Red Hat Enterprise Linux 9

安全网络

配置安全网络和网络通信

Last Updated: 2024-07-13

Red Hat Enterprise Linux 9 安全网络

配置安全网络和网络通信

法律通告

摘要

了解用于提高网络安全性的工具和技术，并降低数据泄露和入侵的风险。

目录

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. 在顶部导航栏中点 **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。包括到文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 使用 OPENSSH 的两个系统间使用安全通讯

SSH(Secure Shell)是一种协议，它使用客户端-服务器架构在两个系统之间提供安全通信，并允许用户远程登录到服务器主机系统。和其它远程沟通协议，如 FTP 或 Telnet 不同，SSH 会加密登录会话，它会阻止入侵者从连接中收集未加密的密码。

Red Hat Enterprise Linux 包括基本的 **OpenSSH** 软件包：通用的 **openssh** 软件包、**openssh-server** 软件包以及 **openssh-clients** 软件包。请注意，**OpenSSH** 软件包需要 **OpenSSL** 软件包 **openssl-libs**，它会安装几个重要的加密库来启用 **OpenSSH** 对通讯进行加密。

1.1. SSH 和 OPENSSH

SSH（安全 Shell）是一个登录远程机器并在该机器上执行命令的程序。SSH 协议通过不安全的网络在两个不可信主机间提供安全加密的通讯。您还可以通过安全频道转发 X11 连接和任意 TCP/IP 端口。

当使用 SSH 协议进行远程 shell 登录或文件复制时，SSH 协议可以缓解威胁，例如，拦截两个系统之间的通信和模拟特定主机。这是因为 SSH 客户端和服务器使用数字签名来验证其身份。另外，所有客户端和服务器系统之间的沟通都是加密的。

主机密钥验证使用 SSH 协议的主机。当首次安装 OpenSSH 或主机第一次引导时，主机密钥是自动生成的加密密钥。

OpenSSH 是 Linux、UNIX 和类似操作系统支持的 SSH 协议的实现。它包括 OpenSSH 客户端和服务器需要的核心文件。OpenSSH 组件由以下用户空间工具组成：

- **ssh** 是一个远程登录程序（SSH 客户端）.
- **sshd** 是一个 OpenSSH SSH 守护进程。
- **scp** 是一个安全的远程文件复制程序。
- **sftp** 是一个安全的文件传输程序。
- **ssh-agent** 是用于缓存私钥的身份验证代理。
- **ssh-add** 为 **ssh-agent**添加私钥身份。
- **ssh-keygen** 生成、管理并转换 **ssh** 验证密钥。
- **ssh-copy-id** 是一个将本地公钥添加到远程 SSH 服务器上的 **authorized_keys** 文件中的脚本。
- **ssh-keyscan** 可以收集 SSH 公共主机密钥。

注意

在 RHEL 9 中，安全复制协议(SCP)默认替换为 SSH 文件传输协议(SFTP)。这是因为 SCP 已经造成安全问题，如 CVE-2020-15778。

如果您的环境无法使用 SFTP 或存在不兼容的情况，您可以使用 **-O** 选项来强制使用原始 SCP/RCP 协议。

如需更多信息，请参阅 Red Hat Enterprise Linux 9 文档中的 OpenSSH SCP 协议弃用 。

RHEL 中的 OpenSSH 套件仅支持 SSH 版本 2。它有一个增强的密钥交换算法，其不会受到较旧版本 1 中已知的漏洞的影响。

OpenSSH 作为 RHEL 的核心加密子系统之一，使用系统范围的加密策略。这样可确保在默认配置中禁用弱密码套件和加密算法。要修改策略，管理员必须使用 **update-crypto-policies** 命令来调整设置，或者手动选择不使用系统范围的加密策略。

OpenSSH 套件使用两组配置文件：一个用于客户端程序（即 **ssh**、**scp** 和 **sftp**），另一个用于服务器（**sshd** 守护进程）。

系统范围的 SSH 配置信息保存在 **/etc/ssh/** 目录中。用户特定的 SSH 配置信息保存在用户主目录中的 **~/.ssh/** 中。有关 OpenSSH 配置文件的详细列表，请查看 **sshd(8)** man page 中的 **FILES** 部分。

其他资源

- 使用 **man -k ssh** 命令显示 man page
- 使用系统范围的加密策略

1.2. 配置并启动 OPENSSH 服务器

您可以更改 OpenSSH 服务器的欢迎横幅和 IP 地址。您还可以切换到较慢的动态网络配置。

请注意，在默认 RHEL 安装后，**sshd** 守护进程已经启动，服务器主机密钥会被自动创建。

先决条件

- 已安装 **openssh-server** 软件包。

流程

1. 如果 **sshd** 服务还没有运行，请在当前会话中启动它，并将其设置为在引导时自动启动：

```
# systemctl enable --now sshd
```

2. 指定与默认值（**0.0.0.0** (IPv4) 或 **::**）不同的地址(IPv6) **/etc/ssh/sshd_config** 配置文件中的 **ListenAddress** 指令，并使用较慢的动态网络配置，将 **network-online.target** 目标单元的依赖关系添加到 **sshd.service** 单元文件。要做到这一点，使用以下内容创建 **/etc/systemd/system/sshd.service.d/local.conf** 文件：

```
[Unit]
Wants=network-online.target
After=network-online.target
```

3. 查看 **/etc/ssh/sshd_config** 配置文件中的 OpenSSH 服务器设置是否满足您的情况要求。

4. 另外，还可通过编辑 **/etc/issue** 文件来更改您的 OpenSSH 服务器在客户端验证前显示的欢迎信息，例如：

```
Welcome to ssh-server.example.com
Warning: By accessing this server, you agree to the referenced terms and conditions.
```

确保 **/etc/ssh/sshd_config** 中未注释掉 **Banner** 选项，并且其值包含 **/etc/issue** ：

```
# less /etc/ssh/sshd_config | grep Banner
Banner /etc/issue
```

请注意：要在成功登录后改变显示的信息，您必须编辑服务器上的 **/etc/motd** 文件。详情请查看 **pam_motd** man page。

5. 重新载入 **systemd** 配置，并重启 **sshd** 以应用修改：

```
# systemctl daemon-reload
# systemctl restart sshd
```

验证

1. 检查 **sshd** 守护进程是否正在运行：

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-11-18 14:59:58 CET; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1149 (sshd)
    Tasks: 1 (limit: 11491)
   Memory: 1.9M
   CGroup: /system.slice/sshd.service
           └─1149 /usr/sbin/sshd -D -oCiphers=aes128-ctr,aes256-ctr,aes128-cbc,aes256-cbc -
oMACs=hmac-sha2-256,>

Nov 18 14:59:58 ssh-server-example.com systemd[1]: Starting OpenSSH server daemon...
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on 0.0.0.0 port 22.
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on :: port 22.
Nov 18 14:59:58 ssh-server-example.com systemd[1]: Started OpenSSH server daemon.
```

2. 使用 SSH 客户端连接到 SSH 服务器。

```
# ssh user@ssh-server-example.com
ECDSA key fingerprint is SHA256:dXbaS0RG/UzlTTku8GtXSz0S1++lPegSy31v3L/FAEc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ssh-server-example.com' (ECDSA) to the list of known hosts.

user@ssh-server-example.com's password:
```

其他资源

- **sshd(8)** 和 **sshd_config(5)** 手册页。

1.3. 为基于密钥的身份验证设置 OPENSSH 服务器

要提高系统安全性，通过在 OpenSSH 服务器上禁用密码身份验证来强制进行基于密钥的身份验证。

先决条件

- 已安装 **openssh-server** 软件包。
- **sshd** 守护进程正在服务器中运行。

流程

1. 在文本编辑器中打开 **/etc/ssh/sshd_config** 配置，例如：

```
# vi /etc/ssh/sshd_config
```

2. 将 **PasswordAuthentication** 选项改为 **no**:

```
PasswordAuthentication no
```

在非新默认安装的系统上，请检查 **PubkeyAuthentication no** 是否没有设置，并将 **KbdInteractiveAuthentication** 指令设为 **no**。如果您要进行远程连接，而不使用控制台或带外访问，在禁用密码验证前测试基于密钥的登录过程。

3. 要在 NFS 挂载的主目录中使用基于密钥的验证，启用 **use_nfs_home_dirs** SELinux 布尔值：

```
# setsebool -P use_nfs_home_dirs 1
```

4. 重新载入 **sshd** 守护进程以应用更改：

```
# systemctl reload sshd
```

其他资源

- **sshd(8)**, **sshd_config(5)** 和 **setsebool(8)** 手册页。

1.4. 生成 SSH 密钥对

您可以通过在本地系统上生成 SSH 密钥对并将生成的公钥复制到 OpenSSH 服务器来在没有提供密码的情况下登录到 OpenSSH 服务器。必须将服务器配置为允许此选项。

先决条件

- 您以 Linux 用户身份登录，该用户将连接到 OpenSSH 服务器。如果以 **root** 身份完成以下步骤，则只有 **root** 用户才能使用该密钥。

流程

1. 为 SSH 协议的版本 2 生成 ECDSA 密钥对：

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_ecdsa.
Your public key has been saved in /home/<username>/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
<username>@<localhost.example.com>
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
```

```
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

您还可以通过输入 **ssh-keygen -t ed25519** 命令，在 **ssh-keygen** 命令或 Ed25519 密钥对中使用 **-t rsa** 选项生成 RSA 密钥对。

2. 将公钥复制到远程机器中：

```
$ ssh-copy-id <username>@<ssh-server-example.com>
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
<username>@<ssh-server-example.com>'s password:
...
Number of key(s) added: 1

Now try logging into the machine, with: "ssh '<username>@<ssh-server-example.com>'" and check to make sure that only the key(s) you wanted were added.
```

将 & ***lt;username> @ <ssh-server-example.com&*** gt; 替换为您的凭证。

如果您没有在会话中使用 **ssh-agent** 程序，上一个命令会复制最新修改的 **~/.ssh/id*.pub** 公钥。要指定另一个公钥文件，或在 **ssh-agent** 内存中缓存的密钥优先选择文件中的密钥，使用带有 **-i** 选项的 **ssh-copy-id** 命令。

3. 可选：要在重新安装系统之间保留之前生成的密钥对，备份 **~/.ssh/** 目录。重新安装后，将其复制到主目录中。您可以为系统中的所有用户（包括 **root** 用户）进行此操作。

验证

1. 在不提供任何密码的情况下登录到 OpenSSH 服务器：

```
$ ssh <username>@<ssh-server-example.com>
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

其他资源

- **ssh-keygen(1)** 和 **ssh-copy-id(1)** 手册页。

1.5. 使用保存在智能卡中的 SSH 密钥

您可以使用保存在 OpenSSH 客户端智能卡中的 RSA 和 ECDSA 密钥。使用智能卡进行验证替换了默认密码验证。

先决条件

- 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 page

1.6. 使 OPENSSH 更安全

在使用 OpenSSH 时，您可以调整系统以提高安全性。

请注意，**/etc/ssh/sshd_config** OpenSSH 配置文件的更改需要重新载入 **sshd** 守护进程才能生效：

```
# systemctl reload sshd
```

警告

大多数安全强化配置更改会降低与不支持最新算法或密码套件的客户端的兼容性。

禁用不安全的连接协议

- 要使 SSH 生效，防止使用由 OpenSSH 套件替代的不安全连接协议。否则，用户的密码可能只会在一个会话中被 SSH 保护，可能会在以后使用 Telnet 登录时被捕获。因此，请考虑禁用不安全的协议，如 telnet、rsh、rlogin 和 ftp。

启用基于密钥的身份验证并禁用基于密码的身份验证

- 禁用密码验证并只允许密钥对可减少安全攻击面，还可节省用户的时间。在客户端中，使用 **ssh-keygen** 工具生成密钥对，并使用 **ssh-copy-id** 工具从 OpenSSH 服务器的客户端复制公钥。要在 OpenSSH 服务器中禁用基于密码的验证，请编辑 **/etc/ssh/sshd_config**，并将 **PasswordAuthentication** 选项改为 **no**:

  ```
  PasswordAuthentication no
  ```

密钥类型

- 虽然 **ssh-keygen** 命令会默认生成一组 RSA 密钥，但您可以使用 **-t** 选项指定它生成 ECDSA 或者 Ed25519 密钥。ECDSA(Elliptic Curve Digital Signature Algorithm)能够在同等的对称密钥强度下，提供比 RSA 更好的性能。它还会生成较短的密钥。Ed25519 公钥算法是 一种变形的 Edwards 曲线的实现，其比 RSA、DSA 和 ECDSA 更安全，也更快。
  如果没有这些密钥，OpenSSH 会自动创建 RSA、ECDSA 和 Ed25519 服务器主机密钥。要在 RHEL 中配置主机密钥创建，请使用 **sshd-keygen@.service** 实例化服务。例如，禁用自动创建 RSA 密钥类型：

  ```
  # systemctl mask sshd-keygen@rsa.service
  ```

注意

在启用了 **cloud-init** 的镜像中，**ssh-keygen** 单元会自动禁用。这是因为 **ssh-keygen template** 服务可能会干扰 **cloud-init** 工具，并导致主机密钥生成问题。要防止这些问题，**etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** 置入配置文件禁用了 **ssh-keygen** 单元（如果 **cloud-init** 正在运行）。

- 要排除 SSH 连接的特定密钥类型，注释 **/etc/ssh/sshd_config** 中的相关行，并重新载入 **sshd** 服务。例如，只允许 Ed25519 主机密钥：

```
# HostKey /etc/ssh/ssh_host_rsa_key
# HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

重要

Ed25519 算法与 FIPS-140 不兼容，OpenSSH 在 FIPS 模式下无法与 Ed25519 密钥一起工作。

非默认端口

- 默认情况下，**sshd** 守护进程侦听 TCP 端口 22。更改端口可降低系统因自动网络扫描而受到攻击的风险，从而通过隐蔽性提高安全性。您可以使用 **/etc/ssh/sshd_config** 配置文件中的 **Port** 指令指定端口。
  您还必须更新默认 SELinux 策略以允许使用非默认端口。要做到这一点，使用 **policycoreutils-python-utils** 软件包中的 **semanage** 工具：

```
# semanage port -a -t ssh_port_t -p tcp <port_number>
```

  另外，更新 **firewalld** 配置：

```
# firewall-cmd --add-port <port_number>/tcp
# firewall-cmd --remove-port=22/tcp
# firewall-cmd --runtime-to-permanent
```

  在前面的命令中，将 *<port_number>* 替换为使用 **Port** 指令指定的新端口号。

root 登录

- 默认情况下，**PermitRootLogin** 设置为 **prohibit-password**。这强制使用基于密钥的身份验证，而不是使用密码以 root 身份登录，并通过防止暴力攻击来降低风险。

警告

以 root 用户身份登录并不是一个安全的做法，因为管理员无法审核运行哪个特权命令。要使用管理命令，请登录并使用 **sudo**。

使用 X 安全性扩展

- Red Hat Enterprise Linux 客户端中的 X 服务器不提供 X 安全性扩展。因此，当连接到带有 X11 转发的不可信 SSH 服务器时，客户端无法请求另一个安全层。大多数应用程序都无法在启用此扩展时运行。
  默认情况下，**/etc/ssh/ssh_config.d/50-redhat.conf** 文件中的 **ForwardX11Trusted** 选项被设置为 **yes**，**ssh -X remote_machine**（不信任的主机）和 **ssh -Y remote_machine**（信任的主机）命令之间没有区别。

如果您的场景根本不需要 X11 转发功能，请将 **/etc/ssh/sshd_config** 配置文件中的 **X11Forwarding** 指令设置为 **no**。

限制对特定用户、组群或者域的访问

- **/etc/ssh/sshd_config** 配置文件服务器中的 **AllowUsers** 和 **AllowGroups** 指令可让您只允许某些用户、域或组连接到您的 OpenSSH 服务器。您可以组合 **AllowUsers** 和 **Allow Groups** 来更准确地限制访问，例如：

