch** 工具来解释它们。

items=1

items 字段包含系统调用记录后面的 PATH 辅助记录的数量。

ppid=2686

ppid 字段记录了父进程ID（PPID）。在这种情况下，**2686** 是父进程（如 **bash**）的 PPID 。

pid=3538

pid 字段记录了进程 ID（PID）。在本例中，**3538** 是 **cat** 进程的 PID。

auid=1000

auid字段记录了审计用户 ID，即loginuid。此 ID 在登录时分配给用户，并被每个进程继承，即使用户的身份改变了，例如使用 **su - john** 命令切换用户帐户。

uid=1000

uid 字段记录了启动分析过程的用户的用户 ID。使用以下命令可以将用户 ID 解释成用户名： **ausearch -i --uid *UID***。

gid=1000

gid 字段记录了启动分析过程的用户的组 ID。

euid=1000

euid 字段记录了启动分析过程的用户的有效用户 ID。

suid=1000

suid 字段记录了启动分析过程的用户的设置用户 ID。

fsuid=1000

fsuid 字段记录了启动分析进程的用户的文件系统用户 ID。

egid=1000

egid 字段记录了启动分析过程的用户的有效组 ID。

sgid=1000

sgid 字段记录了启动分析过程的用户的组 ID。

fsgid=1000

fsgid 字段记录了启动分析进程的用户的文件系统组 ID。

tty=pts0

tty 字段记录了分析过程被调用的终端。

ses=1

ses 字段记录了分析过程被调用的会话的会话 ID。

comm="cat"

comm 字段记录了用于调用分析过程的命令行名称。在本例中，cat **命令用于触发此审计事件**。

exe="/bin/cat"

exe 字段记录了用于调用分析过程的可执行文件的路径。

subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

subj 字段记录了被分析的进程在执行时被标记的 SELinux 上下文。

key="sshd_config"

key 记录了与在审计日志中生成该事件的规则相关联的管理员定义的字符串。

第二条记录

type=CWD

在第二条记录中，**type** 字段值为 **CWD** - 当前工作目录。此类型用于记录从中调用第一条记录中指定的系统调用的进程的工作目录。
此记录的目的是记录当前进程的位置，以防在相关 PATH 记录中捕获到相对路径。这样，就可以重建绝对路径。

msg=audit(1364481363.243:24287)

msg 字段持有与第一条记录中的值相同的时间戳和 ID 值。时间戳使用 Unix 时间格式 - 自 1970 年 1 月 1 日 00:00:00 UTC 以来的秒数。

cwd="/home/*user_name*"

cwd 字段包含系统调用所在目录的路径。

第三条记录

type=PATH

在第三条记录中，**type** 字段值为 **PATH**。审计事件包含作为参数传递给系统调用的每个路径的 **PATH** 类型记录。在这个审计事件中，只有一个路径(**/etc/ssh/sshd_config**) 被用作参数。

msg=audit(1364481363.243:24287):

msg 字段拥有与第一和第二条记录中的值相同的时间戳和 ID 值。

item=0

item 字段表示在 **SYSCALL** 类型记录所引用的项目总数中，当前记录是哪个项目。这个数是以零为基础的；值为 **0** 表示它是第一项。

name="/etc/ssh/sshd_config"

name 字段记录了作为参数传递给系统调用的文件或目录的路径。在本例中，它是 **/etc/ssh/sshd_config** 文件。

inode=409248

inode 字段包含与该事件中记录的文件或目录相关联的 inode 号。以下命令显示与 **409248** inode 号相关联的文件或目录：

```
# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

dev 字段指定了包含该事件中记录的文件或目录的设备的次要和主要 ID。在本例中，值表示 **/dev/fd/0** 设备。

mode=0100600

mode 字段记录文件或目录权限，由数字标记。它是 **st_mode** 字段中的 **stat** 命令返回。如需更多信息，请参阅 **stat(2)** 手册页。在这种情况下，**0100600** 可以解释为 **-rw-------**，这意味着只有 root 用户对 **/etc/ssh/sshd_config** 文件具有读和写的权限。

ouid=0

ouid 字段记录了对象所有者的用户 ID。

ogid=0

ogid 字段记录了对象所有者的组 ID。

rdev=00:00

rdev 字段包含一个记录的设备标识符，仅用于特殊文件。在这种情况下，不会使用它，因为记录的文件是一个常规文件。

obj=system_u:object_r:etc_t:s0

obj 字段记录了 SELinux 上下文，在执行时，记录的文件或目录被贴上了标签。

nametype=NORMAL

nametype 字段记录了每个路径记录在给定系统调用的上下文中的操作意图。

cap_fp=none

cap_fp 字段记录了与设置文件或目录对象的基于文件系统的允许能力有关的数据。

cap_fi=none

cap_fi 字段记录了与文件或目录对象的基于继承文件系统的能力设置有关的数据。

cap_fe=0

cap_fe 字段记录了文件或目录对象基于文件系统能力的有效位的设置。

cap_fver=0

cap_fver 字段记录了文件或目录对象基于文件系统能力的版本。

第四条记录

type=PROCTITLE

type 字段包含记录的类型。在本例中，**PROCTITLE** 值指定此记录提供触发此审计事件的完整命令行，该事件是由对内核的系统调用触发的。

proctitle=636174002F6574632F7373682F737368645F636F6E666967

proctitle 字段记录了用于调用分析过程的命令的完整命令行。该字段采用十六进制表示法编码，不允许用户影响审计日志解析器。对触发此审计事件的命令进行文本解码。当使用 **ausearch** 命令搜索审计记录时，请使用 **-i** 或 **--interpret** 选项来自动将十六进制值转换成人类可读的等效值。**636174002F6574632F7373682F737368645F636F6E666967** 值解释为 **cat /etc/ssh/sshd_config**。

12.6. 使用 AUDITCTL 来定义和执行审计规则

审计系统根据一组规则进行操作，这些规则定义日志文件中所捕获的内容。使用 **auditctl** 工具，可以在命令行或 **/etc/audit/rules.d/** 目录中设置审计规则。

auditctl 命令使您能够控制审计系统的基本功能，并定义决定记录哪些审计事件的规则。

文件系统规则示例

1. 要定义一条规则，记录对 **/etc/passwd** 文件的所有写访问和每个属性的修改：

```
# auditctl -w /etc/passwd -p wa -k passwd_changes
```

2. 要定义一条规则，记录对 **/etc/selinux/** 目录中所有文件的写访问和每个属性的修改：

```
# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

系统调用规则示例

1. 要定义一条规则，当程序每次使用 **adjtimex** 或 **settimeofday** 系统调用时就创建一条日志，系统使用 64 位构架：

```
# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

2. 定义一条规则，在 ID 为 1000 或以上的系统用户每次删除或重命名文件时创建一条日志：

```
# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000 -F auid!=4294967295 -k delete
```

请注意，**-F auid!=4294967295** 选项用于排除未设置登录 UID 的用户。

可执行文件规则

要定义一条规则，记录所有 **/bin/id** 程序的执行，请执行以下命令：

-

```
# auditctl -a always,exit -F exe=/bin/id -F arch=b64 -S execve -k execution_bin_id
```

其他资源

- **auditctl(8)** 手册页。

12.7. 定义持久性审计规则

要定义在重启过程中保持不变的审计规则，必须直接将其包含在 **/etc/audit/rules.d/audit.rules** 文件中，或者使用 **augenrules** 程序读取位于**/etc/audit/rules.d/** 目录中的规则。

请注意，每次 **auditd** 服务启动时都会生成 **/etc/audit/audit.rules** 文件。**/etc/audit/rules.d/** 中的文件使用相同的 **auditctl** 命令行语法来指定规则。哈希符号(#)后面的空行和文本将被忽略。

另外，您可以使用 **auditctl** 命令来从用 **-R** 选项指定的文件中读取规则，例如：

```
# auditctl -R /usr/share/audit/sample-rules/30-stig.rules
```

12.8. 符合标准的预配置的审计规则文件

要配置符合特定认证标准（如 OSPP、PCI DSS 或 STIG）的 Audit，您可以使用 **audit** 软件包安装的一组预配置的规则文件作为起点。示例规则位于 **/usr/share/audit/sample-rules** 目录中。

警告

sample-rules 目录中的审计示例规则既不是详尽的，也不是最新的，因为安全标准是动态的，并处于变化之中。提供这些规则只是用于演示审计规则是如何构建和编写的。它们不能确保立即符合最新的安全标准。要根据特定安全准则使您的系统符合最新的安全标准，请使用 基于 SCAP 的安全合规性工具。

30-nispom.rules
: 满足国家工业安全计划操作手册"信息系统安全"一章中指定的要求的审计规则配置.

30-ospp-v42*.rules
: 满足 OSPP（通用目的操作系统保护配置文件）配置文件版本 4.2 中定义的要求的审计规则配置。

30-pci-dss-v31.rules
: 满足支付卡行业数据安全标准(PCI DSS)v3.1 要求的审计规则配置。

30-stig.rules
: 满足安全技术实施指南(STIG)要求的审计规则配置。

要使用这些配置文件，将其复制到 **/etc/audit/rules.d/** 目录中，并使用 **augenrules --load** 命令，例如：

```
# cd /usr/share/audit/sample-rules/
# cp 10-base-config.rules 30-stig.rules 31-privileged.rules 99-finalize.rules /etc/audit/rules.d/
# augenrules --load
```

您可以使用编号方案对审核规则进行排序。如需更多信息，请参阅 **/usr/share/audit/sample-rules/README-rules** 文件。

其他资源

- **audit.rules(7)** 手册页。

12.9. 使用 AUGENRULES 来定义持久性规则

augenrules脚本读取位于**/etc/audit/rules.d/**目录下的规则，并将它们编译成**audit.rures**文件。这个脚本会根据文件的自然排列顺序，按特定顺序处理以 **.rules** 结尾的所有文件。这个目录中的文件被组织到具有如下含义的组中：

10
: 内核和 auditctl 配置

20
: 匹配常规规则的规则，但您想要一个不同的匹配

30
: 主规则

40
: 可选规则

50
: 特定于服务器的规则

70
: 系统本地规则

90
: 完成(不可变)

规则并非是一次全部使用。它们是策略的一部分，应仔细考虑，并将单个文件复制到 **/etc/audit/rules.d/**。例如，要在 STIG 配置中设置系统，请复制规则 **10-base-config**、**30-stig**、**31-privileged** 和 **99-finalize**。

在 **/etc/audit/rules.d/** 目录中有了规则之后，运行带有 **--load** 参数的 **augenrules** 脚本来加载它们：

```
# augenrules --load
/sbin/augenrules: No change
No rules
enabled 1
failure 1
pid 742
rate_limit 0
...
```

其他资源

- **audit.rules(8)** 和 **augenrules(8)** 手册页。

12.10. 禁用 AUGENRULES

使用以下步骤来禁用 **augenrules** 工具。这会将审计切换为使用 **/etc/audit/audit.rules** 文件中定义的规则。

流程

1. 将 **/usr/lib/systemd/system/auditd.service** 文件复制到 **/etc/systemd/system/** 目录中：