```
AllowUsers *@192.168.1.* *@10.0.0.* !*@192.168.1.2
AllowGroups example-group
```

以上配置行接受来自 192.168.1.* 和 10.0.0.* 子网中所有用户的连接，但 192.168.1.2 地址的系统除外。所有用户都必须在 **example-group** 组中。OpenSSH 服务器拒绝所有其他连接。

OpenSSH 服务器仅允许 **/etc/ssh/sshd_config** 中通过所有 Allow 和 Deny 指令的连接。例如，如果 **AllowUsers** 指令列出的用户不是 **AllowGroups** 指令中列出的组的一部分，则用户无法登录。

请注意，使用允许列表（以 Allow 开头的指令）比使用阻止列表（以 Deny 开始的选项）更安全，因为允许列表也会阻止新的未授权的用户或组。

更改系统范围的加密策略

- OpenSSH 使用 RHEL 系统范围的加密策略，默认的系统范围的加密策略级别为当前威胁模型提供了安全设置。要使您的加密设置更严格，请更改当前的策略级别：

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

警告

如果您的系统在互联网上进行通信，则可能会因为 **FUTURE** 策略的严格设置而面临互操作性问题。

您还可以通过系统范围的加密策略只为 SSH 协议禁用特定的密码。如需更多信息，请参阅 安全强化 文档中的 使用子策略自定义系统范围的加密策略 部分。

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前缀为小于 50 的两位数的加密策略，以便它在字典中位于 **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 **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前缀为小于 50 的两位数的加密策略，以便它在字典中位于 **50-redhat.conf** 文件之前，并具有 **.conf** 后缀，例如 **49-crypto-policy-override.conf**。

其他资源

- **sshd_config(5)**、**ssh-keygen(1)**、 **crypto-policies(7)** 和 **update-crypto-policies(8)** 手册页。
- 在 安全强化 文档中 使用系统范围的加密策略 。
- 如何只为 ssh 服务禁用特定的算法和密码 文章。

1.7. 使用 SSH 跳过主机连接到远程服务器

您可以通过中间服务器（也称为跳过主机）将本地系统连接到远程服务器。

先决条件

- 跳过主机接受来自本地系统的 SSH 连接。
- 远程服务器只接受来自跳过主机的 SSH 连接。

流程

1. 通过编辑本地系统中的 **~/.ssh/config** 文件来定义跳板主机，例如：

```
Host jump-server1
  HostName jump1.example.com
```

   - **Host** 参数定义您可以在 **ssh** 命令中使用的主机的名称或别名。该值可以匹配真实的主机名，但也可以是任意字符串。
   - **HostName** 参数设置跳过主机的实际主机名或 IP 地址。

2. 使用 **ProxyJump** 指令将远程服务器跳板配置添加到本地系统上的 **~/.ssh/config** 文件中，例如：

```
Host remote-server
  HostName remote1.example.com
  ProxyJump jump-server1
```

3. 使用您的本地系统通过跳过服务器连接到远程服务器：

```
$ ssh remote-server
```

   如果省略了配置步骤 1 和 2，则上一命令等同于 **ssh -J skip-server1 remote-server** 命令。

4. 您可以指定更多的跳板服务器，您也可以在提供其完整主机名时跳过在配置文件中添加主机定义，例如：

```
$ ssh -J jump1.example.com,jump2.example.com,jump3.example.com remote1.example.com
```

   如果跳板服务器上的用户名或 SSH 端口与远程服务器上的用户名和端口不同，请只修改上一命令中的主机名表示法，例如：

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.example
.com:75 joesec@remote1.example.com:220
```

其他资源

- **ssh_config(5)** 和 **ssh(1)** 手册页.

1.8. 通过 SSH-AGENT，使用 SSH 密钥连接到远程机器

为了避免在每次发起 SSH 连接时输入密语，您可以使用 **ssh-agent** 工具缓存 SSH 私钥。确保私钥和密语安全。

先决条件

- 您有一个运行 SSH 守护进程的远程主机，并且可通过网络访问。
- 您知道登录到远程主机的 IP 地址或者主机名以及凭证。
- 您已用密码生成了 SSH 密钥对，并将公钥传送到远程机器。

流程

1. 可选：验证您可以使用密钥向远程主机进行身份验证：
   a. 使用 SSH 连接到远程主机：

   ```
   $ ssh example.user1@198.51.100.1 hostname
   ```

   b. 输入您在创建密钥时设定的密码短语以授予对私钥的访问权限。

   ```
   $ ssh example.user1@198.51.100.1 hostname
    host.example.com
   ```

2. 启动 **ssh-agent**。

   ```
   $ eval $(ssh-agent)
   Agent pid 20062
   ```

3. 将密钥添加到 **ssh-agent**。

   ```
   $ ssh-add ~/.ssh/id_rsa
   Enter passphrase for ~/.ssh/id_rsa:
   Identity added: ~/.ssh/id_rsa (example.user0@198.51.100.12)
   ```

验证

- 可选：使用 SSH 登录主机机器。

  ```
  $ ssh example.user1@198.51.100.1

  Last login: Mon Sep 14 12:56:37 2020
  ```

请注意您不必输入密码短语。

1.9. 配置与 SSH 系统角色的安全通信

作为管理员，您可以使用 **sshd** 系统角色配置 SSH 服务器和 **ssh** 系统角色，以使用 Ansible Core 软件包同时在任意数量的 RHEL 系统上配置 SSH 客户端。

1.9.1. sshd RHEL 系统角色的变量

在 **sshd** 系统角色 playbook 中，您可以根据您的首选项和限制定义 SSH 配置文件的参数。

如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的 **sshd_config** 文件。

在所有情况下，布尔值在 **sshd** 配置中都正确呈现为 **yes** 和 **no**。您可以使用 list 来定义多行配置项。例如：

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

呈现为：

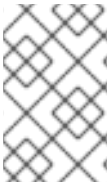
```
ListenAddress 0.0.0.0
ListenAddress ::
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.9.2. 使用 sshd RHEL 系统角色配置 OpenSSH 服务器

您可以通过运行 Ansible playbook，使用 **sshd** RHEL 系统角色来配置多个 SSH 服务器。



注意

您可以将 **sshd** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他 RHEL 系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **sshd** 角色使用命名空间(RHEL 8 和更早的版本)或 drop-in 目录(RHEL 9)。

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: SSH server configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure sshd to prevent root and password login except from particular subnet
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd:
          PermitRootLogin: no
          PasswordAuthentication: no
          Match:
            - Condition: "Address 192.0.2.0/24"
              PermitRootLogin: yes
              PasswordAuthentication: yes
```

playbook 将受管节点配置为 SSH 服务器，以便：

- 禁用密码和 **root** 用户登录
- 只对子网 **192.0.2.0/24** 启用密码和 **root** 用户登录

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. 登录到 SSH 服务器：

```
$ ssh <username>@<ssh_server>
```

2. 验证 SSH 服务器中的 **sshd_config** 文件的内容：

```
$ cat /etc/ssh/sshd_config.d/00-ansible_system_role.conf
#
# Ansible managed
#
PasswordAuthentication no
PermitRootLogin no
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. 检查您是否可以以 root 用户身份从 **192.0.2.0/24** 子网连接到服务器：

   a. 确定您的 IP 地址：

```
$ hostname -I
192.0.2.1
```

如果 IP 地址在 **192.0.2.1** - **192.0.2.254** 范围内，您可以连接到服务器。

b. 以 **root** 用户身份连接到服务器：

```
$ ssh root@<ssh_server>
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.9.3. ssh RHEL 系统角色的变量

在 **ssh** 系统角色 playbook 中，您可以根据您的首选项和限制定义客户端 SSH 配置文件的参数。

如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的全局 **ssh_config** 文件。

在所有情况下，布尔值在 **ssh** 配置中都正确地呈现为 **yes** 或 **no**。您可以使用 list 来定义多行配置项。例如：

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

呈现为：

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```

注意

配置选项区分大小写。

其他资源

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

1.9.4. 使用 ssh RHEL 系统角色配置 OpenSSH 客户端

您可以通过运行 Ansible playbook，使用 **ssh** RHEL 系统角色来配置多个 SSH 客户端。

注意

您可以将 **ssh** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **ssh** 角色使用置入目录（在 RHEL 8 及更新的版本中使用）。

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: SSH client configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: "Configure ssh clients"
      ansible.builtin.include_role:
        name: rhel-system-roles.ssh
      vars:
        ssh_user: root
        ssh:
          Compression: true
          GSSAPIAuthentication: no
          ControlMaster: auto
          ControlPath: ~/.ssh/.cm%C
          Host:
            - Condition: example
              Hostname: server.example.com
              User: user1
        ssh_ForwardX11: no
```

此 playbook 使用以下配置在受管节点上配置 **root** 用户的 SSH 客户端首选项：

- 压缩已启用。
- ControlMaster 多路复用设置为 **auto**。
- 连接到 **server. example.com** 主机的示例别名是 **user1**。
- 创建 **示例** 主机别名，它代表使用 **user1** 用户名连接到 **server.example.com** 主机。
- X11 转发被禁用。

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

- 通过显示 SSH 配置文件来验证受管节点是否具有正确的配置：

```
# cat ~/root/.ssh/config
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

其他资源

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

1.9.5. 将 sshd RHEL 系统角色用于非独占配置

通常，应用 **sshd** 系统角色会覆盖整个配置。如果您之前已调整了配置，例如使用不同的系统角色或 playbook，这可能会出现问题。要只对所选配置选项应用 **sshd** 系统角色，同时保留其他选项，您可以使用非排除配置。

您可以应用非独占配置：

- 在 RHEL 8 及更早版本中，使用配置片断。
- 在 RHEL 9 及更高版本中，使用置入目录中的文件。默认配置文件已放入随时可访问的目录中，存为 **/etc/ssh/sshd_config.d/00-ansible_system_role.conf**。

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** ：
   - 对于运行 RHEL 8 或更早版本的受管节点：

```
---
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure SSHD to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_namespace: <my-application>
```

```
    sshd:
      # Environment variables to accept
      AcceptEnv:
        LANG
        LS_COLORS
        EDITOR
```

- 对于运行 RHEL 9 或更高版本的受管节点：

```
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

在 **sshd_config_file** 变量中，定义 **sshd** 系统角色在其中写入配置选项的 **.conf** 文件。使用两位前缀，例如 ***42-*** 来指定应用配置文件的顺序。

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

- 验证 SSH 服务器上的配置：
  - 对于运行 RHEL 8 或更早版本的受管节点：

```
# cat /etc/ssh/sshd_config.d/42-my-application.conf
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

  - 对于运行 RHEL 9 或更高版本的受管节点：

```
# cat /etc/ssh/sshd_config
...
# BEGIN sshd system role managed block: namespace <my-application>
```

```
Match all
  AcceptEnv LANG LS_COLORS EDITOR
# END sshd system role managed block: namespace <my-application>
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** directory

1.10. 其他资源

- **sshd(8)**、**ssh(1)**、**scp(1)**、**sftp(1)**、**ssh-keygen(1)**、**ssh-copy-id(1)**、**ssh_config(5)**、**ssh_config(5)**、**update-crypto-policies(8)** 和 **crypto-policies(7)** 手册页
- 为使用非标准配置的应用程序和服务配置 SELinux
- 使用 firewalld 控制网络流量

第 2 章 创建并管理 TLS 密钥和证书

您可以使用 TLS （传输层安全）协议加密在两个系统间传输的通信。此标准将非对称加密与私钥和公钥、数字签名和证书一起使用。

2.1. TLS 证书

TLS （传输层安全）是一种协议，它允许客户端-服务器应用程序安全地传递信息。TLS 使用公钥和私钥对系统来加密在客户端和服务器间传输的通信。TLS 是到 SSL （安全套接字层）的后继协议。

TLS 使用 X.509 证书将主机名或机构等身份绑定到使用数字签名的公共密钥。X.509 是一个定义公钥证书格式的标准。

安全应用程序的身份验证取决于应用证书中公钥值的完整性。如果攻击者用自己的公钥替换了公钥，则它可以模拟真正的应用程序，并获得对安全数据的访问权限。为防止此类攻击，所有证书都必须由证书颁发机构(CA)签名。CA 是一个可信节点，用于确认证书中公钥值的完整性。

CA 通过添加公钥签名并发布证书来签署公钥。数字签名是用 CA 的私钥编码的一条消息。通过分发 CA 的证书，CA 的公钥可供应用程序使用。应用程序使用 CA 的公钥解码 CA 的数字签名来验证证书是否为有效签名。

要让证书由 CA 签名，您必须生成一个公钥，并将其发送给 CA 进行签名。这称为证书签名请求(CSR)。CSR 也包含证书的可分辨名称(DN)。您可以为任一证书提供的 DN 信息可以包括您所在国家的两字母国家代码、州或省、城市或乡镇的全名，您机构的名称、电子邮件地址，也可以为空。许多当前商业 CA 首选 Subject Alternative Name 扩展，并在 CSR 中忽略 DN。

RHEL 为使用 TLS 证书提供了两个主要工具包：GnuTLS 和 OpenSSL.您可以使用 **openssl** 软件包中的 **openssl** 工具创建、读取、签名和验证证书。**gnutls-utils** 软件包提供的 **certtool** 工具可以使用不同的语法以及后端中的所有不同库的集合执行相同的操作。

其他资源

- RFC 5280：Internet X.509 公钥基础设施证书和证书撤销列表(CRL)配置文件
- **openssl (1)**, **x509 (1)**, **ca (1)**, **req (1)** 和 **certtool (1)** 手册页

2.2. 使用 OPENSSL 创建私有 CA

当您的情况需要在您的内部网络内验证实体时，私有证书颁发机构(CA)非常有用。例如，当使用基于您控制下的 CA 签名的证书的身份验证创建 VPN 网关时，或者您不想支付商业 CA 时，请使用私有 CA。要在这样的用例中签名证书，私有 CA 使用自签名证书。

先决条件

- 您有 **root** 特权或权限来使用 **sudo** 输入管理命令的命令。需要此类权限的命令标有 **#**。

流程

1. 为您的 CA 生成私钥。例如，以下命令会创建一个 256 位 Elliptic Curve Digital Signature Algorithm(ECDSA)密钥：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <ca.key>
```

密钥生成过程的时间取决于主机的硬件和熵、所选算法以及密钥长度。

2. 使用上一个命令生成的私钥创建证书：

```
$ openssl req -key <ca.key> -new -x509 -days 3650 -addext
keyUsage=critical,keyCertSign,cRLSign -subj "/CN=<Example CA>" -out <ca.crt>
```

生成的 **ca.crt** 文件是一个自签名 CA 证书，可用于为其他证书签名 10 年。对于私有 CA，您可以将 *<Example CA>* 替换为作为通用名称(CN)的任何字符串。

3. 对 CA 的私钥设置安全权限，例如：

```
# chown <root>:<root> <ca.key>
# chmod 600 <ca.key>
```

后续步骤

- 要将自签名 CA 证书用作客户端系统上的信任锚，请将 CA 证书复制到客户端，并以 **root** 用户身份将其添加到客户端的系统范围的信任存储中：

```
# trust anchor <ca.crt>
```

如需更多信息，请参阅 第 3 章 *使用共享的系统证书*。

验证

1. 创建证书签名请求(CSR)，并使用您的 CA 为请求签名。CA 必须成功创建一个基于 CSR 的证书，例如：

```
$ openssl x509 -req -in <client-cert.csr> -CA <ca.crt> -CAkey <ca.key> -CAcreateserial -
days 365 -extfile <openssl.cnf> -extensions <client-cert> -out <client-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书” 。

2. 显示有关自签名 CA 的基本信息：

```
$ openssl x509 -in <ca.crt> -text -noout
Certificate:
…
        X509v3 extensions:
            …
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
…
```

3. 验证私钥的一致性：

```
$ openssl pkey -check -in <ca.key>
Key is valid
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgcagSaTEBn74xZAwO
```

```
18wRpXoCVC9vcPki7WlT+gnmCI+hRANCAARb9NxIvkaVjFhOoZbGp/HtIQxbM78E
lwbDP0BI624xBJ8gK68ogSaq2x4SdezFdV1gNeKScDcU+Pj2pELldmdF
-----END PRIVATE KEY-----
```

其他资源

- **openssl (1)**, **ca (1)**, **genpkey (1)**, **x509 (1)** 和 **req (1)** 手册页

2.3. 使用 OPENSSL 为 TLS 服务器证书创建私钥和 CSR

您只有有了来自证书颁发机构(CA)的有效 TLS 证书时才可以使用 TLS 加密的通信频道。要获取证书，您必须首先为您的服务器创建私钥和证书签名请求(CSR)。

流程

1. 在服务器系统上生成私钥，例如：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <server-private.key>
```

2. 可选：使用您选择的文本编辑器来准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_server.cnf>
[server-cert]
keyUsage = critical, digitalSignature, keyEncipherment, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
C = <US>
O = <Example Organization>
CN = <server.example.com>

[alt_name]
DNS.1 = <example.com>
DNS.2 = <server.example.com>
IP.1 = <192.168.0.1>
IP.2 = <::1>
IP.3 = <127.0.0.1>
```

extendedKeyUsage = serverAuth 选项限制证书的使用。

3. 使用之前创建的私钥创建 CSR：

```
$ openssl req -key <server-private.key> -config <example_server.cnf> -new -out <server-cert.csr>
```

如果省略了 **-config** 选项，**req** 工具会提示您额外的信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: <US>
State or Province Name (full name) []: <Washington>
Locality Name (eg, city) [Default City]: <Seattle>
Organization Name (eg, company) [Default Company Ltd]: <Example Organization>
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: <server.example.com>
Email Address []: <server@example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 进行签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书”。

验证

1. 从 CA 获取请求的证书后，检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ openssl x509 -text -noout -in <server-cert.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

其他资源

- **openssl (1)**, **x509 (1)**, **genpkey (1)**, **req (1)** 和 **config (5)** 手册页

2.4. 使用 OPENSSL 为 TLS 客户端证书创建私钥和 CSR

您只有有了来自证书颁发机构(CA)的有效 TLS 证书时才可以使用 TLS 加密的通信频道。要获取证书，您必须首先为您的客户端创建私钥和证书签名请求(CSR)。

流程

1. 在客户端系统上生成私钥，例如：

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <client-private.key>
```

2. 可选：使用您选择的文本编辑器来准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_client.cnf>
[client-cert]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
CN = <client.example.com>

[clnt_alt_name]
email= <client@example.com>
```

extendedKeyUsage = clientAuth 选项限制证书的使用。

3. 使用之前创建的私钥创建 CSR：

```
$ openssl req -key <client-private.key> -config <example_client.cnf> -new -out <client-cert.csr>
```

如果省略了 **-config** 选项，**req** 工具会提示您额外的信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
…
Common Name (eg, your name or your server's hostname) []: <client.example.com>
Email Address []: <client@example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 进行签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。如需更多信息，请参阅 第 2.5 节 “使用私有 CA 使用 OpenSSL 为 CSR 发布证书”。

验证

1. 检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ openssl x509 -text -noout -in <client-cert.crt>
Certificate:
…
            X509v3 Extended Key Usage:
```

```
            TLS Web Client Authentication
          X509v3 Subject Alternative Name:
            email:client@example.com
…
```

其他资源

- **openssl (1)**, **x509 (1)**, **genpkey (1)**, **req (1)** 和 **config (5)** 手册页

2.5. 使用私有 CA 使用 OPENSSL 为 CSR 发布证书

要让系统建立一条 TLS 加密的通信频道，证书颁发机构(CA)必须为它们提供有效的证书。如果您有私有 CA，您可以通过从系统签署证书签名请求(CSR)来创建请求的证书。

先决条件

- 您已配置了私有 CA。如需更多信息，请参阅 第 2.2 节 “使用 OpenSSL 创建私有 CA”。
- 您有一个包含 CSR 的文件。您可以在 第 2.3 节 “使用 OpenSSL 为 TLS 服务器证书创建私钥和 CSR” 中找到创建 CSR 的示例。

流程

1. 可选：使用您选择的文本编辑器准备一个 OpenSSL 配置文件，以便为证书添加扩展，例如：

```
$ vim <openssl.cnf>
[server-cert]
extendedKeyUsage = serverAuth

[client-cert]
extendedKeyUsage = clientAuth
```

2. 使用 **x509** 工具创建基于 CSR 的证书，例如：

```
$ openssl x509 -req -in <server-cert.csr> -CA <ca.crt> -CAkey <ca.key> -days 365 -extfile
<openssl.cnf> -extensions <server-cert> -out <server-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

其他资源

- **openssl (1)**, **ca (1)**, 和 **x509 (1)** 手册页

2.6. 使用 GNUTLS 创建私有 CA

当您的情况需要在您的内部网络内验证实体时，私有证书颁发机构(CA)非常有用。例如，当使用基于您控制下的 CA 签名的证书的身份验证创建 VPN 网关时，或者您不想支付商业 CA 时，请使用私有 CA。要在这样的用例中签名证书，私有 CA 使用自签名证书。

先决条件

- 您有 **root** 特权或权限来使用 **sudo** 输入管理命令的命令。需要此类权限的命令标有 **#**。

- 您已在系统上安装了 GnuTLS。如果没有，您可以使用这个命令：

```
$ dnf install gnutls-utils
```

流程

1. 为您的 CA 生成私钥。例如，以下命令会创建一个 256 位 ECDSA (Elliptic Curve Digital Signature Algorithm)密钥：

```
$ certtool --generate-privkey --sec-param High --key-type=ecdsa --outfile <ca.key>
```

密钥生成过程的时间取决于主机的硬件和熵、所选算法以及密钥长度。

2. 为证书创建一个模板文件。

   a. 使用您选择的文本编辑器创建一个文件，例如：

```
$ vi <ca.cfg>
```

   b. 编辑该文件以包含必要的认证详情：

```
organization = "Example Inc."
state = "Example"
country = EX
cn = "Example CA"
serial = 007
expiration_days = 365
ca
cert_signing_key
crl_signing_key
```

3. 使用在第 1 步中生成的私钥创建一个签名证书：
   生成的 *<ca.crt>* 文件是一个自签名 CA 证书，您可用来为其他证书签名一年。*<ca.crt>* 文件是公钥(证书)。加载的文件 *<ca.key>* 是私钥。您应该将此文件保存在安全的地方。

```
$ certtool --generate-self-signed --load-privkey <ca.key> --template <ca.cfg> --outfile
<ca.crt>
```

4. 对 CA 的私钥设置安全权限，例如：

```
# chown <root>:<root> <ca.key>
# chmod 600 <ca.key>
```

后续步骤

- 要将自签名 CA 证书用作客户端系统上的信任锚，请将 CA 证书复制到客户端，并以 **root** 用户身份将其添加到客户端的系统范围的信任存储中：

```
# trust anchor <ca.crt>
```

如需更多信息，请参阅 第 3 章 *使用共享的系统证书*。

验证

1. 显示有关自签名 CA 的基本信息：

```
$ certtool --certificate-info --infile <ca.crt>
Certificate:
…
    X509v3 extensions:
        …
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Key Usage: critical
            Certificate Sign, CRL Sign
```

2. 创建证书签名请求(CSR)，并使用您的 CA 为请求签名。CA 必须成功创建一个基于 CSR 的证书，例如：

   a. 为您的 CA 生成私钥：

   ```
   $ certtool --generate-privkey --outfile <example-server.key>
   ```

   b. 在您选择的文本编辑器中打开一个新配置文件，例如：

   ```
   $ vi <example-server.cfg>
   ```

   c. 编辑该文件以包含必要的认证详情：

   ```
   signing_key
   encryption_key
   key_agreement

   tls_www_server

   country = "US"
   organization = "Example Organization"
   cn = "server.example.com"

   dns_name = "example.com"
   dns_name = "server.example.com"
   ip_address = "192.168.0.1"
   ip_address = "::1"
   ip_address = "127.0.0.1"
   ```

   d. 使用之前创建的私钥生成一个请求：

   ```
   $ certtool --generate-request --load-privkey <example-server.key> --template <example-server.cfg> --outfile <example-server.crq>
   ```

   e. 生成证书并使用 CA 的私钥对其签名：

   ```
   $ certtool --generate-certificate --load-request <example-server.crq> --load-ca-certificate <ca.crt> --load-ca-privkey <ca.key> --outfile <example-server.crt>
   ```

   其他资源

- **certtool (1)** 和 **trust (1)** 手册页

2.7. 使用 GNUTLS 为 TLS 服务器证书创建私钥和 CSR

要获取证书，您必须首先为您的服务器创建私钥和证书签名请求(CSR)。

流程

1. 在服务器系统上生成私钥，例如：

```
$ certtool --generate-privkey --sec-param High --outfile <example-server.key>
```

2. 可选：使用您选择的文本编辑器来准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_server.cnf>
signing_key
encryption_key
key_agreement

tls_www_server

country = "US"
organization = "Example Organization"
cn = "server.example.com"

dns_name = "example.com"
dns_name = "server.example.com"
ip_address = "192.168.0.1"
ip_address = "::1"
ip_address = "127.0.0.1"
```

3. 使用之前创建的私钥创建 CSR：

```
$ certtool --generate-request --template <example-server.cfg> --load-privkey <example-server.key> --outfile <example-server.crq>
```

如果省略 **--template** 选项，**certool** 工具会提示您输入额外信息，例如：

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 进行签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。请参阅 第 2.9 节 “使用私有 CA 为 带有 GnuTLS 的 CSR 发布证书” 了解更多信息。

验证

1. 从 CA 获取请求的证书后，检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ certtool --certificate-info --infile <example-server.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

其他资源

- **certtool (1)** 手册页

2.8. 使用 GNUTLS 为 TLS 客户端证书创建私钥和 CSR

要获取证书，您必须首先为您的客户端创建私钥和证书签名请求(CSR)。

流程

1. 在客户端系统上生成私钥，例如：

```
$ certtool --generate-privkey --sec-param High --outfile <example-client.key>
```

2. 可选：使用您选择的文本编辑器来准备一个简化创建 CSR 的配置文件，例如：

```
$ vim <example_client.cnf>
signing_key
encryption_key

tls_www_client
```

```
cn = "client.example.com"
email = "client@example.com"
```

3. 使用之前创建的私钥创建 CSR：

```
$ certtool --generate-request --template <example-client.cfg> --load-privkey <example-client.key> --outfile <example-client.crq>
```

如果省略 **--template** 选项， **certtool** 工具会提示您输入额外信息，例如：

```
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

后续步骤

- 将 CSR 提交给您选择的 CA 进行签名。或者，对于可信网络中的内部使用场景，请使用您的私有 CA 进行签名。请参阅 第 2.9 节 “使用私有 CA 为 带有 GnuTLS 的 CSR 发布证书” 了解更多信息。

验证

1. 检查证书的人类可读部分是否与您的要求匹配，例如：

```
$ certtool --certificate-info --infile <example-client.crt>
Certificate:
…
        X509v3 Extended Key Usage:
            TLS Web Client Authentication
        X509v3 Subject Alternative Name:
            email:client@example.com
…
```

其他资源

- **certtool (1)** 手册页

2.9. 使用私有 CA 为 带有 GNUTLS 的 CSR 发布证书

要让系统建立一条 TLS 加密的通信频道，证书颁发机构(CA)必须为它们提供有效的证书。如果您有私有 CA，您可以通过从系统签署证书签名请求(CSR)来创建请求的证书。

先决条件

- 您已配置了私有 CA。请参阅 第 2.6 节 “使用 GnuTLS 创建私有 CA” 了解更多信息。
- 您有一个包含 CSR 的文件。您可以在 第 2.7 节 “使用 GnuTLS 为 TLS 服务器证书创建私钥和 CSR” 中找到创建 CSR 的示例。

流程

1. 可选：使用您选择的文本编辑器准备一个 GnuTLS 配置文件，以便向证书添加扩展，例如：

```
$ vi <server-extensions.cfg>
honor_crq_extensions
ocsp_uri = "http://ocsp.example.com"
```

2. 使用 **certtool** 工具创建一个基于 CSR 的证书，例如：

```
$ certtool --generate-certificate --load-request <example-server.crq> --load-ca-privkey
<ca.key> --load-ca-certificate <ca.crt> --template <server-extensions.cfg> --outfile
<example-server.crt>
```

其他资源

- **certtool (1)** 手册页

第 3 章 使用共享的系统证书

共享的系统证书存储使 NSS、GnuTLS、OpenSSL 和 Java 能够共享用于检索系统证书锚和块列表信息的默认源。默认情况下，信任存储包含 Mozilla CA 列表，包括正和负信任。系统允许更新核心 Mozilla CA 列表或选择其他证书列表。

3.1. 系统范围的信任存储

在 RHEL 中，整合的系统范围的信任存储位于 **/etc/pki/ca-trust/** 和 **/usr/share/pki/ca-trust-source/** 目录中。对 **/usr/share/pki/ca-trust-source/** 中信任设置的优先级的处理低于 **/etc/pki/ca-trust/** 中的设置。

证书文件根据它们所安装到的子目录处理：

- 信任锚属于
  - **/usr/share/pki/ca-trust-source/anchors/** 或
  - **/etc/pki/ca-trust/source/anchors/**。
- 不信任的证书存储在
  - **/usr/share/pki/ca-trust-source/blocklist/** 或
  - **/etc/pki/ca-trust/source/blocklist/**。
- 扩展的 BEGIN TRUSTED 文件格式的证书位于
  - **/usr/share/pki/ca-trust-source/** 或
  - **/etc/pki/ca-trust/source/**。

注意

在分层加密系统中，信任锚是其他各方认为值得信任的权威实体。在 X.509 架构中，根证书是从中派生信任链的信任锚。要启用链验证，信任方必须首先能够访问信任锚。

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

3.2. 添加新证书

要使用新的信任来源确认系统上的应用程序，请将相应的证书添加到系统范围的存储中，并使用 **update-ca-trust** 命令。

先决条件

- **ca-certificates** 软件包存在于系统中。

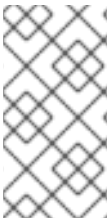
流程

1. 要在简单的 PEM 或 DER 文件格式中添加证书到系统中信任的 CA 列表中,请将证书文件复制到 **/usr/share/pki/ca-trust-source/anchors/** 或 **/etc/pki/ca-trust/source/anchors/** 目录中，例如：

```
# cp ~/certificate-trust-examples/Cert-trust-test-ca.pem /usr/share/pki/ca-trust-source/anchors/
```

2. 要更新系统范围的信任存储配置，请使用 **update-ca-trust** 命令：

```
# update-ca-trust
```



注意

虽然 Firefox 浏览器可以在不预先执行 **update-ca-trust** 的情况下使用一个添加的证书，但在每次 CA 更改后需要输入 **update-ca-trust** 命令。另请注意，浏览器（如 Firefox、Chromium 和 GNOME Web 缓存文件）,您可能需要清除浏览器的缓存或重新启动浏览器来加载当前的系统证书配置。

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

3.3. 管理信任的系统证书

trust 命令提供了一种便捷的方式，来管理共享的系统范围信任存储中的证书。

- 要列出、提取、添加、删除或修改信任锚，请使用 **trust** 命令。要查看这个命令的内置帮助信息，请不要输入任何参数，或使用 **--help** 指令：

```
$ trust
usage: trust command <args>...

Common trust commands are:
  list            List trust or certificates
  extract         Extract certificates and trust
  extract-compat  Extract trust compatibility bundles
  anchor          Add, remove, change trust anchors
  dump            Dump trust objects in internal format

See 'trust <command> --help' for more information
```

- 要列出所有系统信任锚和证书，请使用 **trust list** 命令：

```
$ trust list
pkcs11:id=%d2%87%b4%e3%df%37%27%93%55%f6%56%ea%81%e5%36%cc%8c%1e%3f%bd;type=cert
    type: certificate
    label: ACCVRAIZ1
    trust: anchor
    category: authority

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%ac%50;type=cert
    type: certificate
    label: ACEDICOM Root
```

```
    trust: anchor
    category: authority
...
```

- 要将信任锚存储在系统范围的信任存储中，请使用 **trust anchor** 子命令，并指定证书的路径。将 *<path.to/certificate.crt>* 替换为证书的路径及其文件名：

```
# trust anchor <path.to/certificate.crt>
```

- 要删除证书，请使用证书的路径或证书的 ID：

```
# trust anchor --remove <path.to/certificate.crt>
# trust anchor --remove "pkcs11:id=<%AA%BB%CC%DD%EE>;type=cert"
```

其他资源

- **trust** 命令的所有子命令都提供了详细的内置帮助，例如。