```
# cp -f /usr/lib/systemd/system/auditd.service /etc/systemd/system/
```

2. 在您选择的文本编辑器中编辑 **/etc/systemd/system/auditd.service** 文件，例如：

```
# vi /etc/systemd/system/auditd.service
```

3. 注释掉包含 **augenrules** 的行，将包含 **auditctl -R** 命令的行取消注释：

```
#ExecStartPost=-/sbin/augenrules --load
ExecStartPost=-/sbin/auditctl -R /etc/audit/audit.rules
```

4. 重新载入 **systemd** 守护进程以获取 **auditd.service** 文件中的修改：

```
# systemctl daemon-reload
```

5. 重启 **auditd** 服务：

```
# service auditd restart
```

其他资源

- **augenrules(8)** 和 **audit.rules(8)** 手册页。
- auditd 服务重启将覆盖对 /etc/audit/audit.rules 所做的修改。

12.11. 设置审计来监控软件更新

您可以使用预先配置的规则 **44-installers.rules** 将 Audit 配置为监控以下安装软件的工具：

- **dnf** [3]
- **yum**
- **pip**
- **npm**
- **cpan**
- **gem**
- **luarocks**

要监控 **rpm** 实用程序，请安装 **rpm-plugin-audit** 软件包。然后，审计会在安装或升级软件包时生成 **SOFTWARE_UPDATE** 事件。您可以通过在命令行中输入 **ausearch -m SOFTWARE_UPDATE** 来列出这些事件。

注意

预配置的规则文件不能用于 **ppc64le** 和 **aarch64** 架构的系统。

先决条件

- **auditd** 是根据 为安全环境配置 auditd 中提供的设置进行配置的。

流程

1. 将预先配置的规则文件 **44-installers.rules** 从 **/usr/share/audit/sample-rules/** 目录复制到 **/etc/audit/rules.d/** 目录中：

```
# cp /usr/share/audit/sample-rules/44-installers.rules /etc/audit/rules.d/
```

2. 加载审计规则：

```
# augenrules --load
```

验证

1. 列出载入的规则：

```
# auditctl -l
-p x-w /usr/bin/dnf-3 -k software-installer
-p x-w /usr/bin/yum -k software-installer
-p x-w /usr/bin/pip -k software-installer
-p x-w /usr/bin/npm -k software-installer
-p x-w /usr/bin/cpan -k software-installer
-p x-w /usr/bin/gem -k software-installer
-p x-w /usr/bin/luarocks -k software-installer
```

2. 执行安装，例如：

```
# dnf reinstall -y vim-enhanced
```

3. 在审计日志中搜索最近的安装事件，例如：

```
# ausearch -ts recent -k software-installer
----
time->Thu Dec 16 10:33:46 2021
type=PROCTITLE msg=audit(1639668826.074:298):
proctitle=2F7573722F6C6962657865632F706C6174666F726D2D707974686F6E002F75737
22F62696E2F646E66007265696E7374616C6C002D790076696D2D656E68616E636564
type=PATH msg=audit(1639668826.074:298): item=2 name="/lib64/ld-linux-x86-64.so.2"
inode=10092 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=1 name="/usr/libexec/platform-python"
inode=4618433 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=0 name="/usr/bin/dnf" inode=6886099
dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:rpm_exec_t:s0
```

```
nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1639668826.074:298): cwd="/root"
type=EXECVE msg=audit(1639668826.074:298): argc=5 a0="/usr/libexec/platform-python"
a1="/usr/bin/dnf" a2="reinstall" a3="-y" a4="vim-enhanced"
type=SYSCALL msg=audit(1639668826.074:298): arch=c000003e syscall=59 success=yes
exit=0 a0=55c437f22b20 a1=55c437f2c9d0 a2=55c437f2aeb0 a3=8 items=3 ppid=5256
pid=5375 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=3
comm="dnf" exe="/usr/libexec/platform-python3.6"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="software-installer"
```

12.12. 使用审计监控用户登录时间

要监控特定时间哪个用户登录了，您不需要以任何特殊的方式配置审计。您可以使用 **ausearch** 或 **aureport** 工具，它们提供不同的方法来展示相同的信息。

先决条件

- **auditd** 是根据 为安全环境配置 auditd 中提供的设置进行配置的。

流程

要显示用户登录的时间，请使用以下命令之一：

- 在审计日志中搜索 **USER_LOGIN** 消息类型：

```
# ausearch -m USER_LOGIN -ts '12/02/2020' '18:00:00' -sv no
time->Mon Nov 22 07:33:22 2021
type=USER_LOGIN msg=audit(1637584402.416:92): pid=1939 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login acct="
(unknown)" exe="/usr/sbin/sshd" hostname=? addr=10.37.128.108 terminal=ssh res=failed'
```

  - 您可以使用 **-ts** 选项指定日期和时间。如果不使用这个选项，**ausearch** 将提供从当天开始的结果，如果您省略时间，**ausearch** 将提供从午夜开始的结果。
  - 您可以使用 **-sv yes** 选项来过滤成功的登录尝试，**-sv no** 用来过滤失败的登录尝试。

- 将 **ausearch** 命令的原始输出传送给 **aulast** 工具，它以类似于 **last** 命令的输出格式显示输出。例如：

```
# ausearch --raw | aulast --stdin
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.22.16.106     Mon Nov 22 07:40 - 07:40  (00:00)
reboot   system boot  4.18.0-348.6.el8 Mon Nov 22 07:33
```

- 使用 **aureport** 命令及 **--login -i** 选项来显示登录事件列表。

```
# aureport --login -i

Login Report
============================================
# date time auid host term exe success event
============================================
1. 11/16/2021 13:11:30 root 10.40.192.190 ssh /usr/sbin/sshd yes 6920
2. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6925
```

```
3. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6930
4. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6935
5. 11/16/2021 13:11:33 root 10.40.192.190 ssh /usr/sbin/sshd yes 6940
6. 11/16/2021 13:11:33 root 10.40.192.190 /dev/pts/0 /usr/sbin/sshd yes 6945
```

其他资源

- **ausearch(8)** 手册页。
- **aulast(8)** 手册页。
- **aureport(8)** 手册页。

12.13. 其他资源

- RHEL审计系统参考知识库文章。
- Auditd execution options in a container 知识库文章。
- Linux审计文档项目页面。
- **audit** 软件包在 **/usr/share/doc/audit/** 目录中提供文档。
- **auditd(8)**、**auditctl(8)**、**ausearch(8)**、**audit.rules(7)**、**audispd.conf(5)**、**audispd(8)**、**auditd.conf(5)**、**ausearch-expression(5)**、**aulast(8)**、**aulastlog(8)**、**aureport(8)**、**ausyscall(8)**、**autrace(8)** 和 **auvirt(8)** 手册页。

[3] 由于**dnf** 在 RHEL 中是符号链接，因此**dnf** 审计规则中的路径必须包含符号链接的目标。要接收正确的审计事件，请通过将 **path=/usr/bin/dnf** 路径改为 **/usr/bin/dnf-3** 来修改 **44-installers.rules** 文件。

第 13 章 使用 FAPOLICYD 阻止和允许应用程序

根据规则集设置和强制实施允许或拒绝应用程序执行的策略，可有效防止执行未知的和具有潜在恶意的软件。

13.1. FAPOLICYD 简介

fapolicyd 软件框架根据用户定义的策略来控制应用程序的执行。这是防止在系统上运行不受信任的和可能具有恶意的应用程序的最有效的方法之一。

fapolicyd 框架提供以下组件。

- **fapolicyd** 服务
- **fapolicyd** 命令行工具
- **fapolicyd** RPM 插件
- **fapolicyd** 规则语言
- **fagenrules** 脚本

管理员可以为任何应用程序定义 **allow** 和 **deny** 执行规则，并根据路径、哈希、MIME 类型或信任进行审计。

fapolicyd 框架引入了信任的概念。在被系统软件包管理器正确安装后，应用程序是可信的，因此它会在系统 RPM 数据库中注册。**fapolicyd** 守护进程使用 RPM 数据库作为受信任的二进制文件和脚本的列表。**fapolicyd** RPM 插件注册任何由 DNF 软件包管理器或 RPM 软件包管理器处理的系统更新。插件会通知 **fapolicyd** 守护进程有关此数据库中的更改。添加应用程序的其他方法需要创建自定义规则，并重新启动 **fapolicyd** 服务。

fapolicyd 服务配置位于 **/etc/fapolicyd/** 目录中，结构如下。

- **/etc/fapolicyd/fapolicyd.trust** 文件包含一个可信文件的列表。您还可以在 **/etc/fapolicyd/trust.d/** 目录中使用多个信任文件。
- 包含 **allow** 和 **deny** 执行规则的文件的 **/etc/fapolicyd/rules.d/** 目录。**fagenrules** 脚本将这些组件规则文件合并到 **/etc/fapolicyd/compiled.rules** 文件中。
- **fapolicyd.conf** 文件包含守护进程的配置选项。此文件主要用于性能调优目的。

/etc/fapolicyd/rules.d/ 中的规则被组织在几个文件中，每个文件自代表不同的策略目标。对应文件名开头的数字决定了 **/etc/fapolicyd/compiled.rules** 中的顺序：

10
: 语言规则.

20
: 与 dracut 相关的规则。

21
: 更新者的规则。

30
: 模式。

40

ELF 规则。

41

共享对象规则。

42

可信的 ELF 规则。

70

可信语言规则。

72

Shell 规则.

90

拒绝执行规则。

95

允许打开的规则。

您可以使用以下方法之一进行 **fapolicyd** 完整性检查：

- 文件大小检查
- SHA-256 哈希的比较
- 完整性映射架构（IMA）子系统

默认情况下，**fapolicyd** 不进行完整性检查。根据文件大小进行完整性检查很快，但攻击者可以替换文件的内容并保留其字节大小。计算和检查 SHA-256 校验和更安全，但这会影响系统性能。**fapolicyd.conf** 中的 **integrity = ima** 选项需要在包含可执行文件的所有文件系统上都支持扩展属性（也称为 ***xattr***）。

其他资源

- **fapolicyd(8)** 、**fapolicyd.rules(5)**、**fapolicyd.conf(5)**、**fapolicyd.trust(13)**、**fagenrules(8)** 和 **fapolicyd-cli(1)** 手册页。
- 在 管理、监控和更新内核 文档中，使用内核完整性子系统加强安全性 一章。
- 文档与 **fapolicyd** 软件包一起安装在 **/usr/share/doc/fapolicyd/** 目录和 **/usr/share/fapolicyd/sample-rules/README-rules** 文件中。

13.2. 部署 FAPOLICYD

在 RHEL 中部署 **fapolicyd** 框架：

流程

1. 安装 **fapolicyd** 软件包：

```
# dnf install fapolicyd
```

2. 启用并启动 **fapolicyd** 服务：

```
# systemctl enable --now fapolicyd
```

验证

验证

1. 验证 **fapolicyd** 服务是否正常运行：

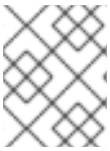
```
# systemctl status fapolicyd
● fapolicyd.service - File Access Policy Daemon
   Loaded: loaded (/usr/lib/systemd/system/fapolicyd.service; enabled; vendor p>
   Active: active (running) since Tue 2019-10-15 18:02:35 CEST; 55s ago
  Process: 8818 ExecStart=/usr/sbin/fapolicyd (code=exited, status=0/SUCCESS)
 Main PID: 8819 (fapolicyd)
    Tasks: 4 (limit: 11500)
   Memory: 78.2M
   CGroup: /system.slice/fapolicyd.service
           └─8819 /usr/sbin/fapolicyd

Oct 15 18:02:35 localhost.localdomain systemd[1]: Starting File Access Policy D>
Oct 15 18:02:35 localhost.localdomain fapolicyd[8819]: Initialization of the da>
Oct 15 18:02:35 localhost.localdomain fapolicyd[8819]: Reading RPMDB into memory
Oct 15 18:02:35 localhost.localdomain systemd[1]: Started File Access Policy Da>
Oct 15 18:02:36 localhost.localdomain fapolicyd[8819]: Creating database
```

2. 以没有 root 权限的用户身份登录，检查 **fapolicyd** 是否正常工作，例如：

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

13.3. 使用其它信任源将文件标记为可信

fapolicyd 框架信任 RPM 数据库中包含的文件。您可以通过在 **/etc/fapolicyd/fapolicyd.trust** 纯文本文件或 **/etc/fapolicyd/trust.d/** 目录下添加相应的条目来将额外文件标记为信任的文件，这支持将信任的文件列表分割成多个文件。您可以直接使用文本编辑器或通过 **fapolicyd-cli** 命令来修改 **fapolicyd.trust** 或 **/etc/fapolicyd/trust.d** 中的文件。



注意

由于性能的原因，使用 **fapolicyd.trust** 或 **trust.d/** 将文件标记为信任的比编写自定义的 **fapolicyd** 规则要好些。

先决条件

- **fapolicyd** 框架部署在您的系统上。

流程

1. 将自定义二进制文件复制到所需的目录中，例如：

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

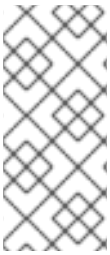
2. 将自定义二进制文件标记为信任的，并将相应的条目添加到 **/etc/fapolicyd/trust.d/** 中的 **myapp** 文件中：

```
# fapolicyd-cli --file add /tmp/ls --trust-file myapp
```

- 如果您跳过 **--trust-file** 选项，则之前的命令会将相应的行添加到 **/etc/fapolicyd/fapolicyd.trust**。
- 要将目录中所有现有的文件标记为信任的，请将目录路径提供为 **--file** 选项的参数，例如：**fapolicyd-cli --file add /tmp/my_bin_dir/ --trust-file myapp**。

3. 更新 **fapolicyd** 数据库：

```
# fapolicyd-cli --update
```



注意

更改信任的文件或目录的内容会改变其校验和，因此 **fapolicyd** 不再将它们视为信任的。

要使新内容再次被信任，请使用 **fapolicyd-cli --file update** 命令刷新文件信任数据库。如果没有提供任何参数，则整个数据库都会刷新。或者，您可以指定特定文件或目录的路径。然后，使用 **fapolicyd-cli --update** 更新数据库。

验证

1. 检查您的自定义二进制文件现在是否可以执行，例如：

```
$ /tmp/ls
ls
```

其他资源

- **fapolicyd.trust(13)** 手册页。

13.4. 为 FAPOLICYD 添加自定义 ALLOW 和 DENY 规则

fapolicyd 包中的默认规则集不影响系统功能。对于自定义场景，如将二进制文件和脚本存储在非标准目录中，或者在没有 **dnf** 或 **rpm** 安装程序的情况下添加应用程序，您必须将额外文件标记为可信或添加新的自定义规则。

对于基本场景，首选 使用额外的信任源来将文件标记为信任的文件 。在更高级的场景中，如仅允许为特定用户和组标识符来执行一个自定义二进制文件，请将新的自定义规则添加到 **/etc/fapolicyd/rules.d/** 目录中。

以下步骤演示了如何添加新的规则以允许自定义二进制文件。

先决条件

- **fapolicyd** 框架部署在您的系统上。

流程

1. 将自定义二进制文件复制到所需的目录中，例如：

```
$ cp /bin/ls /tmp
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

2. 停止 **fapolicyd** 服务：

```
# systemctl stop fapolicyd
```

3. 使用 debug 模式来识别相应的规则。因为 **fapolicyd --debug** 命令的输出很冗长，所以您只能按 **Ctrl**+**C** 或终止相应的进程来停止它，并将错误输出重定向到文件中。在这种情况下，您可以使用 **--debug-deny** 选项而不是 **--debug** 来限制输出只访问拒绝：

```
# fapolicyd --debug-deny 2> fapolicy.output &
[1] 51341
```

或者，您可以在另一个终端中运行 **fapolicyd** debug 模式。

4. 重复 **fapolicyd** 拒绝的命令：

```
$ /tmp/ls
bash: /tmp/ls: Operation not permitted
```

5. 通过在前台恢复并按 **Ctrl**+**C** 来停止 debug 模式：

```
# fg
fapolicyd --debug 2> fapolicy.output
^C
...
```

或者，杀掉 **fapolicyd** debug 模式的进程：

```
# kill 51341
```

6. 查找拒绝执行应用程序的规则：

```
# cat fapolicy.output | grep 'deny_audit'
...
rule=13 dec=deny_audit perm=execute auid=0 pid=6855 exe=/usr/bin/bash : path=/tmp/ls
ftype=application/x-executable trust=0
```

7. 找到包含阻止自定义二进制文件执行的规则的文件。在这种情况下，**deny_audit perm=execute** 规则属于 **90-deny-execute.rules** 文件：

```
# ls /etc/fapolicyd/rules.d/
10-languages.rules  40-bad-elf.rules    72-shell.rules
20-dracut.rules     41-shared-obj.rules    90-deny-execute.rules
21-updaters.rules   42-trusted-elf.rules   95-allow-open.rules
30-patterns.rules   70-trusted-lang.rules


# cat /etc/fapolicyd/rules.d/90-deny-execute.rules
```

```
# Deny execution for anything untrusted

deny_audit perm=execute all : all
```

8. 将一个新的 **allow** 规则添加到规则文件 *之前* 的文件中，该文件包含了拒绝执行 **/etc/fapolicyd/rules.d/** 目录中自定义二进制文件的规则：

```
# touch /etc/fapolicyd/rules.d/80-myapps.rules
# vi /etc/fapolicyd/rules.d/80-myapps.rules
```

将以下规则插入到 **80-myapps.rules** 文件中：

```
allow perm=execute exe=/usr/bin/bash trust=1 : path=/tmp/ls ftype=application/x-executable
trust=0
```

另外，您可以通过将以下规则添加到 **/etc/fapolicyd/rules.d/** 中的规则文件中，来允许执行 **/tmp** 目录中所有的二进制文件：

```
allow perm=execute exe=/usr/bin/bash trust=1 : dir=/tmp/ trust=0
```

重要

要使规则对指定目录下的所有目录递归有效，请将斜杠添加到规则中 **dir=** 参数值的末尾（上例中的 **/tmp/**）。

9. 要防止自定义二进制文件内容的更改，请使用 SHA-256 校验和定义所需的规则：

```
$ sha256sum /tmp/ls
780b75c90b2d41ea41679fcb358c892b1251b68d1927c80fbc0d9d148b25e836  ls
```

将规则改为以下定义：

```
allow perm=execute exe=/usr/bin/bash trust=1 :
sha256hash=780b75c90b2d41ea41679fcb358c892b1251b68d1927c80fbc0d9d148b25e836
```

10. 检查编译的列表是否与 **/etc/fapolicyd/rules.d/** 中设置的规则不同，并更新列表，该列表存储在 **/etc/fapolicyd/compiled.rules** 文件中：

```
# fagenrules --check
/usr/sbin/fagenrules: Rules have changed and should be updated
# fagenrules --load
```

11. 检查您的自定义规则是否在阻止执行的规则之前的 **fapolicyd** 规则列表中：

```
# fapolicyd-cli --list
...
13. allow perm=execute exe=/usr/bin/bash trust=1 : path=/tmp/ls ftype=application/x-
executable trust=0
14. deny_audit perm=execute all : all
...
```

12. 启动 **fapolicyd** 服务：

```
# systemctl start fapolicyd
```

-

验证

1. 检查您的自定义二进制文件现在是否可以执行，例如：

```
$ /tmp/ls
ls
```

其他资源

- **fapolicyd.rules(5)** 和 **fapolicyd-cli(1)** 手册页。
- 文档与 **fapolicyd** 软件包一起安装在 **/usr/share/fapolicyd/sample-rules/README-rules** 文件中。

13.5. 启用 FAPOLICYD 完整性检查

默认情况下，**fapolicyd** 不执行完整性检查。您可以配置 **fapolicyd**，来通过比较文件大小或 SHA-256 哈希执行完整性检查。您还可以使用完整性度量架构(IMA)子系统来设置完整性检查。

先决条件

- **fapolicyd** 框架部署在您的系统上。

流程

1. 在您选择的文本编辑器中打开 **/etc/fapolicyd/fapolicyd.conf** 文件，例如：

```
# vi /etc/fapolicyd/fapolicyd.conf
```

2. 将 **integrity** 选项的值从 **none** 改为 **sha256**，保存文件并退出编辑器：

```
integrity = sha256
```

3. 重启 **fapolicyd** 服务：

```
# systemctl restart fapolicyd
```

验证

1. 备份用于验证的文件：

```
# cp /bin/more /bin/more.bak
```

2. 更改 **/bin/more** 二进制文件的内容：

```
# cat /bin/less > /bin/more
```

3. 以普通用户身份使用更改的二进制文件：

```
# su example.user
$ /bin/more /etc/redhat-release
bash: /bin/more: Operation not permitted
```

4. 恢复更改：

```
# mv -f /bin/more.bak /bin/more
```

13.6. 故障排除与 FAPOLICYD 相关的问题

下面的部分提供了 **fapolicyd** 应用程序框架的基本故障排除技巧，以及使用 **rpm** 命令添加应用程序的指导。

使用 rpm安装应用程序

- 如果使用 **rpm** 命令安装应用程序，则必须手动执行 **fapolicyd** RPM 数据库的刷新：

  1. 安装 *应用程序* ：

     ```
     # rpm -i application.rpm
     ```

  2. 刷新数据库：

     ```
     # fapolicyd-cli --update
     ```

     如果您跳过这一步，系统可能会被冻结，必须重启。

服务状态

- 如果 **fapolicyd** 无法正常工作，请检查服务状态：

  ```
  # systemctl status fapolicyd
  ```

fapolicyd-cli 检查和列表

- **--check-config**、**--check-watch_fs** 和 **--check-trustdb** 选项可帮助您查找语法错误、尚未监视的文件系统和文件不匹配，例如：

  ```
  # fapolicyd-cli --check-config
  Daemon config is OK

  # fapolicyd-cli --check-trustdb
  /etc/selinux/targeted/contexts/files/file_contexts miscompares: size sha256
  /etc/selinux/targeted/policy/policy.31 miscompares: size sha256
  ```