```
$ trust list --help
usage: trust list --filter=<what>

  --filter=<what>     filter of what to export
                      ca-anchors        certificate anchors
...
  --purpose=<usage>   limit to certificates usable for the purpose
                      server-auth       for authenticating servers
...
```

其他资源

- **update-ca-trust(8)** 和 **trust(1)** 手册页

第 4 章 计划并使用 TLS

TLS（传输层安全）是用来保护网络通信的加密协议。在通过配置首选密钥交换协议、身份验证方法和加密算法来强化系统安全设置时，需要记住支持的客户端的范围越广，产生的安全性就越低。相反，严格的安全设置会导致与客户端的兼容性受限，这可能导致某些用户被锁定在系统之外。请确保以最严格的可用配置为目标，并且仅在出于兼容性原因需要时才放宽配置。

4.1. SSL 和 TLS 协议

安全套接字层(SSL)协议最初使由 Netscape 公司开发的，以提供一种在互联网上进行安全通信的机制。因此，该协议被互联网工程任务组(IETF)采纳，并重命名为传输层安全(TLS)。

TLS 协议位于应用协议层和可靠的传输层之间，例如 TCP/IP。它独立于应用程序协议，因此可在很多不同的协议下分层，例如：HTTP、FTP、SMTP 等。

协议版本	用法建议
SSL v2	不要使用。具有严重的安全漏洞。从 RHEL 7 开始从核心加密库中删除了。
SSL v3	不要使用。具有严重的安全漏洞。从 RHEL 8 开始从核心加密库中删除了。
TLS 1.0	不建议使用。已知的无法以保证互操作性方式缓解的问题，且不支持现代密码套件。在 RHEL 9 中，在所有加密策略中禁用。
TLS 1.1	在需要时用于互操作性.不支持现代加密套件。在 RHEL 9 中，在所有加密策略中禁用。
TLS 1.2	支持现代 AEAD 密码组合。此版本在所有系统范围的加密策略中启用，但此协议的可选部分包含漏洞，TLS 1.2 也允许过时的算法。
TLS 1.3	推荐的版本。TLS 1.3 删除了已知有问题的选项，通过加密更多协商握手来提供额外的隐私，由于使用了更有效的现代加密算法，所以可以更快。在所有系统范围的加密策略中也启用了 TLS 1.3。

其他资源

- IETF：传输层安全性(TLS)协议版本 1.3。

4.2. RHEL 9 中 TLS 的安全注意事项

在 RHEL 9 中，TLS 配置是使用系统范围的加密策略机制执行的。不再支持 1.2 以下的 TLS 版本。**DEFAULT**、**FUTURE** 和 **LEGACY** 加密策略只允许 TLS 1.2 和 1.3。如需更多信息，请参阅 使用系统范围的加密策略。

RHEL 9 中包含的库所提供的默认设置对于大多数部署来说已经足够安全了。TLS 实现尽可能使用安全算法，而不阻止来自或到旧客户端或服务器的连接。在具有严格安全要求的环境中应用强化设置，在这些环境中，不支持安全算法或协议的旧客户端或服务器不应连接或不允许连接。

强化 TLS 配置的最简单方法是使用 **update-crypto-policies --set FUTURE** 命令将系统范围的加密策略级别切换到 **FUTURE** 。

警告

为 **LEGACY** 加密策略禁用的算法不符合红帽的 RHEL 9 安全愿景，其安全属性不可靠。考虑放弃使用这些算法，而不是重新启用它们。如果您确实决定重新启用它们（例如，为了与旧硬件的互操作性），请将它们视为不安全的，并应用额外的保护措施，例如将其网络交互隔离到单独的网络段。不要在公共网络中使用它们。

如果您决定不遵循 RHEL 系统范围的加密策略，或根据您的设置创建自定义的加密策略，请在自定义配置中对首选协议、密码套件和密钥长度使用以下建议：

4.2.1. 协议

TLS 的最新版本提供了最佳安全机制。TLS 1.2 现在是最低版本，即使使用 **LEGACY** 加密策略也是如此。通过选择不使用加密策略或提供自定义策略，可以重新启用旧协议版本，但不支持生成的配置。

请注意，尽管 RHEL 9 支持 TLS 版本 1.3，但 RHEL 9 组件并不完全支持这个协议的所有功能。例如，Apache Web 服务器尚不完全支持可降低连接延迟的 0-RTT(Zero R Trip Time)功能。

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"

4.2.2. 密码套件

现代、更安全的密码套件应该优先于旧的不安全密码套件。一直禁止 eNULL 和 aNULL 密码套件的使用，它们根本不提供任何加密或身份验证。如果有可能，基于 RC4 或 HMAC-MD5 的密码套件也必须被禁用。这同样适用于所谓的出口密码套件，它们被有意地弱化了，因此很容易被破解。

虽然不会立即变得不安全，但提供安全性少于 128 位的密码套件在它们的短使用期中不应该被考虑。使用 128 位或者更高安全性的算法可以预期在至少数年内不会被破坏，因此我们强烈推荐您使用此算法。请注意，虽然 3DES 密码公告使用 168 位但它们实际只提供了 112 位的安全性。

始终优先使用支持(完美)转发保密(PFS)的密码套件，这样可确保加密数据的机密性，以防服务器密钥被泄露。此规则排除了快速 RSA 密钥交换，但允许使用 ECDHE 和 DHE。在两者中，ECDHE 更快，因此是首选。

您还应该优先选择 AEAD 密码，如 AES-GCM，使用 CBC 模式密码，因为它们不容易受到 padding oracle 攻击的影响。此外，在很多情况下，在 CBC 模式下，AES-GCM 比 AES 快，特别是当硬件具有 AES 加密加速器时。

另请注意，在使用带有 ECDSA 证书的 ECDHE 密钥交换时，事务的速度甚至比纯 RSA 密钥交换要快。为了给旧客户端提供支持，您可以在服务器上安装两对证书和密钥：一对带有 ECDSA 密钥（用于新客户端），另一对带有 RSA 密钥（用于旧密钥）。

4.2.3. 公钥长度

在使用 RSA 密钥时，总是首选使用至少由 SHA-256 签名的 3072 位的密钥长度，对于真实的 128 位安全性来说，这个值已经足够大。

警告

您的系统安全性仅与链中最弱的连接相同。例如，只是一个强大的密码不能保证良好安全性。密钥和证书以及认证机构(CA)用来签署您的密钥的哈希功能和密钥同样重要。

4.3. 在应用程序中强化 TLS 配置

在 RHEL 中，系统范围的加密策略 提供了一种便捷的方法，来确保您的使用加密库的应用程序不允许已知的不安全协议、密码或算法。

如果要使用自定义加密设置来强化与 TLS 相关的配置，您可以使用本节中描述的加密配置选项，并以最少的需求量覆盖系统范围的加密策略。

无论您选择使用什么配置，请始终确保您的服务器应用程序强制实施 *服务器端密码顺序*，以便使用的密码套件由您配置的顺序来决定。

4.3.1. 将 Apache HTTP 服务器配置为使用 TLS

Apache HTTP 服务器 可以使用 **OpenSSL** 和 **NSS** 库来满足其 TLS 的需求。RHEL 9 通过 eponymous 软件包提供 **mod_ssl** 功能：

```
# dnf install mod_ssl
```

mod_ssl 软件包将安装 **/etc/httpd/conf.d/ssl.conf** 配置文件，该文件可用来修改 **Apache HTTP 服务器** 与 TLS 相关的设置。

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** 文件中详细介绍。**/usr/share/httpd/manual/ssl/ssl_howto.html** 文件中描述了各种设置的示例。

修改 **/etc/httpd/conf.d/ssl.conf** 配置文件中的设置时，请确保至少考虑以下三个指令：

SSLProtocol

使用这个指令指定您要允许的 TLS 或者 SSL 版本。

SSLCipherSuite

使用这个指令来指定您首选的密码套件或禁用您要禁止的密码套件。

SSLHonorCipherOrder

取消注释并将此指令设置为 **on**，以确保连接的客户端遵循您指定的密码顺序。

例如，只使用 TLS 1.2 和 1.3 协议：

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
```

如需更多信息，请参阅 部署 Web 服务器和反向代理 中的 在 Apache HTTP 服务器上配置 TLS 加密 一章。

4.3.2. 将 Nginx HTTP 和代理服务器配置为使用 TLS

要在 **Nginx** 中启用 TLS 1.3 支持，请将 **TLSv1.3** 值添加到 **/etc/nginx/nginx.conf** 配置文件的 **server** 部分的 **ssl_protocols** 选项：

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ....
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers
    ....
}
```

如需更多信息，请参阅 部署 web 服务器和反向代理 文档中的 向 Nginx web 服务器添加 TLS 加密 一章。

4.3.3. 将 Dovecot 邮件服务器配置为使用 TLS

要将 **Dovecot** 邮件服务器的安装配置为使用 TLS，请修改 **/etc/dovecot/conf.d/10-ssl.conf** 配置文件。您可以在 **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt** 文件中找到其提供的一些基本配置指令的说明，该文件与 **Dovecot** 的标准安装一起安装。

修改 **/etc/dovecot/conf.d/10-ssl.conf** 配置文件中的设置时，请确保至少考虑以下三个指令：

ssl_protocols
: 使用这个指令指定您要允许或者禁用的 TLS 或者 SSL 版本。

ssl_cipher_list
: 使用这个指令指定您首选的密码套件或禁用您要禁止的密码套件。

ssl_prefer_server_ciphers
: 取消注释并将此指令设置为 **yes**，以确保连接的客户端遵循您指定的密码顺序。

例如，**/etc/dovecot/conf.d/10-ssl.conf** 中的以下行只允许 TLS 1.1 及之后的版本：

```
ssl_protocols = !SSLv2 !SSLv3 !TLSv1
```

其他资源

- 部署 Web 服务器和反向代理
- **config(5)** 和 **ciphers(1)** 手册页。
- 安全使用传输层安全性(TLS)和数据报传输层安全性(DTLS)的建议。
- Mozilla SSL 配置生成器。

- SSL 服务器测试 。

第 5 章 使用 IPSEC 配置 VPN

在 RHEL 9 中，您可以使用 **IPsec** 协议配置虚拟专用网络(VPN)，Libreswan 应用程序支持该协议。

5.1. LIBRESWAN 作为 IPSEC VPN 的实现

在 RHEL 中，您可以使用 IPsec 协议配置虚拟专用网络(VPN)，Libreswan 应用程序支持该协议。Libreswan 是 Openswan 应用程序的延续，Openswan 文档中的许多示例可以通过 Libreswan 交换。

VPN 的 IPsec 协议使用互联网密钥交换(IKE)协议进行配置。术语 IPsec 和 IKE 可互换使用。IPsec VPN 也称为 IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN 或 IKE/IPsec VPN。IPsec VPN 变体，它使用 Level 2 Tunneling Protocol(L2TP)，被称为 L2TP/IPsec VPN，它需要 **optional** 软件仓库提供的 **xl2tpd** 软件包。

libreswan 是一个开源用户空间 IKE 实现。IKE v1 和 v2 作为用户级别的守护进程实现。IKE 协议也加密。IPsec 协议由 Linux 内核实现，Libreswan 配置内核以添加和删除 VPN 隧道配置。

IKE 协议使用 UDP 端口 500 和 4500。IPsec 协议由两个协议组成：

- 封装安全性 Payload(ESP)，其协议号为 50。
- 经过身份验证的标头(AH)，其协议号为 51。

不建议使用 AH 协议。建议将 AH 用户迁移到使用 null 加密的 ESP。

IPsec 协议提供两种操作模式：

- 隧道模式（默认）
- 传输模式。

您可以用没有 IKE 的 IPsec 来配置内核。这称为 *手动密钥环*。您还可以使用 **ip xfrm** 命令来配置手动密钥，但为了安全起见，强烈建议您不要这样做。Libreswan 使用 Netlink 接口与 Linux 内核进行通信。内核执行数据包加密和解密。

Libreswan 使用网络安全服务 (NSS) 加密库。NSS 已获得 *联邦信息处理标准(FIPS*)出版物 140-2 的使用认证。

重要

IKE/IPsec VPN（由 Libreswan 和 Linux 内核实现）是 RHEL 中推荐的唯一 VPN 技术。在不了解这样做风险的情况下不要使用任何其他 VPN 技术。

在 RHEL 中，Libreswan 默认**遵循系统范围的加密策略**。这样可确保 Libreswan 将当前威胁模型包括（IKEv2）的安全设置用作默认协议。如需更多信息，请参阅 使用系统范围的加密策略。

Libreswan 没有使用术语"源（source）"和"目的地（destination）"或"服务器（server）"和"客户端（client）"，因为 IKE/IPsec 使用对等（peer to peer）协议。相反，它使用术语"左"和"右"来指端点（主机）。这也允许您在大多数情况下在两个端点使用相同的配置。但是，管理员通常选择始终对本地主机使用"左"，对远程主机使用"右"。

leftid 和 **rightid** 选项充当身份验证过程中相应主机的标识。详情请查看 **ipsec.conf(5)** 手册页。

5.2. LIBRESWAN 中的身份验证方法

Libreswan 支持多种身份验证方法，每种方法适合不同的场景。

预共享密钥(PSK)

预共享密钥(PSK)是最简单的身份验证方法。出于安全考虑，请勿使用小于 64 个随机字符的 PSK。在 FIPS 模式中，PSK 必须符合最低强度要求，具体取决于所使用的完整性算法。您可以使用 **authby=secret** 连接来设置 PSK。

原始 RSA 密钥

原始 RSA 密钥 通常用于静态主机到主机或子网到子网 IPsec 配置。每个主机都使用所有其他主机的公共 RSA 密钥手动配置，Libreswan 在每对主机之间建立 IPsec 隧道。对于大量主机，这个方法不能很好地扩展。

您可以使用 **ipsec newhostkey** 命令在主机上生成原始 RSA 密钥。您可以使用 **ipsec showhostkey** 命令列出生成的密钥。使用 CKA ID 密钥的连接配置需要 **leftrsasigkey=** 行。原始 RSA 密钥使用 **authby=rsasig** 连接选项。

X.509 证书

X.509 证书 通常用于大规模部署连接到通用 IPsec 网关的主机。中心 *证书颁发机构*(CA)为主机或用户签署 RSA 证书。此中心 CA 负责中继信任，包括单个主机或用户的撤销。

例如，您可以使用 **openssl** 命令和 NSS **certutil** 命令来生成 X.509 证书。因为 Libreswan 使用 **leftcert=** 配置选项中证书的昵称从 NSS 数据库读取用户证书，所以在创建证书时请提供昵称。

如果使用自定义 CA 证书，则必须将其导入到网络安全服务(NSS)数据库中。您可以使用 **ipsec import** 命令将 PKCS #12 格式的任何证书导入到 Libreswan NSS 数据库。

警告

Libreswan 需要互联网密钥交换(IKE)对等 ID 作为每个对等证书的主题替代名称(SAN)，如 RFC 4945 的 3.1 章节 所述。通过更改 **require-id-on-certificated=** 选项禁用此检查可能会导致系统容易受到中间人攻击。

使用 **authby=rsasig** 连接选项，根据使用带 SHA-2 的 RSA 的 X.509 证书进行身份验证。您可以进一步对它进行限制，对于 ECDSA 数字签名使用 SHA-2（将 **authby=** 设置为 **ecdsa**），以及基于 RSA Probabilistic Signature Scheme (RSASSA-PSS) 数据签名的验证使用 SHA-2（**authby=rsa-sha2**）。默认值为 **authby=rsasig,ecdsa**。

证书和 **authby=** 签名方法应匹配。这提高了互操作性，并在一个数字签名系统中保留身份验证。

NULL 身份验证

NULL 身份验证 用来在没有身份验证的情况下获得网状加密。它可防止被动攻击，但不能防止主动攻击。但是，因为 IKEv2 允许非对称身份验证方法，因此 NULL 身份验证也可用于互联网规模的机会主义 IPsec。在此模型中，客户端对服务器进行身份验证，但服务器不对客户端进行身份验证。此模型类似于使用 TLS 的安全网站。使用 **authby=null** 进行 NULL 身份验证。

保护量子计算机

除了上述身份验证方法外，您还可以使用 *Post-quantum Pre-shared Key* (PPK)方法来防止量子计算机可能的攻击。单个客户端或客户端组可以通过指定与带外配置的预共享密钥对应的 PPK ID 来使用它们自己的 PPK。

使用带有预共享密钥的 IKEv1 防止量子攻击者。重新设计 IKEv2 不会原生提供这种保护。Libreswan 提供使用 *Post-quantum Pre-shared Key* (PPK)来保护 IKEv2 连接免受量子攻击。

要启用可选的 PPK 支持，请在连接定义中添加 **ppk=yes**。如需要 PPK，请添加 **ppk=insist**。然后，可以给每个客户端一个带有 secret 值的 PPK ID，该 secret 值在带外进行通信（最好是量子安全的）。PPK 应该具有很强的随机性，而不是基于字典中的单词。PPK ID 和 PPK 数据保存在 **ipsec.secrets** 文件中，例如：

```
@west @east : PPKS "user1" "thestringismeanttobearandomstr"
```

PPKS 选项指的是静态 PPK。这个实验性功能使用基于一次性平板的动态 PPK。在每个连接中，一次性平板的一个新部件用作 PPK。当使用时，文件中动态 PPK 的那部分被零覆盖，以防止重复使用。如果没有剩下一次性资源，连接会失败。详情请查看 **ipsec.secrets(5)** 手册页。

警告

动态 PPK 的实现是作为不受支持的技术预览提供的。请谨慎使用。

5.3. 安装 LIBRESWAN

在通过 Libreswan IPsec/IKE 实现设置 VPN 之前，您必须安装相应的软件包，启动 **ipsec** 服务，并在防火墙中允许服务。

先决条件

- **AppStream**存储库已启用。

流程

1. 安装 **libreswan** 软件包：

```
# dnf install libreswan
```

2. 如果要重新安装 Libreswan，请删除其旧的数据库文件，并创建一个新的数据库：

```
# systemctl stop ipsec
# rm /var/lib/ipsec/nss/*db
# ipsec initnss
```

3. 启动 **ipsec** 服务，并启用该服务，以便其在引导时自动启动：

```
# systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 和 AH 协议的 500 和 4500/UDP 端口：

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

5.4. 创建主机到主机的 VPN

您可以使用原始 RSA 密钥身份验证将 Libreswan 配置为在两个称为 *left* 和 *right* 的主机之间创建主机到主机的 IPsec VPN。

先决条件

- Libreswan 已安装，并在每个节点上启动了 **ipsec** 服务。

流程

1. 在每台主机上生成原始 RSA 密钥对：

```
# ipsec newhostkey
```

2. 上一步返回生成的密钥的 **ckaid**。在 *左* 主机上使用 **ckaid** 和以下命令，例如：

```
# ipsec showhostkey --left --ckaid 2d3ea57b61c9419dfd6cf43a1eb6cb306c0e857d
```