- 使用 **--list** 选项检查规则的当前列表及其顺序：

  ```
  # fapolicyd-cli --list
  ...
  9. allow perm=execute all : trust=1
  10. allow perm=open all : ftype=%languages trust=1
  11. deny_audit perm=any all : ftype=%languages
  ```

```
12. allow perm=any all : ftype=text/x-shellscript
13. deny_audit perm=execute all : all
...
```

Debug 模式

- Debug 模式提供关于匹配规则、数据库状态等的详细信息。将 **fapolicyd** 切换到 debug 模式：

  1. 停止 **fapolicyd** 服务：

     ```
     # systemctl stop fapolicyd
     ```

  2. 使用 debug 模式来识别相应的规则：

     ```
     # fapolicyd --debug
     ```

     因为 **fapolicyd --debug** 命令的输出非常详细，所以您可以将错误输出重定向到文件中：

     ```
     # fapolicyd --debug 2> fapolicy.output
     ```

     另外，若要在 **fapolicyd** 拒绝访问时仅将输出限制为条目，请使用 **--debug-deny** 选项：

     ```
     # fapolicyd --debug-deny
     ```

删除 fapolicyd 数据库

- 要解决与 **fapolicyd** 数据库相关的问题，请尝试删除数据库文件：

```
# systemctl stop fapolicyd
# fapolicyd-cli --delete-db
```

警告

不要删除 **/var/lib/fapolicyd/** 目录。**fapolicyd** 框架只自动恢复这个目录下的数据库文件。

转储 fapolicyd 数据库

- **fapolicyd** 包含了所有启用的信任源的条目。您可以在转储数据库后检查条目：

```
# fapolicyd-cli --dump-db
```

应用程序管道

- 在个别情况下，删除 **fapolicyd** 管道文件可以解决锁定问题：

```
# rm -f /var/run/fapolicyd/fapolicyd.fifo
```

其他资源

- **fapolicyd-cli(1)** 手册页。

13.7. 使用 FAPOLICYD RHEL 系统角色配置防止未知代码的执行

您可以通过运行 Ansible playbook，使用 **fapolicyd** 系统角色来防止未知代码的执行。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- name: Preventing execution of unknown code
  hosts: all
  vars:
    fapolicyd_setup_integrity: sha256
    fapolicyd_setup_trust: rpmdb,file
    fapolicyd_add_trusted_file:
      - </usr/bin/my-ls>
      - </opt/third-party/app1>
      - </opt/third-party/app2>
  roles:
    - rhel-system-roles.fapolicyd
```

   您可以使用 **linux-system-roles.fapolicyd** RHEL 系统角色的以下变量来进一步自定义保护：

   fapolicyd_setup_integrity
   您可以设置以下一种完整性类型：**none**、**sha256** 和 **size**。

   fapolicyd_setup_trust
   您可以设置信任文件类型 **file**、**rpmd** 和 **deb**。

   fapolicyd_add_trusted_file
   您可以列出信任且 **fapolicyd** 不会阻止其执行的可执行文件。

2. 验证 playbook 语法：

```
# ansible-playbook ~/playbook.yml --syntax-check
```

   请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
# ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.fapolicyd/README.md** 文件

13.8. 其他资源

- 使用 **man -k fapolicyd** 命令列出与 **fapolicyd** 相关的手册页。
- FOSDEM 2020 fapolicyd 演示。

第 14 章 保护系统免受入侵 USB 设备的攻击

USB 设备可能会加载间谍软件、恶意软件或特洛伊木马，这可能会窃取你的数据或破坏你的系统。作为 Red Hat Enterprise Linux 管理员，您可以使用**USBGuard** 来防止此类 USB 攻击。

14.1. USBGUARD

借助 USBGuard 软件框架，您可以根据内核中的 USB 设备授权功能，使用允许和禁止设备的基本列表来防止系统免受入侵 USB 设备的攻击。

USBGuard 框架提供以下组件：

- 带有用于动态交互和策略实施的进程间通信(IPC)接口的系统服务组件
- 与正在运行的 **usbguard** 系统服务交互的命令行接口
- 编写 USB 设备授权策略的规则语言
- 用于与共享库中实施的系统服务组件交互的 C++ API

usbguard系统服务配置文件（**/etc/usbguard/usbguard-daemon.conf**）包括授权用户和组使用 IPC 接口的选项。

重要

系统服务提供 USBGuard 公共 IPC 接口。在 Red Hat Enterprise Linux 中，对此接口的访问默认只限于 root 用户。

考虑设置 **IPCAccessControlFiles** 选项（推荐）或 **IPCAllowedUsers** 和 **IPCAllowedGroups** 选项，来限制对 IPC 接口的访问。

确保您没有未配置 Access Control List(ACL)，因为这会将 IPC 接口公开给所有本地用户，允许他们操作 USB 设备的授权状态，并修改 USBGuard 策略。

14.2. 安装 USBGUARD

使用这个流程安装并启动 USBGuard 框架。

流程

1. 安装 **usbguard** 软件包：

```
# dnf install usbguard
```

2. 创建初始规则集：

```
# usbguard generate-policy > /etc/usbguard/rules.conf
```

3. 启动 **usbguard** 守护进程，并确保它会在引导时自动启动：

```
# systemctl enable --now usbguard
```

验证

1. 验证 **usbguard** 服务是否正在运行：

```
# systemctl status usbguard
● usbguard.service - USBGuard daemon
   Loaded: loaded (/usr/lib/systemd/system/usbguard.service; enabled; vendor preset:
disabled)
   Active: active (running) since Thu 2019-11-07 09:44:07 CET; 3min 16s ago
     Docs: man:usbguard-daemon(8)
 Main PID: 6122 (usbguard-daemon)
    Tasks: 3 (limit: 11493)
   Memory: 1.2M
   CGroup: /system.slice/usbguard.service
           └─6122 /usr/sbin/usbguard-daemon -f -s -c /etc/usbguard/usbguard-daemon.conf

Nov 07 09:44:06 localhost.localdomain systemd[1]: Starting USBGuard daemon...
Nov 07 09:44:07 localhost.localdomain systemd[1]: Started USBGuard daemon.
```

2. 列出 USBGuard 识别的 USB 设备：

```
# usbguard list-devices
4: allow id 1d6b:0002 serial "0000:02:00.0" name "xHCI Host Controller" hash...
```

其他资源

- **usbguard(1)** 和 **usbguard-daemon.conf(5)** 手册页。

14.3. 使用 CLI 阻止和授权 USB 设备

您可以通过在终端中使用 **usbguard** 命令将 USBGuard 设置为授权和阻止 USB 设备。

先决条件

- **usbguard** 服务已安装并运行。

流程

1. 列出 USBGuard 识别的 USB 设备，例如：

```
# usbguard list-devices
1: allow id 1d6b:0002 serial "0000:00:06.7" name "EHCI Host Controller" hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" parent-hash
"4PHGcaDKWtPjKDwYpIRG722cB9SlGz9l9lea93+Gt9c=" via-port "usb1" with-interface
09:00:00
...
6: block id 1b1c:1ab1 serial "000024937962" name "Voyager" hash
"CrXgiaWIf2bZAU+5WkzOE7y0rdSO82XMzubn7HDb95Q=" parent-hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" via-port "1-3" with-interface
08:06:50
```

2. 授权设备 *<6>* 来与系统进行交互：

```
# usbguard allow-device <6>
```

3. 取消授权并删除设备 *<6>*:

```
# usbguard reject-device <6>
```

4. 取消授权并保留设备 *<6>* ：

```
# usbguard block-device <6>
```

注意

USBGuard 使用术语 ***block*** 和 ***reject***，具有以下含义：

block
现在不要与此设备进行交互。

reject
忽略这个设备，就像它不存在一样。

其他资源

- **usbguard (1)** 手册页
- **usbguard --help** 命令

14.4. 永久阻止和授权 USB 设备

您可以使用 **-p** 选项永久阻止和授权 USB 设备。这会在当前策略中添加一条特定于设备的规则。

先决条件

- **usbguard** 服务已安装并运行。

流程

1. 配置 SELinux，以允许 **usbguard** 守护进程编写规则。

   a. 显示与 **usbguard** 相关的 **semanage** 布尔值。

```
# semanage boolean -l | grep usbguard
usbguard_daemon_write_conf     (off  ,  off)  Allow usbguard to daemon write conf
usbguard_daemon_write_rules    (on   ,   on)  Allow usbguard to daemon write rules
```

   b. 可选：如果 **usbguard_daemon_write_rules** 布尔值已关闭，请打开它。

```
# semanage boolean -m --on usbguard_daemon_write_rules
```

2. 列出 USBGuard 识别的 USB 设备：

```
# usbguard list-devices
1: allow id 1d6b:0002 serial "0000:00:06.7" name "EHCI Host Controller" hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" parent-hash
"4PHGcaDKWtPjKDwYpIRG722cB9SlGz9l9lea93+Gt9c=" via-port "usb1" with-interface
09:00:00
```

```
...
6: block id 1b1c:1ab1 serial "000024937962" name "Voyager" hash
"CrXgiaWIf2bZAU+5WkzOE7y0rdSO82XMzubn7HDb95Q=" parent-hash
"JDOb0BiktYs2ct3mSQKopnOOV2h9MGYADwhT+oUtF2s=" via-port "1-3" with-interface
08:06:50
```

3. 永久授权设备 ***6***，以与系统进行交互：

```
# usbguard allow-device 6 -p
```

4. 永久取消授权并删除设备 ***6*** ：

```
# usbguard reject-device 6 -p
```

5. 永久取消授权并保留设备 ***6*** ：

```
# usbguard block-device 6 -p
```

注意

USBGuard 使用术语 ***block*** 和 ***reject***，具有以下含义：

block
现在不要与此设备进行交互。

reject
忽略这个设备，就像它不存在一样。

验证

1. 检查 USBGuard 规则是否包含您所做的更改。

```
# usbguard list-rules
```

其他资源

- **usbguard(1)** 手册页。
- 使用 **usbguard --help** 命令列出内置的帮助信息。

14.5. 为 USB 设备创建自定义策略

以下流程包含了为 USB 设备创建反映您场景需求的规则集的步骤。

先决条件

- **usbguard** 服务已安装并运行。
- **/etc/usbguard/rules.conf** 文件包含了由**usbguard generate-policy** 命令生成的初始规则集。

流程

1. 创建一个策略，其授权当前连接的 USB 设备，并将生成的规则保存到 **rules.conf** 文件中：

```
# usbguard generate-policy --no-hashes > ./rules.conf
```

--no-hashes 选项不会为设备生成哈希属性。在配置设置中避免哈希属性，因为它们可能不是永久的。

2. 使用您选择的文本编辑器编辑 **rules.conf** 文件，例如：

```
# vi ./rules.conf
```

3. 根据需要添加、删除或编辑规则。例如，以下规则只允许带有一个大容量存储接口的设备与系统进行交互：

```
allow with-interface equals { 08:*:* }
```

有关详细的规则语言描述和更多示例，请参阅 **usbguard-rules.conf(5)**手册页。

4. 安装更新的策略：

```
# install -m 0600 -o root -g root rules.conf /etc/usbguard/rules.conf
```

5. 重启 **usbguard** 守护进程以应用您的更改：

```
# systemctl restart usbguard
```

验证

1. 检查您的自定义规则是否在活动的策略中，例如：

```
# usbguard list-rules
...
4: allow with-interface 08:*:*
...
```

其他资源

- **usbguard-rules.conf(5)** 手册页。

14.6. 为 USB 设备创建结构化自定义策略

您可以在 **/etc/usbguard/rules.d/** 目录中的几个 **.conf** 文件中组织自定义 USBGuard 策略。然后 **usbguard-daemon** 将主 **rules.conf** 文件与目录中的 **.conf** 文件按字母顺序组合在一起。

先决条件

- **usbguard** 服务已安装并运行。

流程

1. 创建一个授权当前连接的 USB 设备的策略，并将生成的规则保存到一个新的 **.conf** 文件，如 ***policy.conf***。

```
# usbguard generate-policy --no-hashes > ./policy.conf
```

--no-hashes 选项不会为设备生成哈希属性。在配置设置中避免哈希属性，因为它们可能不是永久的。

2. 使用您选择的文本编辑器显示 ***policy.conf*** 文件，例如：

```
# vi ./policy.conf
...
allow id 04f2:0833 serial "" name "USB Keyboard" via-port "7-2" with-interface { 03:01:01
03:00:00 } with-connect-type "unknown"
...
```

3. 将所选行移到一个单独的 **.conf** 文件中。

注意

文件名开头的两位数字指定守护进程读取配置文件的顺序。

例如，将键盘的规则复制到一个新的 **.conf** 文件中。

```
# grep "USB Keyboard" ./policy.conf > ./10keyboards.conf
```

4. 将新策略安装到 **/etc/usbguard/rules.d/** 目录中。

```
# install -m 0600 -o root -g root 10keyboards.conf /etc/usbguard/rules.d/10keyboards.conf
```

5. 将其余的行移到主 **rules.conf** 文件中。

```
# grep -v "USB Keyboard" ./policy.conf > ./rules.conf
```

6. 安装其余的规则。

```
# install -m 0600 -o root -g root rules.conf /etc/usbguard/rules.conf
```

7. 重新启动 **usbguard** 守护进程，以应用您的更改。

```
# systemctl restart usbguard
```

验证

1. 显示所有活动的 USBGuard 规则。

```
# usbguard list-rules
...
15: allow id 04f2:0833 serial "" name "USB Keyboard" hash
"kxM/iddRe/WSCocgiuQlVs6Dn0VEza7KiHoDeTz0fyg=" parent-hash
"2i6ZBJfTl5BakXF7Gba84/Cp1gslnNc1DM6vWQpie3s=" via-port "7-2" with-interface {
03:01:01 03:00:00 } with-connect-type "unknown"
...
```

2. 显示 **/etc/usbguard/rules.d/** 目录中的 **rules.conf** 文件以及所有 **.conf** 文件的内容。

```
# cat /etc/usbguard/rules.conf /etc/usbguard/rules.d/*.conf
```

3. 验证活动的规则是否包含文件中的所有规则，并且顺序正确。

其他资源

- **usbguard-rules.conf(5)** 手册页。

14.7. 授权用户和组使用 USBGUARD IPC 接口

使用这个流程授权特定用户或组使用 USBGuard 公共 IPC 接口。默认情况下，只有 root 用户可以使用此接口。

先决条件

- **usbguard** 服务已安装并运行。
- **/etc/usbguard/rules.conf** 文件包含了由**usbguard generate-policy** 命令生成的初始规则集。

流程

1. 使用您选择的文本编辑器编辑 **/etc/usbguard/usbguard-daemon.conf** 文件：

```
# vi /etc/usbguard/usbguard-daemon.conf
```

2. 例如，添加一行规则，允许 **wheel** 组中的所有用户使用 IPC 接口，并保存文件：

```
IPCAllowGroups=wheel
```

3. 您还可以使用 **usbguard** 命令添加用户或组。例如，以下命令可让 *joesec* 用户拥有访问 **Devices** 和 **Exceptions** 部分的所有权限。另外，*joesec* 可以列出并修改当前的策略：

```
# usbguard add-user joesec --devices ALL --policy modify,list --exceptions ALL
```

若要删除对 *joesec* 用户授予的权限，可使用 **usbguard remove-user joesec** 命令。

4. 重启 **usbguard** 守护进程以应用您的更改：

```
# systemctl restart usbguard
```

其他资源

- **usbguard(1)** 和 **usbguard-rules.conf(5)** 手册页。

14.8. 将 USBGUARD 授权事件记录到 LINUX 审计日志中

使用以下步骤将 USBguard 授权事件记录集成到标准的 Linux 审计日志中。默认情况下，**usbguard** 守护进程将事件记录到 **/var/log/usbguard/usbguard-audit.log** 文件中。

先决条件

- **usbguard** 服务已安装并运行。

- **auditd** 服务正在运行。

流程

1. 使用您选择的文本编辑器编辑 **usbguard-daemon.conf** 文件：

```
# vi /etc/usbguard/usbguard-daemon.conf
```

2. 将 **AuditBackend** 选项从 **FileAudit** 改为 **LinuxAudit**：

```
AuditBackend=LinuxAudit
```

3. 重启 **usbguard** 守护进程以应用配置更改：

```
# systemctl restart usbguard
```

验证

1. 查询 USB 授权事件的 **audit** 守护进程日志，例如：

```
# ausearch -ts recent -m USER_DEVICE
```

其他资源

- **usbguard-daemon.conf(5)** 手册页。

14.9. 其他资源

- **usbguard(1)**、**usbguard-rules.conf(5)**、**usbguard-daemon(8)** 和 **usbguard-daemon.conf(5)** 手册页。
- USBGuard 主页 。

第 15 章 配置远程日志记录解决方案

要确保在日志记录服务器中集中记录来自环境中各种机器的日志，您可以将 **Rsyslog** 应用程序配置为将适合客户端系统中特定条件的日志记录到服务器。

15.1. RSYSLOG 日志记录服务

Rsyslog 应用程序与 **systemd-journald** 服务相结合，在 Red Hat Enterprise Linux 中提供了本地和远程记录支持。**rsyslogd** 守护进程持续读取 **systemd-journald** 服务收到的 **syslog** 消息。**rsyslogd** 过滤这些 **syslog** 事件并记录到 **rsyslog** 日志文件，或者根据其配置将它们转发到其他服务。

rsyslogd 守护进程还提供扩展的过滤、加密受保护的转发消息、输入和输出模块，并支持使用 TCP 和 UDP 协议进行传输。

在 **/etc/rsyslog.conf** 中，是 **rsyslog** 的主要配置文件，您可以根据 **rsyslogd** 处理消息来指定规则。通常，您可以通过其来源和主题（设施）和紧急情况（优先级）对消息进行分类，然后分配在消息适合这些条件时应执行的操作。

在 **/etc/rsyslog.conf** 中，您还可以看到 **rsyslogd** 维护的日志文件列表。大多数日志文件位于 **/var/log/** 目录中。**httpd** 和 **samba** 等一些应用将其日志文件存储在 **/var/log/** 中的子目录中。

其他资源

- **rsyslogd (8)** 和 **rsyslog.conf (5)** man page。
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

15.2. 安装 RSYSLOG 文档

Rsyslog 应用程序在 https://www.rsyslog.com/doc/ 上提供了广泛的在线文档，但您也可以在本地安装 **rsyslog-doc** 文档软件包。

先决条件

- 您已在系统中激活了 **AppStream** 软件仓库。
- 您有权使用 **sudo** 安装新软件包。

流程

- 安装 **rsyslog-doc** 软件包：

```
# dnf install rsyslog-doc
```

验证

- 在您选择的浏览器中打开 **/usr/share/doc/rsyslog/html/index.html** 文件，例如：

```
$ firefox /usr/share/doc/rsyslog/html/index.html &
```

15.3. 通过 TCP 配置服务器进行远程记录

Rsyslog 应用程序可让您运行日志服务器并配置各个系统将其日志文件发送到日志记录服务器。要通过 TCP 使用远程日志，请同时配置服务器和客户端。服务器收集和分析由一个或多个客户端系统发送的日志。

使用 Rsyslog 应用程序，您可以维护一个集中的日志系统，该系统可通过网络将日志消息转发到服务器。为了避免服务器不可用时消息丢失，您可以为转发操作配置操作队列。这样，无法发送的消息将存储在本地，直到服务器再次可访问为止。请注意，此类队列无法针对使用 UDP 协议的连接配置。

omfwd 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。由于插件内置在内，因此不必加载它。

默认情况下，**rsyslog** 使用端口 **514** 上的 TCP。

先决条件

- rsyslog 已安装在服务器系统上。
- 您以 **root** 身份登录到服务器中。
- 使用 **semanage** 命令，为可选步骤安装 **policycoreutils-python-utils** 软件包。
- **firewalld** 服务在运行。

流程

1. 可选：要将不同的端口用于 **rsyslog** 流量，在端口中添加 **syslogd_port_t** SELinux 类型。例如，启用端口 **30514** ：

```
# semanage port -a -t syslogd_port_t -p tcp 30514
```

2. 可选：要使用不同的端口用于 **rsyslog** 流量，将 **firewalld** 配置为允许该端口上传入的 **rsyslog** 流量。例如，允许端口 **30514** 上的 TCP 流量：

```
# firewall-cmd --zone=<zone-name> --permanent --add-port=30514/tcp
success
# firewall-cmd --reload
```

3. 在 **/etc/rsyslog.d/** 目录中创建一个新文件（例如，**remotelog.conf**），并插入以下内容：

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }
```

```
# Provides TCP syslog reception
module(load="imtcp")

# Adding this ruleset to process remote messages
ruleset(name="remote1"){
    authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
     *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="30514" ruleset="remote1")
```

4. 将更改保存到 **/etc/rsyslog.d/remotelog.conf** 文件。
5. 测试 **/etc/rsyslog.conf** 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

6. 确保 **rsyslog** 服务在日志记录服务器中运行并启用：

```
# systemctl status rsyslog
```

7. 重新启动 **rsyslog** 服务。

```
# systemctl restart rsyslog
```

8. 可选：如果没有启用 **rsyslog**，请确保 **rsyslog** 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

您的日志服务器现在已配置为从您环境中的其他系统接收和存储日志文件。

其他资源

- **rsyslogd(8)**, **rsyslog.conf(5)**, **semanage(8)**, 和 **firewall-cmd(1)** man pages.
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

15.4. 通过 TCP 配置远程日志记录到服务器

按照以下步骤配置通过 TCP 协议将日志消息转发到服务器。**omfwd** 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。因为插件内置在内，所以不必加载它。

先决条件

- **rsyslog** 软件包安装在应该向服务器报告的客户端系统上。
- 您已为远程日志记录配置了服务器。
- 在 SELinux 中允许指定的端口并在防火墙中打开。