上一命令的输出生成了配置所需的 **leftrsasigkey=** 行。在第二台主机（*右*）上执行相同的操作：

```
# ipsec showhostkey --right --ckaid a9e1f6ce9ecd3608c24e8f701318383f41798f03
```

3. 在 **/etc/ipsec.d/** 目录中，创建一个新的 **my_host-to-host.conf** 文件。将上一步中 **ipsec showhostkey** 命令的输出中的 RSA 主机密钥写入新文件。例如：

```
conn mytunnel
    leftid=@west
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
```

4. 导入密钥后，重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

5. 加载连接：

```
# ipsec auto --add mytunnel
```

6. 建立隧道：

```
# ipsec auto --up mytunnel
```

7. 要在 **ipsec** 服务启动时自动启动隧道，请在连接定义中添加以下行：

```
auto=start
```

5.5. 配置站点到站点的 VPN

要创建站点到站点的 IPsec VPN，通过加入两个网络，在两个主机之间创建一个 IPsec 隧道。主机因此充当端点，它们配置为允许来自一个或多个子网的流量通过。因此您可以将主机视为到网络远程部分的网关。

站点到站点 VPN 的配置只能与主机到主机 VPN 不同，同时必须在配置文件中指定一个或多个网络或子网。

先决条件

- 已配置了主机到主机的 VPN。

流程

1. 将带有主机到主机 VPN 配置的文件复制到新文件中，例如：

```
# cp /etc/ipsec.d/my_host-to-host.conf /etc/ipsec.d/my_site-to-site.conf
```

2. 在上一步创建的文件中添加子网配置，例如：

```
conn mysubnet
    also=mytunnel
    leftsubnet=192.0.1.0/24
    rightsubnet=192.0.2.0/24
    auto=start

conn mysubnet6
    also=mytunnel
    leftsubnet=2001:db8:0:1::/64
    rightsubnet=2001:db8:0:2::/64
    auto=start

# the following part of the configuration file is the same for both host-to-host and site-to-site connections:

conn mytunnel
   leftid=@west
   left=192.1.2.23
   leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
   rightid=@east
   right=192.1.2.45
   rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
   authby=rsasig
```

5.6. 配置远程访问 VPN

公路勇士是指拥有移动客户端和动态分配的 IP 地址的旅行用户。移动客户端使用 X.509 证书进行身份验证。

以下示例显示了 **IKEv2** 的配置，并且避免使用 **IKEv1** XAUTH 协议。

在服务器中：

```
conn roadwarriors
    ikev2=insist
    # support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    fragmentation=yes
    left=1.2.3.4
    # if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
    # leftsubnet=10.10.0.0/16
    leftsubnet=0.0.0.0/0
    leftcert=gw.example.com
    leftid=%fromcert
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow or deny client
    # pam-authorize=yes
    # load connection, do not initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

在移动客户端（即 road warrior 的设备）上，使用与之前配置稍有不同的配置：

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
```

```
# allow narrowing to the server's suggested assigned IP and remote subnet
narrowing=yes
# support (roaming) MOBIKE clients (RFC 4555)
mobike=yes
# initiate connection
auto=start
```

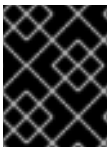
5.7. 配置网格 VPN

网格 VPN 网络（也称为 *any-to-any* VPN ） 是一个所有节点都使用 IPsec 进行通信的网络。该配置可以对于无法使用 IPsec 的节点进行例外处理。可使用两种方式配置网格 VPN 网络：

- 需要 IPsec。
- 首选 IPsec，但允许回退到使用明文通信。

节点之间的身份验证可以基于 X.509 证书或 DNS 安全扩展(DNSSEC)。

您可以对 *opportunistic IPsec* 使用任何常规 IKEv2 验证方法，因为这些连接是常规的 Libreswan 配置，除了由 **right=%opportunisticgroup** 条目定义的机会 IPsec 之外。常见的身份验证方法是用于主机使用常用的共享认证机构(CA)根据 X.509 证书互相验证。作为标准流程的一部分，云部署通常为云中的每个节点发布证书。



重要

不要使用 PreSharedKey (PSK)身份验证，因为一个被破坏的主机会导致组 PSK secret 也被破坏。

您可以使用 NULL 身份验证部署没有身份验证的节点间加密，这只防止被动攻击者。

以下流程使用 X.509 证书。您可以使用任何类型的 CA 管理系统（如 Dogtag 证书系统）生成这些证书。dogtag 假设每个节点的证书以 PKCSautomationhub 格式(**.p12** 文件)提供，其中包含私钥、节点证书和用于验证其他节点的 X.509 证书的 Root CA 证书。

每个节点的配置与其 X.509 证书不同。这允许在不重新配置网络中的任何现有节点的情况下添加新节点。PKCS #12 文件需要一个"友好名称"，为此，我们使用名称"节点"，这样引用友好名称的配置文件对所有节点都是相同的。

先决条件

- Libreswan 已安装 ，并在每个节点上启动了 **ipsec** 服务。
- 新的 NSS 数据库已初始化。

1. 如果您已经有旧的 NSS 数据库，请删除旧的数据库文件：

```
# systemctl stop ipsec
# rm /var/lib/ipsec/nss/*db
```

2. 您可以使用以下命令初始化新数据库：

```
# ipsec initnss
```

流程

1. 在每个节点中导入 PKCS #12 文件。此步骤需要用于生成 PKCS #12 文件的密码：

```
# ipsec import nodeXXX.p12
```

2. 为 **IPsec 需要的**（专用）、**IPsec 可选的** (private-or-clear)和 **No IPsec** (clear)配置文件创建以下三个连接定义：

```
# cat /etc/ipsec.d/mesh.conf
conn clear
 auto=ondemand (1)
 type=passthrough
 authby=never
 left=%defaultroute
 right=%group

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 ikev2=insist
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert (2)
 rightid=%fromcert
 right=%opportunisticgroup

conn private-or-clear
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=passthrough
 negotiationshunt=passthrough
 # left
 left=%defaultroute
 leftcert=nodeXXXX (3)
 leftid=%fromcert
 leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup
```

(1) **auto** 变量有几个选项：

您可以使用带有机会 IPsec 的 **ondemand** 连接选项来启动 IPsec 连接，或者显式配置不需要一直处于活动状态的连接。这个选项在内核中设置一个陷阱 XFRM 策略，使 IPsec 连接在收到与该策略匹配的第一个数据包时开始。

您可以使用以下选项有效地配置和管理 IPsec 连接，无论是使用机会 IPsec 还是明确配置的连接：

add 选项
加载连接配置，并准备好响应远程启动。但是，连接不会自动从本地端启动。您可以使用命令 **ipsec auto --up** 手动启动 IPsec 连接。

start 选项
加载连接配置，并准备好响应远程启动。此外，它会立即启动到远程对等点的连接。您可以将这个选项用于永久的和一直活跃的连接。

2 **leftid** 和 **rightid** 变量标识 IPsec 隧道连接的右侧和左侧频道。如果您配置了一个，您可以使用这些变量来获取本地 IP 地址或本地证书的主题 DN 的值。

3 **leftcert** 变量定义您要使用的 NSS 数据库的 nickname。

3. 将网络的 IP 地址添加到对应的类中。例如，如果所有节点都位于 **10.15.0.0/16** 网络中，则所有节点都必须使用 IPsec 加密：

```
# echo "10.15.0.0/16" >> /etc/ipsec.d/policies/private
```

4. 要允许某些节点（如 **10.15.34.0/24** ）使用或不使用 IPsec，请将这些节点添加到 private-or-clear 组中：

```
# echo "10.15.34.0/24" >> /etc/ipsec.d/policies/private-or-clear
```

5. 要将一个不支持 IPsec 的主机（如 **10.15.1.2** ）定义到 clear 组，请使用：

```
# echo "10.15.1.2/32" >> /etc/ipsec.d/policies/clear
```

您可以从每个新节点的模板中创建 **/etc/ipsec.d/policies** 目录中创建文件，也可以使用 Puppet 或 Ansible 来置备它们。

请注意，每个节点都有相同的异常列表或不同的流量预期。因此，两个节点可能无法通信，因为一个节点需要 IPsec，而另一个节点无法使用 IPsec。

6. 重启节点将其添加到配置的网格中：

```
# systemctl restart ipsec
```

验证

1. 使用 **ping** 命令打开 IPsec 隧道：

```
# ping <nodeYYY>
```

2. 使用导入的认证显示 NSS 数据库：

```
# certutil -L -d sql:/etc/ipsec.d

Certificate Nickname    Trust Attributes
                        SSL,S/MIME,JAR/XPI

west                     u,u,u
ca                      CT,,
```

3. 查看节点上打开了哪个隧道：

```
# ipsec trafficstatus
006 #2: "private#10.15.0.0/16"[1] ...<nodeYYY>, type=ESP, add_time=1691399301,
```

```
inBytes=512, outBytes=512, maxBytes=2^63B, id='C=US, ST=NC, O=Example
Organization, CN=east'
```

其他资源

- **IPsec.conf(5)** 手册页。
- 有关 **authby** 变量的更多信息，请参阅 6.2.Libreswan 中的身份验证方法。

5.8. 部署 FIPS 兼容 IPSEC VPN

您可以使用 Libreswan 部署 FIPS 兼容 IPsec VPN 解决方案。要做到这一点，您可以识别哪些加密算法可用，且 Libreswan 在 FIPS 模式中禁用。

先决条件

- **AppStream**存储库已启用。

流程

1. 安装 **libreswan** 软件包：

```
# dnf install libreswan
```

2. 如果您要重新安装 Libreswan，请删除其旧的 NSS 数据库：

```
# systemctl stop ipsec

# rm /var/lib/ipsec/nss/*db
```

3. 启动 **ipsec** 服务，并启用该服务，以便其在引导时自动启动：

```
# systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 和 AH 协议的 **500** 和 **4500** UDP 端口：

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

5. 将系统切换到 FIPS 模式：

```
# fips-mode-setup --enable
```

6. 重启您的系统以允许内核切换到 FIPS 模式：

```
# reboot
```

验证

1. 确认 Libreswan 在 FIPS 模式下运行：

```
# ipsec whack --fipsstatus
000 FIPS mode enabled
```

2. 或者，检查 **systemd** 日志中的 **ipsec** 单元条目：

```
$ journalctl -u ipsec
...
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Mode: YES
```

3. 以 FIPS 模式查看可用算法：

```
# ipsec pluto --selftest 2>&1 | head -6
Initializing NSS using read-write database "sql:/var/lib/ipsec/nss"
FIPS Mode: YES
NSS crypto library initialized
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity support [disabled]
```

4. 使用 FIPS 模式查询禁用的算法：

```
# ipsec pluto --selftest 2>&1 | grep disabled
Encryption algorithm CAMELLIA_CTR disabled; not FIPS compliant
Encryption algorithm CAMELLIA_CBC disabled; not FIPS compliant
Encryption algorithm NULL disabled; not FIPS compliant
Encryption algorithm CHACHA20_POLY1305 disabled; not FIPS compliant
Hash algorithm MD5 disabled; not FIPS compliant
PRF algorithm HMAC_MD5 disabled; not FIPS compliant
PRF algorithm AES_XCBC disabled; not FIPS compliant
Integrity algorithm HMAC_MD5_96 disabled; not FIPS compliant
Integrity algorithm HMAC_SHA2_256_TRUNCBUG disabled; not FIPS compliant
Integrity algorithm AES_XCBC_96 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. 在 FIPS 模式中列出所有允许的算法和密码：

```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
aes_ccm, aes_ccm_c
aes_ccm_b
aes_ccm_a
NSS(CBC)  3des
NSS(GCM)  aes_gcm, aes_gcm_c
NSS(GCM)  aes_gcm_b
NSS(GCM)  aes_gcm_a
NSS(CTR)  aesctr
NSS(CBC)  aes
aes_gmac
NSS       sha, sha1, sha1_96, hmac_sha1
NSS       sha512, sha2_512, sha2_512_256, hmac_sha2_512
NSS       sha384, sha2_384, sha2_384_192, hmac_sha2_384
NSS       sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
```

```
NSS(MODP) null, dh0
NSS(MODP) dh14
NSS(MODP) dh15
NSS(MODP) dh16
NSS(MODP) dh17
NSS(MODP) dh18
NSS(ECP)  ecp_256, ecp256
NSS(ECP)  ecp_384, ecp384
NSS(ECP)  ecp_521, ecp521
```

其他资源

- 使用系统范围的加密策略。

5.9. 使用密码保护 IPSEC NSS 数据库

默认情况下，IPsec 服务在第一次启动时使用空密码创建其网络安全服务(NSS)数据库。要提高安全性，您可以添加密码保护。

先决条件

- **/var/lib/ipsec/nss/** 目录包含 NSS 数据库文件。

流程

1. 为 Libreswan 的 **NSS** 数据库启用密码保护：

```
# certutil -N -d sql:/var/lib/ipsec/nss
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
```

2. 创建 **/etc/ipsec.d/nsspassword** 文件，其中包含您在上一步中设置的密码，例如：

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:_<password>_
```

nsspassword 文件使用以下语法：

```
<token_1>:<password1>
<token_2>:<password2>
```

默认的 NSS 软件令牌是 **NSS 证书 数据库**。如果您的系统以 FIPS 模式运行，则令牌的名称为 **NSS FIPS 140-2 证书数据库**。

3. 根据您的场景，在完成了 **nsspassword** 文件后，启动或重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 在其 NSS 数据库中添加非空密码后，检查 **ipsec** 服务是否运行：

```
# systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
  Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor preset: disable>
  Active: active (running)...
```

2. （可选）检查 **Journal** 日志是否包含确认成功初始化的条目：

```
# journalctl -u ipsec
...
pluto[6214]: Initializing NSS using read-write database "sql:/var/lib/ipsec/nss"
pluto[6214]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[6214]: NSS crypto library initialized
...
```

其他资源

- **certutil(1)** 手册页。
- 合规性活动和 政府标准 知识库文章 FIPS 140-2 和 FIPS 140-3。

5.10. 配置 IPSEC VPN 以使用 TCP

Libreswan 支持 IKE 和 IPsec 数据包的 TCP 封装，如 RFC 8229 所述。有了这个功能，您可以在网络上建立 IPsec VPN，以防止通过 UDP 和封装安全负载(ESP)传输的流量。您可以将 VPN 服务器和客户端配置为使用 TCP 作为回退，或者作为主 VPN 传输协议。由于 TCP 封装的性能成本较高，因此只有在您的场景中需要永久阻止 UDP 时，才使用 TCP 作为主 VPN 协议。

先决条件

- 已配置了 远程访问 VPN。

流程

1. 在 **/etc/ipsec.conf** 文件的 **config setup** 部分中添加以下选项：

```
listen-tcp=yes
```

2. 要在第一次尝试 UDP 失败时使用 TCP 封装作为回退选项，请在客户端的连接定义中添加以下两个选项：

```
enable-tcp=fallback
tcp-remoteport=4500
```

另外，如果您知道 UDP 会被永久阻止，请在客户端的连接配置中使用以下选项：

```
enable-tcp=yes
tcp-remoteport=4500
```

其他资源

- IETF RFC 8229:IKE 和 IPsec Packets 的 TCP 封装 。

5.11. 配置自动检测和使用 ESP 硬件卸载来加速 IPSEC 连接

卸载硬件的封装安全负载(ESP)来加速以太网上的 IPsec 连接。默认情况下，Libreswan 会检测硬件是否支持这个功能，并因此启用 ESP 硬件卸载。如果这个功能被禁用或被明确启用，您可以切回到自动检测。

先决条件

- 网卡支持 ESP 硬件卸载。
- 网络驱动程序支持 ESP 硬件卸载。
- IPsec 连接已配置且可以正常工作。

步骤

1. 编辑应使用 ESP 硬件卸载支持的自动检测连接的 **/etc/ipsec.d/** 目录中的 Libreswan 配置文件。
2. 确保 **nic-offload** 参数没有在连接的设置中设置。
3. 如果您删除了 **nic-offload**，请重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 显示 IPsec 连接使用的以太网设备的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 10
     rx_ipsec: 10
```

2. 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

```
# ping -c 5 remote_ip_address
```