- 系统包含 **policycoreutils-python-utils** 软件包，它为 SELinux 配置中添加非标准端口提供 **semanage** 命令。

流程

1. 在 **/etc/rsyslog.d/** 目录中创建一个名为 的新文件，如 **10-remotelog.conf**，并插入以下内容：

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="30514" protocol="tcp"
    )
```

其中：

- **queue.type="linkedlist"** 设置启用 LinkedList 内存中队列，
- **queue.filename** 设置定义磁盘存储。备份文件是使用 **example_fwd** 前缀，在之前全局 **workDirectory** 指令指定的工作目录中创建的。
- **action.resumeRetryCount -1** 设置防止 **rsyslog** 在服务器没有响应而重试连接时丢弃消息，
- 如果 **rsyslog** 关闭，**queue.saveOnShutdown="on"** 设置会保存内存中的数据。
- 最后一行将所有收到的消息转发到日志记录服务器。端口规格是可选的。
  使用这个配置，**rsyslog** 会向服务器发送消息，但如果远程服务器无法访问，则会将消息保留在内存中。只有 **rsyslog** 耗尽配置的内存队列空间或需要关闭时，才能创建磁盘上的文件，从而让系统性能受益。

注意

rsyslog 按照一般顺序处理配置文件 **/etc/rsyslog.d/**。

2. 重新启动 **rsyslog** 服务。

```
# systemctl restart rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统上，查看 **/var/log/messages** 日志，例如：

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

其中 *hostname* 是客户端系统的主机名。请注意，日志包含输入 **logger** 命令的用户的用户名，本例中为 **root**。

其他资源

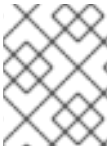
- **rsyslogd(8)** 和 **rsyslog.conf(5)** 手册页。
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

15.5. 配置 TLS 加密的远程日志记录

默认情况下，Rsyslog 以纯文本格式发送 remote-logging 通信。如果您的场景需要保护这个通信频道，您可以使用 TLS 加密它。

要通过 TLS 使用加密传输，请同时配置服务器和客户端。服务器收集和分析由一个或多个客户端系统发送的日志。

您可以使用 **ossl** 网络流驱动程序(OpenSSL)或 **gtls** 流驱动程序(GnuTLS)。



注意

如果您的系统具有更高的安全性，例如，没有连接到任何网络或有严格授权的系统，请使用独立的系统作为认证授权(CA)。

您可以使用 **全局**、**模块** 和 **输入** 级别在服务器端使用流驱动程序在 **全局** 和操作级别上自定义连接设置。**更为具体的配置会覆盖更常规的配置。例如，您可以对大多数连接和 gtls 在全局设置中使用 ossl，而只为特定连接使用 gtls。**

先决条件

- **有对客户端和服务器系统的 root 访问权限。**
- **以下软件包安装在服务器和客户端系统中：**
  - **rsyslog 软件包。**
  - **对于 ossl 网络流驱动程序，rsyslog-openssl 软件包。**
  - **对于 gtls 网络流驱动程序，rsyslog-gnutls 软件包。**
  - **要使用 certtool 命令(gnutls-utils 软件包)生成证书。**
- **在您的日志服务器中，以下证书位于 /etc/pki/ca-trust/source/anchors/ 目录中，并使用 update-ca-trust 命令更新您的系统配置：**

- ca-cert.pem - 一个 CA 证书，它可以在日志记录服务器和客户端上验证密钥和证书。
- server-cert.pem - 日志记录服务器的公钥。
- server-key.pem - 日志记录服务器的私钥。

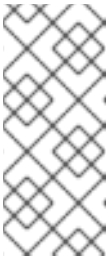
- 在您的日志记录客户端中，以下证书位于 /etc/pki/ca-trust/source/anchors/ 目录中，并使用 update-ca-trust 来更新您的系统配置：
  - ca-cert.pem - 一个 CA 证书，它可以在日志记录服务器和客户端上验证密钥和证书。
  - client-cert.pem - 客户端的公钥。
  - client-key.pem - 客户端的私钥。
  - 如果服务器运行 RHEL 9.2 或更高版本，且启用了 FIPS 模式，客户端必须支持扩展 Master Secret (EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

流程

1. 配置服务器以从您的客户端系统接收加密日志：
   a. 在 /etc/rsyslog.d/ 目录中创建一个新文件，例如 securelogser.conf。
   b. 要加密通信，配置文件必须包含指向服务器的证书文件的路径、所选身份验证方法，以及支持 TLS 加密的流驱动程序。在 /etc/rsyslog.d/securelogser.conf 文件中添加以下行：

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/server-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/server-key.pem"
)
```

```
# TCP listener
module(
  load="imtcp"
  PermittedPeer=["client1.example.com", "client2.example.com"]
  StreamDriver.AuthMode="x509/name"
  StreamDriver.Mode="1"
  StreamDriver.Name="ossl"
)

# Start up listener at port 514
input(
  type="imtcp"
  port="514"
)
```



注意

如果您更喜欢 GnuTLS 驱动程序，请使用 StreamDriver.Name="gtls" 配置选项。有关比 x509/name 严格性低的验证模式的更多信息，请参阅使用 rsyslog-doc 软件包安装的文档。

c. **可选：在 RHEL 9.4 中提供的 Rsyslog 版本 8.2310 中，您可以自定义连接配置。要做到这一点，将 input 部分替换为以下内容：**

```
input(
  type="imtcp"
  Port="50515"
  StreamDriver.Name="<driver>"
  streamdriver.CAFile="/etc/rsyslog.d/<ca1>.pem"
  streamdriver.CertFile="/etc/rsyslog.d/<server1-cert>.pem"
  streamdriver.KeyFile="/etc/rsyslog.d/<server1-key>.pem"
)
```

- **根据您要使用的 *驱动程序*，将 <driver> 替换为 ossl 或 gtls。**
- **将 *<ca1&* gt; 替换为 CA 证书，*<server1-cert>* 替换为证书，*<server1-key>* 替换为自定义连接的密钥。**

d. **将更改保存到 /etc/rsyslog.d/securelogser.conf 文件。**

e. **验证 /etc/rsyslog.conf 文件的语法以及 /etc/rsyslog.d/ 目录中的任何文件：**

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

f. **确保 rsyslog 服务在日志记录服务器中运行并启用：**

```
# systemctl status rsyslog
```

g. **重启 rsyslog 服务：**

```
# systemctl restart rsyslog
```

h. **可选：如果没有启用 Rsyslog，请确保 rsyslog 服务在重启后自动启动：**

```
# systemctl enable rsyslog
```

2. **配置客户端以将加密日志发送到服务器：**

a. **在客户端系统上，在 /etc/rsyslog.d/ 目录中创建一个名为 的新文件，如 securelogcli.conf。**

b. **在 /etc/rsyslog.d/securelogcli.conf 文件中添加以下行：**

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/client-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/client-key.pem"
)


# Set up the action for all messages
*.* action(
  type="omfwd"
  StreamDriver="ossl"
  StreamDriverMode="1"
  StreamDriverPermittedPeers="server.example.com"
  StreamDriverAuthMode="x509/name"
  target="server.example.com" port="514" protocol="tcp"
)
```

注意

如果您更喜欢 GnuTLS 驱动程序，请使用 StreamDriver.Name="gtls" 配置选项。

c. **可选：在 RHEL 9.4 中提供的 Rsyslog 版本 8.2310 中，您可以自定义连接配置。要做到这一点，将 action 部分替换为以下内容：**

```
local1.* action(
  type="omfwd"
  StreamDriver="<driver>"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  streamDriver.CAFile="/etc/rsyslog.d/<ca1>.pem"
  streamDriver.CertFile="/etc/rsyslog.d/<client1-cert>.pem"
  streamDriver.KeyFile="/etc/rsyslog.d/<client1-key>.pem"
  target="server.example.com" port="514" protocol="tcp"
  )
```

- **根据您要使用的 *驱动程序*，将 <driver> 替换为 ossl 或 gtls。**
- **将 *<ca1&* gt; 替换为 CA 证书，*<client1-cert* > 替换为证书，<client1-key > 替换为自定义连接的密钥。**

d. **将更改保存到 /etc/rsyslog.d/securelogcli.conf 文件。**

e. **验证 /etc/rsyslog.conf 文件的语法以及 /etc/rsyslog.d/ 目录中的其他文件：**

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

f. **确保 rsyslog 服务在日志记录服务器中运行并启用：**

```
# systemctl status rsyslog
```

g. **重启 rsyslog 服务：**

```
# systemctl restart rsyslog
```

-

h. 可选：如果没有启用 Rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统上，查看 /var/log/messages 日志，例如：

```
# cat /var/log/remote/msg/<hostname>/root.log
Feb 25 03:53:17 <hostname> root[6064]: test
```

其中 < hostname> 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- certtool (1), openssl (1), update-ca-trust (8), rsyslogd(8) 和 rsyslog.conf (5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 上安装了 rsyslog-doc 软件包的文档。
- 将 logging 系统角色与 TLS 一起使用。

15.6. 配置服务器以通过 UDP 接收远程日志信息

Rsyslog 应用程序可让您将系统配置为从远程系统接收日志信息。要通过 UDP 使用远程日志记录，请同时配置服务器和客户端。接收服务器收集并分析一个或多个客户端系统发送的日志。默认情况下，rsyslog 使用端口 514 上的 UDP 从远程系统接收日志信息。

按照以下步骤配置服务器，以通过 UDP 协议收集和分析一个或多个客户端系统发送的日志。

先决条件

- rsyslog 已安装在服务器系统上。
- 您以 root 身份登录到服务器中。
- 使用 semanage 命令，为可选步骤安装 policycoreutils-python-utils 软件包。
- firewalld 服务在运行。

流程

1. 可选：对于与默认端口 514 不同的 rsyslog 流量，请使用其他端口：

   a. 将 syslogd_port_t SELinux 类型添加到 SELinux 策略配置中，使用您要 rsyslog 的端口号替换 *portno* ：

   ```
   # semanage port -a -t syslogd_port_t -p udp portno
   ```

   b. 配置 firewalld 以允许传入的 rsyslog 流量，使用您要 rsyslog 使用的端口替换 *portno*，区替换 zone：

   ```
   # firewall-cmd --zone=zone --permanent --add-port=portno/udp
   success
   # firewall-cmd --reload
   ```

   c. 重新载入防火墙规则：

   ```
   # firewall-cmd --reload
   ```

2. 在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，如 remotelogserv.conf，并插入以下内容：

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides UDP syslog reception
module(load="imudp")

# This ruleset processes remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imudp" port="514" ruleset="remote1")
```

其中 514 是 rsyslog 默认使用的端口号。您可以指定不同的端口。

3. 验证 **/etc/rsyslog.conf** 文件以及 **/etc/rsyslog.d/** 目录中的所有 **.conf** 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
```

4. 重新启动 **rsyslog** 服务。

```
# systemctl restart rsyslog
```

5. 可选：如果没有启用 **rsyslog**，请确保 **rsyslog** 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

其他资源

- rsyslogd(8) , rsyslog.conf(5), semanage(8), 和 firewall-cmd(1) man page。
- 在 /usr/share/doc/rsyslog/html/index.html 文件中通过 rsyslog-doc 软件包安装的文档。

15.7. 通过 UDP 配置远程日志记录到服务器

按照以下步骤配置通过 UDP 协议将日志消息转发到服务器。omfwd 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。因为插件内置在内，所以不必加载它。

先决条件

- rsyslog 软件包安装在应该向服务器报告的客户端系统上。
- 您已为远程日志记录配置了服务器，如配置服务器 以通过 UDP 接收远程日志信息。

流程

1. 在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，如 10-remotelogcli.conf，并插入以下内容：

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="portno" protocol="udp"
    )
```

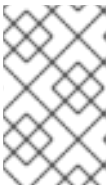
其中：

- queue.type="linkedlist" 设置启用一个 LinkedList 内存中队列。
- queue.filename 设置定义磁盘存储。备份文件使用之前全局 workDirectory 指令指定的工作目录中的 *example_fwd* 前缀创建。
- action.resumeRetryCount -1 设置可防止 rsyslog 在重试时丢弃消息（如果服务器没

有响应）。

- 如果 rsyslog 关闭，启用的 queue.saveOnShutdown="on" 设置会保存内存中的数据。
- *portno* 值是您要 rsyslog 使用的端口号。默认值为 514。
- 最后一行将所有收到的消息转发到日志记录服务器，端口规格是可选的。

使用这个配置，rsyslog 会向服务器发送消息，但如果远程服务器无法访问，则会将消息保留在内存中。只有 rsyslog 耗尽配置的内存队列空间或需要关闭时，才能创建磁盘上的文件，从而让系统性能受益。



注意

rsyslog 按照一般顺序处理配置文件 /etc/rsyslog.d/。

2. 重新启动 rsyslog 服务。

```
# systemctl restart rsyslog
```

3. 可选：如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统中，查看 /var/log/remote/msg/*hostname*/root.log 日志，例如：

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

其中 hostname 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- rsyslogd(8) 和 rsyslog.conf(5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 上安装了 rsyslog-doc 软件包的文档。

15.8. RSYSLOG 中的负载均衡帮助程序

RebindInterval 设置指定当前连接中断的时间间隔，并被重新建立。此设置适用于 TCP、UDP 和 RELP 流量。负载平衡器将信息作为新连接，并将消息转发到另一个物理目标系统。

当目标系统更改其 IP 地址时，RebindInterval 设置非常有用。Rsyslog 应用程序在连接建立时缓存 IP 地址，因此信息会发送到同一服务器。如果 IP 地址更改，UDP 数据包将会丢失，直到 Rsyslog 服务重启为止。重新建立连接将确保 DNS 再次解析 IP。

```
action(type="omfwd" protocol="tcp" RebindInterval="250" target="example.com" port="514" …)

action(type="omfwd" protocol="udp" RebindInterval="250" target="example.com" port="514" …)

action(type="omrelp" RebindInterval="250" target="example.com" port="6514" …)
```

15.9. 配置可靠的远程日志记录

通过可靠的事件日志记录协议(RELP)，您可以降低消息丢失的风险通过 TCP 发送和接收 syslog 消息。RELP 提供可靠的事件消息交付，这对于无法接受消息丢失的环境中非常有用。要使用 RELP，请配置服务器上运行的 imrelp 输入模块并接收日志，以及在客户端上运行的 omrelp 输出模块，并将日志发送到日志记录服务器。

先决条件

- 您已在服务器和客户端系统中安装了 rsyslog、librelp 和 rsyslog-relp 软件包。

- 在 SELinux 中允许指定的端口并在防火墙中打开。

流程

1. 配置客户端系统以可靠远程记录：

   a. 在客户端系统上，在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，例如 relpclient.conf，并插入以下内容：

   ```
   module(load="omrelp")
   *.* action(type="omrelp" target="_target_IP_" port="_target_port_")
   ```

   其中：

   - *target_IP* 是日志记录服务器的 IP 地址。
   - *target_port* 是日志记录服务器的端口。

   b. 保存对 /etc/rsyslog.d/relpclient.conf 文件的更改。

   c. 重新启动 rsyslog 服务。

   ```
   # systemctl restart rsyslog
   ```

   d. 可选：如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

   ```
   # systemctl enable rsyslog
   ```

2. 配置服务器系统以可靠远程记录：

   a. 在服务器系统中，在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，例如 reserv.conf，并插入以下内容：

   ```
   ruleset(name="relp"){
   ```

```
*.* action(type="omfile" file="_log_path_")
}


module(load="imrelp")
input(type="imrelp" port="_target_port_" ruleset="relp")
```

其中：

- *log_path* 指定存储消息的路径。
- *target_port* 是日志记录服务器的端口。使用与客户端配置文件中相同的值。

b. 保存对 /etc/rsyslog.d/relpserv.conf 文件的更改。

c. 重新启动 rsyslog 服务。

```
# systemctl restart rsyslog
```

d. 可选：如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统中，查看指定 *log_path* 的日志，例如：

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

其中 hostname 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- rsyslogd(8) 和 rsyslog.conf(5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 文件中通过 rsyslog-doc 软件包安装的文档。

15.10. 支持的 RSYSLOG 模块

要扩展 Rsyslog 应用程序的功能，您可以使用特定的模块。模块提供额外的输入(Input 模块)、输出（输出模块）和其他功能。模块也可以提供在加载模块后可用的其他配置指令。

您可以输入以下命令列出您系统中安装的输入和输出模块：

```
# ls /usr/lib64/rsyslog/{i,o}m*
```

在安装了 rsyslog-doc 软件包后，您可以在 /usr/share/doc/rsyslog/html/configuration/modules/idx_output.html 文件中查看所有可用的 rsyslog 模块。

15.11. 配置 NETCONSOLE 服务为将内核信息记录到远程主机

当无法登录到磁盘或使用串行控制台时，您可以使用 netconsole 内核模块和同名的服务将内核消息通过网络记录到远程 rsyslog 服务中。

先决条件

- 远程主机上已安装系统日志服务，如 rsyslog。
- 远程系统日志服务被配置为接收来自此主机的日志条目。

流程

1. 安装 netconsole-service 软件包：

```
# dnf install netconsole-service
```

2. 编辑 /etc/sysconfig/netconsole 文件，并将 SYSLOGADDR 参数设为远程主机的 IP 地址：

```
# SYSLOGADDR=192.0.2.1
```

3. 启用并启动 netconsole 服务：

```
# systemctl enable --now netconsole
```

验证步骤

- 在远程系统日志服务器上显示 /var/log/messages 文件。

其他资源

配置远程日志记录解决方案

15.12. 其他资源

- 在 /usr/share/doc/rsyslog/html/index.html 文件中安装有 rsyslog-doc 软件包的文档
- rsyslog.conf (5) 和 rsyslogd (8) man page
- 在不使用 journald 或最小化 journald 的情况下配置系统日志记录知识库文章。
- RHEL 默认日志设置对性能的负面影响及其缓解措施
- 使用日志记录系统角色 章节

第 16 章 使用 LOGGING 系统角色

作为系统管理员，您可以使用 logging 系统角色将 Red Hat Enterprise Linux 主机配置为日志服务器，以从多个客户端系统收集日志。

16.1. LOGGING RHEL 系统角色

使用 logging RHEL 系统角色，您可以在本地和远程主机上部署日志记录配置。

日志记录解决方案提供多种读取日志和多个日志记录输出的方法。

例如，日志记录系统可接受以下输入：

- 本地文件
- systemd/journal
- 网络上的另一个日志记录系统

另外，日志记录系统还可有以下输出：

- 日志存储在 /var/log 目录中的本地文件中
- 日志被发送到 Elasticsearch
- 日志被转发到另一个日志系统

使用 logging RHEL 系统角色，您可以组合输入和输出以适应您的场景。例如，您可以配置一个日志解决方案，将来自 日志 的输入存储在本地文件中，而从文件读取的输入则被转发到另一个日志系统，并存储在本地日志文件中。

其他资源

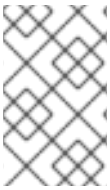
- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录
- RHEL 系统角色

16.2. 应用本地 LOGGING RHEL 系统角色

准备并应用 Ansible playbook，以便对一组独立的机器配置日志记录解决方案。每台机器在本地记录日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。



注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying basics input and implicit files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: system_input
```

```
    type: basics
  logging_outputs:
    - name: files_output
      type: files
  logging_flows:
    - name: flow1
      inputs: [system_input]
      outputs: [files_output]
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 测试 /etc/rsyslog.conf 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. 验证系统是否向日志发送信息：

   a. 发送测试信息：

   ```
   # logger test
   ```

   b. 查看 /var/log/messages 日志，例如：

   ```
   # cat /var/log/messages
   Aug  5 13:48:31 <hostname> root[6778]: test
   ```

   其中 *<hostname>* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用

户的用户名，本例中为 root。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

16.3. 使用本地 LOGGING RHEL 系统角色过滤日志

您可以部署一个日志解决方案，该方案基于 rsyslog 属性的过滤器过滤日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying files input and configured files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
```

```
logging_outputs:
  - name: files_output0
    type: files
    property: msg
    property_op: contains
    property_value: error
    path: /var/log/errors.log
  - name: files_output1
    type: files
    property: msg
    property_op: "!contains"
    property_value: error
    path: /var/log/others.log
logging_flows:
  - name: flow0
    inputs: [files_input]
    outputs: [files_output0, files_output1]
```

使用这个配置，包含 error 字符串的所有消息都被记录在 /var/log/errors.log 中，所有其他消息都被记录在 /var/log/others.log 中。

您可以将 error 属性值替换为您要过滤的字符串。

您可以根据您的偏好修改变量。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 测试 /etc/rsyslog.conf 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. 验证系统是否向日志发送包含 error 字符串的信息：

   a. 发送测试信息：

   ```
   # logger error
   ```

   b. 查看 /var/log/errors.log 日志，例如：

   ```
   # cat /var/log/errors.log
   Aug  5 13:48:31 hostname root[6778]: error
   ```

   其中 *hostname* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

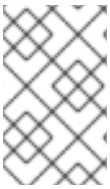
16.4. 使用 LOGGING RHEL 系统角色应用一个远程日志解决方案

按照以下步骤准备并应用 Red Hat Ansible Core playbook 来配置远程日志记录解决方案。在本 playbook 中，一个或多个客户端从 systemd-journal 获取日志，并将它们转发到远程服务器。服务器从 remote_rsyslog 和 remote_files 接收远程输入，并将日志输出到由远程主机名命名的目录中的本地文件。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。

- 用于连接到受管节点的帐户具有 sudo 权限。



注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: managed-node-02.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: <host1.example.com>
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: <host1.example.com>
```

```
      tcp_port: 601
  logging_flows:
    - name: flows0
      inputs: [basic_input]
      outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