3. 再次显示以太网设备的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 15
     rx_ipsec: 15
```

如果计数器值增加了，ESP 硬件卸载正常工作。

其他资源

- 使用 IPsec 配置 VPN

5.12. 在绑定中配置 ESP 硬件卸载以加快 IPSEC 连接

将封装安全负载(ESP)卸载到硬件可加速 IPsec 连接。如果出于故障转移原因而使用网络绑定，配置 ESP 硬件卸载的要求和流程与使用常规以太网设备的要求和流程不同。例如，在这种情况下，您可以对绑定启用卸载支持，内核会将设置应用到绑定的端口。

先决条件

- 绑定中的所有网卡都支持 ESP 硬件卸载。
- 网络驱动程序支持对绑定设备的 ESP 硬件卸载。在 RHEL 中，只有 **ixgbe** 驱动程序支持此功能。
- 绑定已配置且可以正常工作。
- 该绑定使用 **active-backup** 模式。绑定驱动程序不支持此功能的任何其他模式。
- IPsec 连接已配置且可以正常工作。

步骤

1. 对网络绑定启用 ESP 硬件卸载支持：

```
# nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

   这个命令在对 **bond0** 连接启用 ESP 硬件卸载支持。

2. 重新激活 **bond0** 连接：

```
# nmcli connection up bond0
```

3. 编辑应使用 ESP 硬件卸载的连接的 **/etc/ipsec.d/** 目录中的 Libreswan 配置文件，并将 **nic-offload=yes** 语句附加到连接条目：

```
conn example
    ...
    nic-offload=yes
```

4. 重启 **ipsec** 服务：

```
# systemctl restart ipsec
```

验证

1. 显示绑定的活动端口：

```
# grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

2. 显示活动端口的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 10
     rx_ipsec: 10
```

3. 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

```
# ping -c 5 remote_ip_address
```

4. 再次显示活动端口的 **tx_ipsec** 和 **rx_ipsec** 计数器：

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 15
     rx_ipsec: 15
```

如果计数器值增加了，ESP 硬件卸载正常工作。

其他资源

- 配置网络绑定
- 使用 IPsec 配置 VPN

5.13. 使用 RHEL 系统角色配置带有 IPSEC 的 VPN 连接

使用 **vpn** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 系统上配置 VPN 连接。您可以使用它来设置主机到主机、网络到网络、VPN 远程访问服务器和网格配置。

对于主机到主机连接，角色使用默认参数在 **vpn_connections** 列表中的每一对主机之间设置 VPN 通道，包括根据需要生成密钥。另外，您还可以将其配置为在列出的所有主机之间创建 机会主义网格配置。该角色假定 **hosts** 下的主机名称与 Ansible 清单中使用的主机的名称相同，并且您可以使用这些名称来配置通道。

注意

vpn RHEL 系统角色目前仅支持 Libreswan （即 IPsec 实现），作为 VPN 供应商。

5.13.1. 使用 vpn RHEL 系统角色使用 IPsec 创建主机到主机的 VPN

您可以通过在控制节点上运行 Ansible playbook 来使用 **vpn** 系统角色配置主机到主机的连接，这将配置清单文件中列出的所有受管节点。

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: Host to host VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
```

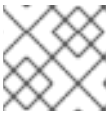
```
      managed-node-01.example.com:
      managed-node-02.example.com:
  vpn_manage_firewall: true
  vpn_manage_selinux: true
```

此 playbook 通过使用系统角色自动生成的密钥，配置 **managed-node-01.example.com-to-managed-node-02.example.com**。因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

要配置从受管主机到清单文件中未列出的外部主机的连接，请将以下部分添加到主机的 **vpn_connections** 列表中：

```
vpn_connections:
  - hosts:
      managed-node-01.example.com:
      <external_node>:
        hostname: <IP_address_or_hostname>
```

这将配置一个额外的连接： **managed-node-01.example.com-to-*<external_node>***



注意

连接仅在受管节点上配置，而不在外部节点上配置。

2. 可选：您可以使用 **vpn_connections** 中的附加部分为受管节点指定多个 VPN 连接，如 control plane 和数据平面：

```
- name: Multiple VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 192.0.2.0 # IP for the control plane
          managed-node-02.example.com:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 10.0.0.1 # IP for the data plane
          managed-node-02.example.com:
            hostname: 10.0.0.2
```

3. 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。

4. 运行 playbook：

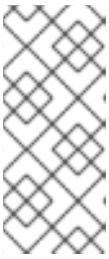
```
$ ansible-playbook ~/playbook.yml
```

验证

1. 在受管节点上，确认连接已成功载入：

```
# ipsec status | grep <connection_name>
```

将 ***<connection_name>*** > 替换为来自此节点的连接的名称，如 **managed_node1-to-managed_node2**。



注意

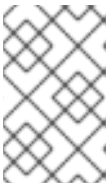
默认情况下，从每个系统的角度来看，角色为其创建的每个连接生成一个描述性名称。例如，当在 **managed_node1** 和 **managed_node2** 之间创建连接时，此连接在 **managed_node1** 上的描述性名称为 **managed_node1-to-managed_node2**，但在 **managed_node2** 上，连接的描述性名称为 **managed_node2-to-managed_node1**。

2. 在受管节点上，确认连接是否成功启动：

```
# ipsec trafficstatus | grep <connection_name>
```

3. 可选：如果连接没有成功加载，请输入以下命令来手动添加连接。这提供了更具体的信息，说明连接为何不能建立：

```
# ipsec auto --add <connection_name>
```



注意

加载和启动连接过程中可能会出现的任何错误都在 **/var/log/pluto.log** 文件中报告。由于这些日志很难解析，因此改为手动添加连接，以从标准输出获得日志消息。

其他资源

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

5.13.2. 使用 vpn RHEL 系统角色创建与 IPsec 的 opportunistic mesh VPN 连接

您可以使用 **vpn** 系统角色来配置机会主义网格 VPN 连接，该连接通过在控制节点上运行 Ansible playbook 来使用证书进行身份验证，这将配置清单文件中列出的所有受管节点。

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。

- **/etc/ipsec.d/** 目录中的 IPsec 网络安全服务(NSS)加密库包含必要的证书。

流程

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

```
- name: Mesh VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com, managed-node-
03.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
            cidr: 192.0.2.7/32
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

通过在 playbook 中定义 **auth_method: cert** 参数来配置用证书进行身份验证。默认情况下，节点名称用作证书的昵称。在本例中，这是 **managed-node-01.example.com**。您可以使用清单中的 **cert_name** 属性来定义不同的证书名称。

在本例中，控制节点是您运行 Ansible playbook 的系统，与两个受管节点(192.0.2.0/24)共享相同的无类别域间路由(CIDR)数，并且 IP 地址 192.0.2.7。因此，控制节点属于为 CIDR 192.0.2.0/24 自动创建的私有策略。

为防止在操作期间出现 SSH 连接丢失，控制节点的清晰策略包含在策略列表中。请注意，在策略列表中还有一个项 CIDR 等于 default。这是因为此 playbook 覆盖了默认策略中的规则，以使其为私有，而非私有或清晰。

因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 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
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** file

- **/usr/share/doc/rhel-system-roles/vpn/** directory

5.14. 配置选择不使用系统范围的加密策略的 IPSEC 连接

为连接覆盖系统范围的加密策略

RHEL 系统范围的加密策略会创建一个名为 **%default** 的特殊连接。此连接包含 **ikev2**、**esp** 和 **ike** 选项的默认值。但是，您可以通过在连接配置文件中指定上述选项来覆盖默认值。

例如，以下配置允许使用带有 AES 和 SHA-1 或 SHA-2的 IKEv1 连接，以及带有 AES-GCM 或 AES-CBC 的 IPsec(ESP) 连接：

```
conn MyExample
  ...
  ikev2=never
  ike=aes-sha2,aes-sha1;modp2048
  esp=aes_gcm,aes-sha2,aes-sha1
  ...
```

请注意，AES-GCM 可用于 IPsec(ESP)和 IKEv2，但不适用于 IKEv1。

为所有连接禁用系统范围的加密策略

要禁用所有 IPsec 连接的系统范围的加密策略，请在 **/etc/ipsec.conf** 文件中注释掉以下行：

```
include /etc/crypto-policies/back-ends/libreswan.config
```

然后将 **ikev2=never** 选项添加到连接配置文件。

其他资源

- 使用系统范围的加密策略。

5.15. IPSEC VPN 配置故障排除

与 IPsec VPN 配置相关的问题通常是由于几个主要原因造成的。如果您遇到此类问题，您可以检查问题的原因是否符合一下任何一种情况，并应用相应的解决方案。

基本连接故障排除

VPN 连接的大多数问题都发生在新部署中，管理员使用不匹配的配置选项配置了端点。此外，正常工作的配置可能会突然停止工作，通常是由于新引入的不兼容的值。这可能是管理员更改配置的结果。或者，管理员可能已安装了固件更新，或者使用某些选项的不同默认值（如加密算法）安装了软件包更新。

要确认已建立 IPsec VPN 连接：

```
# ipsec trafficstatus
006 #8: "vpn.example.com"[1] 192.0.2.1, type=ESP, add_time=1595296930, inBytes=5999,
outBytes=3231, id='@vpn.example.com', lease=100.64.13.5/32
```

如果输出为空或者没有显示具有连接名称的条目，则隧道将断开。

检查连接中的问题：

1. 重新载入 *vpn.example.com* 连接：

```
# ipsec auto --add vpn.example.com
002 added connection description "vpn.example.com"
```

2. 下一步，启动 VPN 连接：

```
# ipsec auto --up vpn.example.com
```

与防火墙相关的问题

最常见的问题是，其中一个 IPsec 端点或端点之间路由器上的防火墙将所有互联网密钥交换(IKE)数据包丢弃。

- 对于 IKEv2，类似以下示例的输出说明防火墙出现问题：

```
# ipsec auto --up vpn.example.com
181 "vpn.example.com"[1] 192.0.2.2 #15: initiating IKEv2 IKE SA
181 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: sent v2I1, expected v2R1
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 0.5
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 1
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 2
seconds for
...
```

- 对于 IKEv1，启动命令的输出如下：

```
# ipsec auto --up vpn.example.com
002 "vpn.example.com" #9: initiating Main Mode
102 "vpn.example.com" #9: STATE_MAIN_I1: sent MI1, expecting MR1
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 0.5 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 1 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 2 seconds for
response
...
```

由于用于设置 IPsec 的 IKE 协议已经加密，因此您只能使用 **tcpdump** 工具排除一小部分问题。如果防火墙丢弃了 IKE 或 IPsec 数据包，您可以尝试使用 **tcpdump** 工具来查找原因。但是，**tcpdump** 无法诊断 IPsec VPN 连接的其他问题。

- 捕获**eth0** 接口上的 VPN 协商以及所有加密数据：

```
# tcpdump -i eth0 -n -n esp or udp port 500 or udp port 4500 or tcp port 4500
```

不匹配的算法、协议和策略

VPN 连接要求端点具有匹配的 IKE 算法、IPsec 算法和 IP 地址范围。如果发生不匹配，连接会失败。如果您使用以下方法之一发现不匹配，请通过匹配算法、协议或策略来修复它。

- 如果远程端点没有运行 IKE/IPsec，您可以看到一个 ICMP 数据包来指示它。例如：

```
# ipsec auto --up vpn.example.com
...
000 "vpn.example.com"[1] 192.0.2.2 #16: ERROR: asynchronous network error report on
wlp2s0 (192.0.2.2:500), complainant 198.51.100.1: Connection refused [errno 111, origin
ICMP type 3 code 3 (not authenticated)]
...
```

- 不匹配 IKE 算法示例：

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com"[1] 193.110.157.148 #3: dropping unexpected IKE_SA_INIT message
containing NO_PROPOSAL_CHOSEN notification; message payloads: N; missing payloads:
SA,KE,Ni
```

- 不匹配 IPsec 算法示例：

```
# ipsec auto --up vpn.example.com
...
182 "vpn.example.com"[1] 193.110.157.148 #5: STATE_PARENT_I2: sent v2I2, expected
v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_256
group=MODP2048}
002 "vpn.example.com"[1] 193.110.157.148 #6: IKE_AUTH response contained the error
notification NO_PROPOSAL_CHOSEN
```

不匹配的 IKE 版本还可导致远程端点在没有响应的情况下丢弃请求。这与丢弃所有 IKE 数据包的防火墙相同。

- IKEv2 不匹配的 IP 地址范围示例（称为流量选择器 - TS）：

```
# ipsec auto --up vpn.example.com
...
1v2 "vpn.example.com" #1: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
002 "vpn.example.com" #2: IKE_AUTH response contained the error notification
TS_UNACCEPTABLE
```

- IKEv1 的不匹配 IP 地址范围示例：

```
# ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0
retransmits.  No acceptable response to our first Quick Mode message: perhaps peer likes
no proposal
```

- 当在 IKEv1 中使用预共享密钥(PSK)时，如果双方没有放入相同的 PSK，则整个 IKE 信息将无法读取：

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com" #1: received Hash Payload does not match computed value
223 "vpn.example.com" #1: sending notification INVALID_HASH_INFORMATION to
192.0.2.23:500
```

- 在 IKEv2 中，不匹配-PSK 错误会导致 AUTHENTICATION_FAILED 信息：

```
# ipsec auto --up vpn.example.com
...
002 "vpn.example.com" #1: IKE SA authentication request rejected by peer:
AUTHENTICATION_FAILED
```

最大传输单元

除防火墙阻止 IKE 或 IPsec 数据包外，网络问题的最常见原因与加密数据包的数据包大小增加有关。网络硬件对于大于最大传输单元(MTU)的数据包进行分片处理，例如 1500 字节。通常，片会丢失，数据包无法重新组装。当使用小数据包的 ping 测试可以正常工作，但其他流量失败时，这会导致间歇性故障。在这种情况下，您可以建立一个 SSH 会话，但是一使用它，终端就会冻结，例如，在远程主机上输入 'ls -al /usr' 命令。

要临时解决这个问题，请通过将 **mtu=1400** 选项添加到隧道配置文件中来减小 MTU 大小。

另外，对于 TCP 连接，启用更改 MSS 值的 iptables 规则：

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

如果上一命令没有解决您场景中的问题，请在 **set-mss** 参数中直接指定较小的数值：

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1380
```

网络地址转换(NAT)

当 IPsec 主机也充当 NAT 路由器时，可能会意外地重新映射数据包。以下示例配置演示了这个问题：

```
conn myvpn
    left=172.16.0.1
    leftsubnet=10.0.2.0/24
    right=172.16.0.2
    rightsubnet=192.168.0.0/16
…
```

地址为 172.16.0.1 的系统有一个 NAT 规则：

```
iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
```

如果地址为 10.0.2.33 的系统将数据包发送到 192.168.0.1，那么路由器会在应用 IPsec 加密前将源 10.0.2.33 转换为 172.16.0.1。

然后，源地址为 10.0.2.33 的数据包不再与 **conn myvpn** 配置匹配， IPsec 不会加密此数据包。

要解决这个问题，请在路由器上插入目标 IPsec 子网范围不包含 NAT 的规则，例如：

```
iptables -t nat -I POSTROUTING -s 10.0.2.0/24 -d 192.168.0.0/16 -j RETURN
```

内核 IPsec 子系统错误

例如，当 bug 导致 IKE 用户空间和 IPsec 内核不同步时，内核 IPsec 子系统可能会失败。检查此问题：

```
$ cat /proc/net/xfrm_stat
```

```
XfrmInError               0
XfrmInBufferError         0
...
```

上一命令输出中的任何非零值都表示有问题。如果您遇到这个问题，请开一个新的 支持问题单，并附上上一命令的输出与对应的 IKE 日志。

libreswan 日志

默认情况下，Libreswan 使用 **syslog** 协议的日志。您可以使用 **journalctl** 命令来查找与 IPsec 有关的日志条目。因为日志中相应的条目是由 **pluto** IKE 守护进程发送的，因此请搜索 "pluto" 关键字，例如：

```
$ journalctl -b | grep pluto
```

显示 **ipsec** 服务的实时日志：

```
$ journalctl -f -u ipsec
```

如果默认日志记录级别没有显示您的配置问题，请将 **plutodebug=all** 选项添加到 **/etc/ipsec.conf** 文件的 **config setup** 部分来启用调试日志。

请注意，调试日志记录会生成大量的条目，**journald** 或 **syslogd** 服务的速率可能会抑制 **syslog** 消息。要确保您有完整的日志，请将日志记录重定向到文件中。编辑 **/etc/ipsec.conf**，并在 **config setup** 部分中添加 **logfile=/var/log/pluto.log**。

其他资源

- 使用日志文件 对问题进行故障排除。
- **tcpdump(8)** 和 **ipsec.conf(5)** 手册页。
- 使用和配置 firewalld

5.16. 其他资源

- **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)** 和 **ipsec_rsasigkey(8)** 手册页。
- **/usr/share/doc/libreswan-*版本*/** 目录。
- Libreswan 项目 Wiki 。
- 所有 Libreswan 手册页 。
- NIST Special Publication 800-77:IPsec VPN 指南 .

第 6 章 保护网络服务

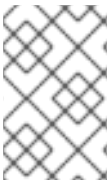
Red Hat Enterprise Linux 9 支持许多不同类型的网络服务器。这些服务器提供的网络服务可能会暴露系统的安全性，从而可能会面临各种类型的攻击，如拒绝服务攻击(DoS)、分布式拒绝服务攻击(DDoS)、脚本漏洞攻击和缓冲区溢出攻击。

为增加系统的安全性，务必要监控您使用的活跃网络服务。例如，当网络服务在计算机上运行时，其守护进程会侦听网络端口的连接，这可能会降低系统的安全性。要限制暴露会受到攻击的网络，应关闭所有未使用的服务。

6.1. 保护 RPCBIND 服务

rpcbind 服务是用于远程过程调用(RPC)服务的动态端口分配守护进程，如网络信息服务(NIS)和网络文件共享(NFS)。由于其身份验证机制较弱，并可为其控制的服务分配大量端口，因此保护 **rpcbind** 服务非常重要。

您可以通过限制访问所有网络并使用服务器上的防火墙规则来定义特定例外来保护 **rpcbind**。



注意

- **NFSv3** 服务器需要 **rpcbind** 服务。
- **NFSv4** 上不需要 **rpcbind** 服务。

先决条件

- 已安装 **rpcbind** 软件包。
- **firewalld** 软件包已安装，且服务正在运行。

流程

1. 添加防火墙规则，例如：
   - 限制 TCP 连接，并只接受来自 **192.168.0.0/24** 主机 **111** 端口的包：

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24" invert="True" drop'
     ```

   - 限制 TCP 连接，并只接受来自本地主机 **111** 端口的包：

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept'
     ```

   - 限制 UDP 连接，并只接受来自 **192.168.0.0/24** 主机 **111** 端口的包：

     ```
     # firewall-cmd --permanent --add-rich-rule='rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24" invert="True" drop'
     ```

     要使防火墙设置永久化，请在添加防火墙规则时使用 **--permanent** 选项。

2. 重新载入防火墙以应用新规则：

   ```
   # firewall-cmd --reload
   ```

-

验证步骤

- 列出防火墙规则：

```
# firewall-cmd --list-rich-rule
rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24"
invert="True" drop
rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept
rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24"
invert="True" drop
```

其他资源

- 有关 **只使用 NFSv4** 的服务器的详情，请参考配置只使用 NFSv4 的服务器
- 使用和配置 firewalld

6.2. 保护 RPC.MOUNTD 服务

rpc.mountd 守护进程实现 NFS 挂载协议的服务器端。NFS 版本 3(RFC 1813)使用 NFS 挂载协议。

您可以通过对服务器添加防火墙规则来保护 **rpc.mountd** 服务。您可以限制对所有网络的访问，并使用防火墙规则定义特定的异常。

先决条件

- 已安装 **rpc.mountd** 软件包。
- **firewalld** 软件包已安装，且服务正在运行。

流程

1. 在服务器中添加防火墙规则，例如：
   - 接受来自 **192.168.0.0/24** 主机的**mountd** 连接：

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" service name="mountd" source
address="192.168.0.0/24" invert="True" drop'
```

   - 接受来自本地主机的 **mountd** 连接：

```
# firewall-cmd --permanent --add-rich-rule 'rule family="ipv4" source address="127.0.0.1"
service name="mountd" accept'
```

   要使防火墙设置永久化，请在添加防火墙规则时使用 **--permanent** 选项。

2. 重新载入防火墙以应用新规则：

```
# firewall-cmd --reload
```

验证步骤

- 列出防火墙规则：

```
# firewall-cmd --list-rich-rule
rule family="ipv4" service name="mountd" source address="192.168.0.0/24" invert="True"
drop
rule family="ipv4" source address="127.0.0.1" service name="mountd" accept
```

其他资源

- 使用和配置 firewalld

6.3. 保护 NFS 服务

您可以使用 Kerberos 验证并加密所有文件系统操作来保护网络文件系统 4(NFSv4)。在将 NFSv4 与网络地址转换(NAT)或防火墙搭配使用时，您可以通过修改 **/etc/default/nfs** 文件来关闭委托。委托（Delegation）是服务器将文件管理委派给客户端的一种技术。

与其相反，NFSv3 不使用 Kerberos 锁定和挂载文件。

NFS 服务在所有 NFS 版本中使用 TCP 发送流量。该服务支持 Kerberos 用户和组身份验证，作为 **RPCSEC_GSS** 内核模块的一部分。

NFS 允许远程主机通过网络挂载文件系统，并与这些文件系统进行交互，就像它们被挂载到本地一样。您可以在集中服务器中合并资源，并在共享文件系统时额外自定义 **/etc/nfsmount.conf** 文件中的 NFS 挂载选项。

6.3.1. 保护 NFS 服务器的导出选项

NFS 服务器决定有关将哪些文件系统导出到 **/etc/exports** 文件中的目录和主机的列表结构。

警告

/etc/exports 文件语法中的额外空格可能会导致配置中的主要更改。

在以下示例中，**/tmp/nfs/** 目录与 **bob.example.com** 主机共享，并且具有读取和写入权限。

```
/tmp/nfs/     bob.example.com(rw)
```

以下示例与前一个相同，但对 **bob.example.com** 主机共享具有只读权限的相同的目录，由于主机名后面有一个空格字符，因此可以对 *world* 共享具有读写权限的目录。

```
/tmp/nfs/     bob.example.com (rw)
```

您可以通过输入 **showmount -e < *hostname> 命令来检查* 系统上的共享目录。

您可以在 **/etc/exports** 文件中使用以下导出选项：

警告

要导出整个文件系统，因为导出文件系统的子目录不安全。攻击者可能会访问部分导出的文件系统中的未导出部分。

ro

将 NFS 卷导出为只读。

rw

允许在 NFS 卷上读取和写入请求。请小心使用这个选项，因为允许写入访问会增加攻击的风险。如果您的场景需要使用 **rw** 选项挂载目录，请确保所有用户都无法写入，以减少可能的风险。

root_squash

将来自 **uid/gid** 0 的请求映射到匿名 **uid/gid**。这不适用于其它可能比较敏感的 **uid** 或 **gid**，如 **bin** 用户或 **staff** 组。

no_root_squash

关闭 root 压缩。默认情况下，NFS 共享将 **root** 用户改为 **nobody** 用户，这是一个非特权用户帐户。这会将所有 **root** 创建的文件的所有者改为 **nobody**，这样可以防止上传设置了 **setuid** 位的程序。如果使用 **no_root_squash** 选项，则远程 root 用户可以更改共享文件系统上的任何文件，并将感染特洛伊木马的应用程序留给其他用户。

secure

将导出限制为保留端口。默认情况下，服务器只允许客户端通信通过保留的端口。但是在网络上，任何人都可以容易地成为客户端上的 **root** 用户，因此，对于服务器来说，假设通过保留端口的通信都具有特权是不安全的。因此，对保留端口的限制具有有限的值；最好根据 Kerberos、防火墙和对特定客户端的导出限制来决定。

另外，在导出 NFS 服务器时请考虑以下最佳实践：

- 导出主目录存在风险，因为某些应用以纯文本或弱加密格式存储密码。您可以通过检查并改进应用程序代码来降低风险。
- 有些用户未对 SSH 密钥设置密码，这再次给主目录带来风险。您可以通过强制使用密码或使用 Kerberos 来降低这些风险。
- 将 NFS 导出仅限制为所需的客户端。在 NFS 服务器上使用 **showmount -e** 命令来检查服务器正在导出什么。不要导出不需要的任何内容。
- 不要允许不必要的用户登录到服务器，以减少攻击风险。您可以定期检查谁可以访问服务器，以及可以访问服务器的什么数据。

其他资源

- 使用红帽身份管理时在 IdM 中使用自动挂载
- **exports(5)** 和 **nfs(5)** 手册页

6.3.2. 保护 NFS 客户端的挂载选项

您可以将以下选项传递给 **mount** 命令，以增加基于 NFS 的客户端的安全性：

nosuid

使用 **nosuid** 选项禁用 **set-user-identifier** 或 **set-group-identifier** 位。这样可防止远程用户通过运行 **setuid** 程序获得更高的特权，并可使用此选项与 **setuid** 选项相反。

noexec

使用 **noexec** 选项禁用客户端上的所有可执行文件。使用此选项可防止用户意外执行位于共享文件系统中的文件。

nodev

使用 **nodev** 选项防止客户端将设备文件作为硬件设备处理。

resvport

使用 **resvport** 选项将通信限制到保留端口，您可以使用特权源端口与服务器通信。保留的端口是为特权用户和进程保留的，如 **root** 用户。

秒

使用 NFS 服务器上的 **sec** 选项，选择 RPCGSS security 类别来访问挂载点上的文件。有效的安全类别包括 **none**, **sys**, **krb5**, **krb5i**, 和 **krb5p**。

重要

krb5-libs 软件包提供的 MIT Kerberos 库不支持新部署中的数据加密标准(DES)算法。因为安全和兼容性的原因，在 Kerberos 库中默认禁用 DES。出于兼容性的原因，请使用更新、更安全的算法而不是 DES。

其他资源

- 常用的 NFS 挂载选项

6.3.3. 使用防火墙保护 NFS

要保护 NFS 服务器上的防火墙，请仅开放所需的端口。不要将 NFS 连接端口号用于任何其他服务。

先决条件

- 已安装 **nfs-utils** 软件包。
- **firewalld** 软件包已安装并运行。

流程

- 在 NFSv4 上，防火墙必须打开 TCP 端口 **2049**。
- 在 NFSv3 上，使用 **2049** 打开四个额外端口：

1. **rpcbind** 服务动态分配 NFS 端口，这可能会在创建防火墙规则时导致问题。要简化这个过程，使用 **/etc/nfs.conf** 文件指定要使用哪些端口：

   a. 在 **[mountd]** 部分为 **mountd** (**rpc.mountd**) 设置 TCP 和 UDP 端口，格式为 **port=*<value>***。

   b. 在 **[statd]** 部分为 **statd** (**rpc.statd**) 设置 TCP 和 UDP 端口，格式为 **port=*<value>***。

2. 在 **/etc/nfs.conf** 文件中为 NFS 锁定管理器(**nlockmgr**)设置 TCP 和 UDP 端口：

a. 在 **[lockd]** 部分为 **nlockmgr** (**rpc.statd**) 设置 TCP 端口，格式为 **port=value**。或者，也可以使用 **/etc/modprobe.d/lockd.conf** 文件中的 **nlm_tcpport** 选项。

b. 在 **[lockd]** 部分为 **nlockmgr** (**rpc.statd**) 设置 UDP 端口，格式为 **udp-port=value**。或者，您可以使用 **/etc/modprobe.d/lockd.conf** 文件中的 **nlm_udpport** 选项。

验证步骤

- 列出 NFS 服务器中的活跃端口和 RPC 程序：

```
$ rpcinfo -p
```

其他资源

- 使用红帽身份管理时在 IdM 中使用自动挂载
- **exports(5)** 和 **nfs(5)** 手册页

6.4. 保护 FTP 服务

您可以使用文件传输协议(FTP)来通过网络传输文件。因为服务器的所有 FTP 事务（包括用户身份验证）均未加密，因此请确保它被安全地配置。

RHEL 9 提供了两个 FTP 服务器：

红帽内容加速器(tux)

具有 FTP 功能的内核空间 Web 服务器。

非常安全的 FTP 守护进程(vsftpd)

FTP 服务的独立、面向安全的实现。

以下安全指南是用于设置 **vsftpd** FTP 服务：

6.4.1. 保护 FTP 的问候横幅

当用户连接到 FTP 服务时，FTP 会显示一个问候标语，它默认包含版本信息。攻击者可以利用此信息来识别系统中的弱点。您可以通过更改默认的横幅来隐藏此信息。

您可以通过编辑 **/etc/banners/ftp.msg** 文件来定义自定义横幅。它可以直接包含一个单行消息，或引用一个单独的文件，其中包含多行消息。

流程

- 要定义单行消息，请在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下选项：

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- 在单独的文件中定义信息：
  - 创建一个 **.msg** 文件，其中包含横幅消息，如 **/etc/banners/*ftp*.msg** ：

```
######### Hello, all activity on ftp.example.com is logged. #########
```

为了简化多个横幅的管理，请将所有横幅放在 **/etc/banners/** 目录中。

- 将横幅文件的路径添加到 **/etc/vsftpd/vsftpd.conf** 文件中的 **banner_file** 选项中：

```
banner_file=/etc/banners/ftp.msg
```

验证

- 显示修改后的横幅：

```
$ ftp localhost
Trying ::1…
Connected to localhost (::1).
Hello, all activity on ftp.example.com is logged.
```

6.4.2. 防止匿名访问并在 FTP 中上传

默认情况下，安装 **vsftpd** 软件包会为匿名用户创建 **/var/ftp/** 目录和一个目录树，用户对这些目录有只读权限。由于匿名用户可以访问数据，因此请勿将敏感数据存储在这些目录中。

要增加系统的安全性，您可以将 FTP 服务器配置为允许匿名用户将文件上传到特定目录并阻止匿名用户读取数据。在以下流程中，匿名用户可以将文件上传到 **root** 用户拥有的目录中，但不能更改该它。

流程

- 在 **/var/ftp/pub/** 目录中创建只写的目录：

```
# mkdir /var/ftp/pub/upload
# chmod 730 /var/ftp/pub/upload
# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

- 在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下行：

```
anon_upload_enable=YES
anonymous_enable=YES
```

- 可选：如果您的系统启用了 SELinux 并使用强制（enforcing）模式，请启用 SELinux 布尔值属性 **allow_ftpd_anon_write** 和 **allow_ftpd_full_access**。

警告

允许匿名用户在目录中读取和写入，可能会导致服务器成为盗取软件的存储库。

6.4.3. 为 FTP 保护用户帐户

FTP 通过不安全的网络以未加密的方式传输用户名和密码以进行身份验证。您可以通过拒绝系统用户从其用户帐户访问服务器来提高 FTP 安全性。

请根据您的配置执行以下几个步骤。

流程

- 通过在 **/etc/vsftpd/vsftpd.conf** 文件中添加以下行来禁用 **vsftpd** 服务器中的所有用户帐户：

```
local_enable=NO
```

- 要禁用特定帐户或特定帐户组（如 **root** 用户和带有 **sudo** 权限的组）对 FTP 的访问，您可以将用户名添加到 **/etc/pam.d/vsftpd** PAM 配置文件。
- 通过在 **/etc/vsftpd/ftpusers** 文件中添加用户名来禁用用户帐户。

6.4.4. 其他资源

- **ftpd_selinux(8)** 手册页

6.5. 保护 HTTP 服务器

6.5.1. httpd.conf 中的安全增强

您可以通过在 **/etc/httpd/conf/httpd.conf** 文件中配置安全选项来提高 Apache HTTP 服务器的安全性。

在将脚本加入到生产环境前，需要验证它们是否可以正常工作。

确保只有 **root** 用户对包含脚本或通用网关接口(CGI)的任何目录具有写入权限。要将目录所有权改为具有写入权限的 **root**，请输入以下命令：

```
# chown root <directory_name>
# chmod 755 <directory_name>
```

在 **/etc/httpd/conf/httpd.conf** 文件中，您可以配置以下选项：

FollowSymLinks

这个指令默认为启用，并遵循目录中的符号链接。

索引

这个指令被默认启用。禁用这个指令，以防止访问者浏览服务器上的文件。

UserDir

这个指令默认为禁用，因为它可以确认系统中存在用户帐户。要激活用户目录浏览 **/root/** 以外的所有用户目录，使用 **UserDir enabled** 和 **UserDir disabled** root 指令。要将用户添加到禁用帐户列表中，请在 **UserDir disabled** 行中添加以空格分隔的用户列表。

ServerTokens

这个指令控制向客户端发送的服务器响应标头字段。您可以使用以下参数来自定义信息：

ServerTokens Full

提供所有可用信息，如 Web 服务器版本号、服务器操作系统详情、已安装的 Apache 模块，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux) MyMod/1.2
```

ServerTokens Full-Release

提供与发行版本相关的所有可用信息，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux) (Release 41.module+el8.5.0+11772+c8e0c271)
```

ServerTokens Prod / ServerTokens ProductOnly

提供 Web 服务器名称，例如：

```
Apache
```

ServerTokens Major

提供 Web 服务器主版本，例如：

```
Apache/2
```

ServerTokens Minor

提供 Web 服务器次版本，例如：

```
Apache/2.4
```

ServerTokens Min / ServerTokens Minimal

提供 Web 服务器最小发行版本，例如：

```
Apache/2.4.37
```

ServerTokens OS

提供 Web 服务器发行版本和操作系统，例如：

```
Apache/2.4.37 (Red Hat Enterprise Linux)
```

使用 **ServerTokens Prod** 选项降低攻击者获取您系统的任何宝贵信息的风险。

重要

不要删除 **IncludesNoExec** 指令。默认情况下，Server Side Includes (SSI) 模块无法执行命令。更改这个设置可让攻击者在系统中输入命令。