其中 *<host1.example.com>* 是日志记录服务器。

注意

您可以修改 playbook 中的参数以符合您的需要。

警告

日志解决方案只适用于在服务器或者客户端系统的 SELinux 策略中定义的端口并在防火墙中打开。默认 SELinux 策略包括端口 601、514、6514、10514 和 20514。要使用其他端口，请修改客户端和服务器系统上的 SELinux 策略。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 在客户端和服务器系统上测试 /etc/rsyslog.conf 文件的语法：

-

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 验证客户端系统向服务器发送信息：

   a. 在客户端系统中发送测试信息：

   ```
   # logger test
   ```

   b. 在服务器系统上，查看 /var/log/*<host2.example.com>*/messages 日志，例如：

   ```
   # cat /var/log/<host2.example.com>/messages
   Aug  5 13:48:31 <host2.example.com> root[6778]: test
   ```

   其中 *<host2.example.com>* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

16.5. 使用带有 TLS 的 LOGGING RHEL 系统角色

传输层安全性(TLS)是一种加密协议，旨在允许计算机网络上的安全通信。

作为管理员，您可以使用 logging RHEL 系统角色来使用 Red Hat Ansible Automation Platform 配置日志的安全传输。

16.5.1. 配置带有 TLS 的客户端日志

您可以使用 Ansible playbook 和 logging RHEL 系统角色，来在 RHEL 客户端上配置日志记录，并使用 TLS 加密将日志传送到远程日志记录系统。

此流程创建一个私钥和证书，并在 Ansible 清单中客户端组的所有主机上配置 TLS。TLS 对信息的传输进行加密，确保日志在网络安全传输。

注意

您不必在 playbook 中调用 certificate RHEL 系统角色来创建证书。logging RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。
- 受管节点已在 IdM 域中注册。
- 如果要在管理节点上配置的日志服务器运行 RHEL 9.2 或更高版本，且启用了 FIPS 模式，则客户端必须支持Extended Master Secre(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying files input and forwards output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
```

```
    ca: ipa
  logging_pki_files:
   - ca_cert: /local/path/to/ca_cert.pem
     cert: /local/path/to/logging_cert.pem
     private_key: /local/path/to/logging_cert.pem
  logging_inputs:
   - name: input_name
     type: files
     input_log_path: /var/log/containers/*.log
  logging_outputs:
   - name: output_name
     type: forwards
     target: your_target_host
     tcp_port: 514
     tls: true
     pki_authmode: x509/name
     permitted_server: 'server.example.com'
  logging_flows:
   - name: flow_name
     inputs: [input_name]
     outputs: [output_name]
```

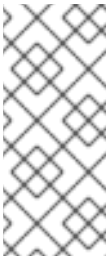
playbook 使用以下参数：

logging_certificates

此参数的值被传给 certificate RHEL 系统角色中的 certificate_requests，并用来创建私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找 CA 证书和用于 TLS 的密钥文件的路径和其他设置，使用以下子参数指定：ca_cert、ca_cert_src、cert、cert_src、private_key、private_key_src 和 tls。



注意

如果您使用 logging_certificates 在目标节点上创建文件，请不要使用 ca_cert_src、cert_src 和 private_key_src，它们用于复制由 logging_certificates 创建的文件。

ca_cert

代表目标节点上 CA 证书文件的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

代表目标节点上私钥文件的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

代表控制节点上 CA 证书文件的路径，该路径被复制到目标主机上由 ca_cert 指定的位置。如果使用 logging_certificates，请不要使用它。

cert_src

代表控制节点上证书文件的路径，该文件的路径被复制到目标主机上由 cert 指定的位置。如果使用 logging_certificates，请不要使用它。

private_key_src

代表控制节点上私钥文件的路径，该路径被复制到目标主机上由 private_key 指定的位置。如果使用 logging_certificates，请不要使用它。

tls

将此参数设为 true 以确保通过网络安全地传输日志。如果您不想要一个安全的包装程序，您可以设置 tls: false。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件

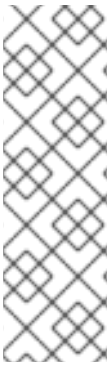
- /usr/share/doc/rhel-system-roles/logging/ 目录

- 使用 RHEL 系统角色请求证书。

16.5.2. 配置带有 TLS 的服务器日志

您可以使用 Ansible playbook 和 logging RHEL 系统角色，来在 RHEL 服务器上配置日志记录，并将它们设置为使用 TLS 加密从远程日志记录系统接收日志。

此流程创建一个私钥和证书，并在 Ansible 清单中服务器组中的所有主机上配置 TLS。



注意

您不必在 playbook 中调用 certificate RHEL 系统角色来创建证书。logging RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

先决条件

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。

- 用于连接到受管节点的帐户具有 sudo 权限。

- 受管节点已在 IdM 域中注册。

- 如果要在管理节点上配置的日志服务器运行 RHEL 9.2 或更高版本，且启用了 FIPS 模式，则客户端必须支持Extended Master Secre(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

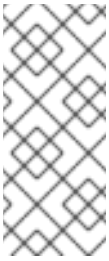
playbook 使用以下参数：

logging_certificates

此参数的值被传给 certificate RHEL 系统角色中的 certificate_requests，并用来创建私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找 CA 证书和用于 TLS 的密钥文件的路径和其他设置，使用以下子参数指定：ca_cert、ca_cert_src 、cert、cert_src、private_key、private_key_src 和 tls。



注意

如果您使用 **logging_certificates** 在目标节点上创建文件，请不要使用 **ca_cert_src**、**cert_src** 和 **private_key_src**，它们用于复制由 **logging_certificates** 创建的文件。

ca_cert

代表目标节点上 **CA** 证书文件的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。

private_key

代表目标节点上私钥文件的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

ca_cert_src

代表控制节点上 **CA** 证书文件的路径，该路径被复制到目标主机上由 **ca_cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

cert_src

代表控制节点上证书文件的路径，该文件的路径被复制到目标主机上由 **cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

private_key_src

代表控制节点上私钥文件的路径，该路径被复制到目标主机上由 **private_key** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

tls

将此参数设为 **true** 以确保通过网络安全地传输日志。如果您不想要一个安全的包装程序，您可以设置 **tls: false**。

2. 验证 **playbook** 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录
- 使用 RHEL 系统角色请求证书。

16.6. 使用带有 RELP 的 LOGGING RHEL 系统角色

可靠的事件日志协议(RELP)是一种通过 TCP 网络记录数据和消息的网络协议。它确保了事件消息的可靠传递，您可以在不容许任何消息丢失的环境中使用它。

RELP 发送者以命令的形式传输日志条目，接收者在处理后确认这些条目。为确保一致性，RELP 将事务数保存到传输的命令中，以便进行任何类型的消息恢复。

您可以考虑在 RELP 客户端和 RELP Server 间的远程日志系统。RELP 客户端将日志传送给远程日志系统，RELP 服务器接收由远程日志系统发送的所有日志。

管理员可以使用 logging RHEL 系统角色将日志记录系统配置为可靠地发送和接收日志条目。

16.6.1. 配置带有 RELP 的客户端日志

您可以使用 logging RHEL 系统角色在 RHEL 系统中配置日志记录，这些日志记录在本地机器上，并可以通过运行 Ansible playbook 将日志转发到带有 RELP 的远程日志记录系统。

此流程对 Ansible 清单中 客户端 组中的所有主机配置 RELP。RELP 配置使用传输层安全(TLS)来加密消息传输，保证日志在网络上安全传输。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying basic input and relp output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: logging.server.com
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/client-cert.pem
        private_key: /etc/pki/tls/private/client-key.pem
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: example_flow
        inputs: [basic_input]
        outputs: [relp_client]
```

playbook 使用以下设置：

target

这是一个必需的参数，用于指定运行远程日志系统的主机名。

port

远程日志记录系统正在监听的端口号。

tls

确保日志在网络上安全地传输。如果您不想要安全打包程序，可以将 tls 变量设置为 false。在与 RELP 工作时，默认的 tls 参数被设置为 true，且需要密钥/证书和 triplets {ca_cert、cert、private_key} 和/或 {ca_cert_src,cert_src,private_key_src}。

- 如果设置了 {ca_cert_src,cert_src,private_key_src} 三元组，则使用默认位置 /etc/pki/tls/certs 和 /etc/pki/tls/private 作为受管节点上的目的地，以便从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同

- 如果设置了 {ca_cert,cert,private_key} 三元组，则预期文件应在日志配置前的默认路径上。

- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

ca_cert

表示 CA 证书的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示证书的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

表示私钥的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

表示复制到目标主机的本地 CA 证书文件路径。如果指定了 ca_cert，则会将其复制到该位置。

cert_src

表示复制到目标主机的本地证书文件路径。如果指定了 cert，则会将其复制到该位置。

private_key_src

表示复制到目标主机的本地密钥文件的路径。如果指定了 private_key，则会将其复制到该位置。

pki_authmode

接受身份验证模式为 name 或 fingerprint。

permitted_servers

日志客户端允许通过 TLS 连接和发送日志的服务器列表。

输入

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

16.6.2. 配置带有 RELP 的服务器日志

您可以使用 logging RHEL 系统角色在作为服务器的 RHEL 系统中配置日志记录，并可以通过运行 Ansible playbook 从具有 RELP 的远程日志记录系统接收日志。

此流程对 Ansible 清单中 服务器 组中的所有主机配置 RELP。RELP 配置使用 TLS 加密消息传输，以保证在网络上安全地传输日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/server-cert.pem
        private_key: /etc/pki/tls/private/server-key.pem
        pki_authmode: name
        permitted_clients:
          - '*example.client.com'
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: example_flow
        inputs: relp_server
        outputs: remote_files_output
```

-

playbook 使用以下设置：

port

远程日志记录系统正在监听的端口号。

tls

确保日志在网络上安全地传输。如果您不想要安全打包程序，可以将 tls 变量设置为 false。在与 RELP 工作时，默认的 tls 参数被设置为 true，且需要密钥/证书和 triplets {ca_cert、cert、private_key} 和/或 {ca_cert_src,cert_src,private_key_src}。

- 如果设置了 {ca_cert_src,cert_src,private_key_src} 三元组，则使用默认位置 /etc/pki/tls/certs 和 /etc/pki/tls/private 作为受管节点上的目的地，以便从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同

- 如果设置了 {ca_cert,cert,private_key} 三元组，则预期文件应在日志配置前的默认路径上。

- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

ca_cert

表示 CA 证书的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示证书的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

表示私钥的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

表示复制到目标主机的本地 CA 证书文件路径。如果指定了 ca_cert，则会将其复制到该位置。

cert_src

表示复制到目标主机的本地证书文件路径。如果指定了 cert，则会将其复制到该位置。

private_key_src

表示复制到目标主机的本地密钥文件的路径。如果指定了 private_key，则会将其复制到该位置。

pki_authmode

接受身份验证模式为 name 或 fingerprint。

permitted_clients

日志记录服务器允许通过 TLS 连接和发送日志的客户端列表。

输入

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

16.7. 其他资源

- 准备一个控制节点和受管节点以使用 RHEL 系统角色
- 随 rhel-system-roles 软件包安装在 /usr/share/ansible/roles/rhel-system-roles.logging/README.html 中的文档。
- RHEL 系统角色
- ansible-playbook(1) 手册页。