删除 httpd 模块

您可以删除 **httpd** 模块来限制 HTTP 服务器的功能。编辑 **/etc/httpd/conf.modules.d/** 或 **/etc/httpd/conf.d/** 目录中的配置文件。例如，要删除代理模块：

```
echo '# All proxy modules disabled' > /etc/httpd/conf.modules.d/00-proxy.conf
```

其他资源

- Apache HTTP 服务器
- 为 Apache HTTP 服务器自定义 SELinux 策略

6.5.2. 保护 Nginx 服务器配置

Nginx 是一个高性能 HTTP 和代理服务器。您可以使用以下配置选项强化 Nginx 配置。

流程

- 要禁用版本字符串，修改 **server_tokens** 配置选项：

```
server_tokens off;
```

这个选项将停止显示其他详细信息，如服务器版本号。此配置仅显示 Nginx 提供的所有请求中的服务器名称，例如：

```
$ curl -sI http://localhost | grep Server
Server: nginx
```

- 添加额外的安全标头，以缓解特定 **/etc/nginx/** conf 文件中的某些已知 Web 应用程序漏洞：
  - 例如，**X-Frame-Options** 标头选项拒绝您的域之外的任何页面来帧由 Nginx 提供的任何内容，缓解了攻击：

```
add_header X-Frame-Options "SAMEORIGIN";
```

  - 例如，**x-content-type** 标头在某些较旧的浏览器中可以防止 MIME-type sniffing:

```
add_header X-Content-Type-Options nosniff;
```

  - 例如，**X-XSS-Protection** 标头启用跨站点脚本过滤(XSS)过滤，这可防止浏览器渲染由 Nginx 中包含的潜在的恶意内容：

```
add_header X-XSS-Protection "1; mode=block";
```

- 您可以限制公开的服务，并限制它们对访问者执行的操作，例如：

```
limit_except GET {
   allow 192.168.1.0/32;
   deny  all;
}
```

该片段将限制对除 **GET** 和 **HEAD** 外的所有方法的访问。

- 您可以禁用 HTTP 方法，例如：

```
# Allow GET, PUT, POST; return "405 Method Not Allowed" for all others.
if ( $request_method !~ ^(GET|PUT|POST)$ ) {
   return 405;
}
```

- 您可以配置 SSL 来保护 Nginx web 服务器提供的数据，请考虑仅通过 HTTPS 提供它。另外，您可以使用 Mozilla SSL 配置生成器生成在 Nginx 服务器中启用 SSL 的安全配置配置文件。生成的配置确保了，所有已知存在安全漏洞的协议（例如, SSLv2 和 SSLv3），加密程序和哈希算法（例如 3DES 和 MD5) 都已禁用。您还可以使用 SSL 服务器测试来验证您的配置是否满足现代安全要求。

其他资源

- Mozilla SSL 配置生成器
- SSL 服务器测试

6.6. 通过限制对经过身份验证的用户的访问来保护 POSTGRESQL

PostgreSQL 是一个对象相关数据库管理系统(DBMS)。在 Red Hat Enterprise Linux 中，PostgreSQL 由 **postgresql-server** 软件包提供。

您可以通过配置客户端身份验证来降低攻击的风险。**pg_hba.conf** 配置文件存储在数据库集群的数据目录中，控制客户端身份验证。按照以下步骤为基于主机的验证配置 PostgreSQL。

流程

1. 安装 PostgreSQL：

```
# yum install postgresql-server
```

2. 使用以下选项之一初始化数据库存储区域：

   a. 使用 **initdb** 工具：

   ```
   $ initdb -D /home/postgresql/db1/
   ```

   带有 **-D** 选项的 **initdb** 命令会创建您指定的目录（如果尚未存在），例如 ***/home/postgresql/db1/***。然后，此目录包含数据库中存储的所有数据以及客户端身份验证配置文件。

   b. 使用 **postgresql-setup** 脚本：

   ```
   $ postgresql-setup --initdb
   ```

   默认情况下，该脚本使用 **/var/lib/pgsql/data/** 目录。此脚本可以帮助具有基本数据库集群管理的系统管理员。

3. 要允许任何经过身份验证的用户使用其用户名访问任何数据库，请在 **pg_hba.conf** 文件中修改以下行：

```
local   all             all                                     trust
```

   当使用创建数据库用户和没有本地用户的层次应用程序时，这可能会造成问题。如果您不想显式控制系统中所有用户名，请从 **pg_hba.conf** 文件中删除 **local** 行条目。

4. 重启数据库以应用更改：

```
# systemctl restart postgresql
```

   上一命令更新数据库，同时还验证配置文件的语法。

6.7. 保护 MEMCACHED 服务

Memcached 是一个开源、高性能分布式内存对象缓存系统。它可以通过降低数据库负载来提高动态 Web 应用程序的性能。

Memcached 是一个内存键值存储，用于任意数据（如字符串和对象）的小块，来自于数据库调用、API 调用或页面渲染的结果。Memcached 允许将内存从未充分利用的区域分配给需要更多内存的应用程序。

2018 年，发现了向公共互联网公开的 Memcached 服务器漏洞 DDoS 扩展攻击。这些攻击利用了使用 UDP 协议进行传输的 Memcached 通信。这个攻击非常有效，因为其具有非常高的放大比率，具有几百字节大小的请求会产生带有几兆字节甚至几百兆字节的响应。

在大多数情况下，**memcached** 服务不需要向公共互联网公开。如果公开，其本身可能会带有安全问题。远程攻击者可能会泄漏或修改存储在 Memcached 中的信息。

6.7.1. 针对 DDoS 强化 Memcached

要降低安全风险，请根据您的配置执行以下步骤。

流程

- 在 LAN 中配置防火墙。如果您的 Memcached 服务器应只在本地网络访问，请不要将外部流量路由到 **memcached** 服务使用的端口。例如，从允许的端口列表中删除默认端口 **11211** ：

```
# firewall-cmd --remove-port=11211/udp
# firewall-cmd --runtime-to-permanent
```

- 如果您在与应用程序相同的机器上使用单一 Memcached 服务器，请设置 **memcached** 来仅侦听 localhost 流量。修改 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 值：

```
OPTIONS="-l 127.0.0.1,::1"
```

- 启用简单验证和安全层(SASL)身份验证：

  1. 修改或添加 **/etc/sasl2/memcached.conf** 文件：

     ```
     sasldb_path: /path.to/memcached.sasldb
     ```

  2. 在 SASL 数据库中添加帐户：

     ```
     # saslpasswd2 -a memcached -c cacheuser -f /path.to/memcached.sasldb
     ```

  3. 确保 **memcached** 用户和组可以访问数据库：

     ```
     # chown memcached:memcached /path.to/memcached.sasldb
     ```

  4. 通过在 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 参数中添加 **-S** 值在 Memcached 中启用 SASL 支持：

     ```
     OPTIONS="-S"
     ```

  5. 重启 Memcached 服务器以应用更改：

     ```
     # systemctl restart memcached
     ```

6. 将 SASL 数据库中创建的用户名和密码添加到应用程序的 Memcached 客户端配置中。

- 使用 TLS 加密 Memcached 客户端和服务器间的通信：

1. 通过在 **/etc/sysconfig/memcached** 文件中的 **OPTIONS** 参数中添加 **-Z** 值来启用 Memcached 客户端和服务器间的加密通信：

```
OPTIONS="-Z"
```

2. 使用 **-o ssl_chain_cert** 选项，以 PEM 格式添加证书链文件路径。

3. 使用 **-o ssl_key** 选项添加私钥文件路径。

第 7 章 使用 MACSEC 加密同一物理网络中的第 2 层流量

您可以使用 MACsec 来保护两个设备（点到点）之间的通信。例如，您的分支办公室通过城际以太网与中心办公室连接，您可以在连接办公室的两个主机上配置 MACsec，以提高安全性。

介质访问控制安全(MACsec)是一种第 2 层的协议，用于保护以太网链路上的不同的流量类型，包括：

- 动态主机配置协议(DHCP)
- 地址解析协议(ARP)
- 互联网协议版本 4 / 6(**IPv4** / **IPv6**)以及
- 任何使用 IP的流量（如 TCP 或 UDP）

MACsec 默认使用 GCM-AES-128 算法加密并验证 LAN 中的所有流量，并使用预共享密钥在参与的主机之间建立连接。如果要更改预共享密钥，您需要更新网络中使用 MACsec 的所有主机上的 NM 配置。

MACsec 连接将以太网设备（如以太网网卡、VLAN 或隧道设备）用作父设备。您可以只在 MACsec 设备上设置 IP 配置，以便只使用加密连接与其他主机进行通信，也可以在父设备上设置 IP 配置。在后者的情况下，您可以使用父设备使用未加密连接和 MACsec 设备加密连接与其他主机通信。

macsec 不需要任何特殊硬件。例如，您可以使用任何交换机，除非您只想在主机和交换机之间加密流量。在这种情况下，交换机还必须支持 MACsec。

换句话说，有 2 种常用的方法来配置 MACsec：

- 主机到主机，和
- 主机到交换机，然后交换机到其他主机

重要

您只能在相同（物理或虚拟）LAN 的主机间使用 MACsec。

7.1. 使用 NMCLI配置 MACSEC 连接

您可以使用 **nmcli** 工具将以太网接口配置为使用 MACsec。例如，您可以在通过以太网连接的两个主机之间创建一个 MACsec 连接。

流程

1. 在配置 MACsec 的第一个主机上：
   - 为预共享密钥创建连接关联密钥(CAK)和连接关联密钥名称(CKN)：
     a. 创建一个 16 字节的十六进制 CAK：

```
# dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
50b71a8ef0bd5751ea76de6d6c98c03a
```

     b. 创建一个 32 字节的十六进制 CKN：

```
# dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

2. 在您要通过 MACsec 连接连接的两个主机上：

3. 创建 MACsec 连接：

```
# nmcli connection add type macsec con-name macsec0 ifname macsec0
connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-
cak 50b71a8ef0bd5751ea76de6d6c98c03a macsec.mka-ckn
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

在**macsec.mka-cak** 和 **macsec.mka-ckn** 参数中使用上一步生成的 CAK 和 CKN。在 MACsec-protected 网络的每个主机上，这些值必须相同。

4. 配置 MACsec 连接中的 IP 设置。

   a. 配置 **IPv4** 设置。例如，要为 **macsec0** 连接设置静态 **IPv4** 地址、网络掩码、默认网关和 DNS 服务器，请输入：

```
# nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
'192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
```

   b. 配置 **IPv6** 设置。例如，要为 **macsec0** 连接设置静态 **IPv6** 地址、网络掩码、默认网关和 DNS 服务器，请输入：

```
# nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
'2001:db8:1::1/32' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd'
```

5. 激活连接：

```
# nmcli connection up macsec0
```

验证

1. 验证流量是否加密：

```
# tcpdump -nn -i enp1s0
```

2. 可选：显示未加密的流量：

```
# tcpdump -nn -i macsec0
```

3. 显示 MACsec 统计信息：

```
# ip macsec show
```

4. 显示每种保护类型的单独的计数器：仅完整性（关闭加密）和加密（打开加密）

```
# ip -s macsec show
```

7.2. 使用 NMSTATECTL 配置 MACSEC 连接

您可以通过 **nmstatectl** 工具，以声明的方式将以太网接口配置为使用 MACsec。例如，在 YAML 文件中，您可以描述网络所需的状态，假设其在通过以太网连接的两个主机之间有一个 MACsec 连接。**nmstatectl** 工具解释 YAML 文件，并在主机之间部署持久和一致的网络配置。

使用 MACsec 安全标准保护链路层的通信，也称为 Open Systems Interconnection (OSI)模型的第 2 层，提供以下显著优点：

- 第 2 层的加密消除了在第 7 层加密单个服务的需要。这减少了管理与每个主机上每个端点的大量证书关联的开销。
- 直接连接的网络设备（如路由器和交换机）之间的点对点安全性。
- 不需要对应用程序和高层协议进行更改。

先决条件

- 服务器配置中存在一个物理或虚拟以太网网络接口控制器(NIC)。
- **nmstate** 软件包已安装。

流程

1. 在您配置 MACsec 的第一个主机上，为预共享密钥创建连接关联密钥(CAK)和连接关联密钥名称(CKN)：
   a. 创建一个 16 字节的十六进制 CAK：

   ```
   # dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
   50b71a8ef0bd5751ea76de6d6c98c03a
   ```

   b. 创建一个 32 字节的十六进制 CKN：

   ```
   # dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
   f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
   ```

2. 在您要通过 MACsec 连接连接的两个主机上，完成以下步骤：
   a. 使用以下设置创建一个 YAML 文件，如 **create-macsec-connection.yml** ：

   ```
   ---
   routes:
     config:
     - destination: 0.0.0.0/0
       next-hop-interface: macsec0
       next-hop-address: 192.0.2.2
       table-id: 254
     - destination: 192.0.2.2/32
       next-hop-interface: macsec0
       next-hop-address: 0.0.0.0
       table-id: 254
   dns-resolver:
     config:
       search:
       - example.com
       server:
       - 192.0.2.200
   ```

```
    - 2001:db8:1::ffbb
interfaces:
- name: macsec0
  type: macsec
  state: up
  ipv4:
    enabled: true
    address:
    - ip: 192.0.2.1
      prefix-length: 32
  ipv6:
    enabled: true
    address:
    - ip: 2001:db8:1::1
      prefix-length: 64
  macsec:
    encrypt: true
    base-iface: enp0s1
    mka-cak: 50b71a8ef0bd5751ea76de6d6c98c03a
    mka-ckn: f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
    port: 0
    validation: strict
    send-sci: true
```

b. 在 **mka-cak** 和 **mka-ckn** 参数中使用上一步中生成的 CAK 和 CKN。在 MACsec-protected 网络的每个主机上，这些值必须相同。

c. 可选：在同一个 YAML 配置文件中，您还可以配置以下设置：

- 静态 IPv4 地址 - **192.0.2.1**，子网掩码为 **/32**
- 静态 IPv6 地址 - **2001:db8:1::1**，子网掩码为 **/64**
- IPv4 默认网关 - **192.0.2.2**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

3. 将设置应用到系统：

```
# nmstatectl apply create-macsec-connection.yml
```

验证

1. 以 YAML 格式显示当前状态：

```
# nmstatectl show macsec0
```

2. 验证流量是否加密：

```
# tcpdump -nn -i enp0s1
```

3. 可选：显示未加密的流量：

```
# tcpdump -nn -i macsec0
```

4. 显示 MACsec 统计信息：

```
# ip macsec show
```

5. 显示每种保护类型的单独的计数器：仅完整性（关闭加密）和加密（打开加密）

```
# ip -s macsec show
```

其他资源

- MACsec：加密网络流量的一个不同的解决方案

7.3. 其他资源

- MACsec：加密网络流量的另一种解决方案 博客。

第 8 章 保护 POSTFIX 服务

Postfix 是一个邮件传输代理 (MTA)，它使用简单邮件传输协议 (SMTP) 在不同的 MTA 间发送电子信息，并向电子邮件客户端或传输代理发送电子邮件。虽然 MTA 可以在不同代理间加密流量，但在默认情况下不会进行加密。您还可以通过将设置改为更安全的值来降低各种攻击的风险。

8.1. 减少 POSTFIX 网络相关的安全风险

要降低攻击者通过网络对系统进行攻击的风险，请尽可能多地执行以下任务。

- 不要在网络文件系统 (NFS) 共享卷上共享 **/var/spool/postfix/** mail spool 目录。NFSv2 和 NFSv3 不维护对用户和组 ID 的控制。因此，如果两个或多个用户具有相同的 UID，则他们可以接收和读取彼此的邮件，这是一个安全风险。

注意

此规则不适用于使用 Kerberos 的 NFSv4，因为 **SECRPC_GSS** 内核模块不使用基于 UID 的身份验证。但是，为了降低安全风险，您不应该将 mail spool 目录放在 NFS 共享卷上。

- 为了减少 Postfix 服务器漏洞的可能性，邮件用户必须使用一个电子邮件程序来访问 Postfix 服务器。在邮件服务器上不允许 shell 帐户，并将 **/etc/passwd** 文件中的所有用户 shell 设置为 **/sbin/nologin** (**root** 用户可能例外)。

- 为了防止 Postfix 被网络攻击，默认设置为仅侦听本地回环地址。您可以通过查看 **/etc/postfix/main.cf** 文件中的 **inet_interfaces = localhost** 行来验证这一点。这样可确保 Postfix 仅接受来自本地系统（而不是来自网络）的邮件（如 **cron** 作业报告）。这是默认设置，者可以保护 Postfix 免受网络攻击。要删除 localhost 限制并允许 Postfix 侦听所有接口，请将 **/etc/postfix/main.cf** 中的 **inet_interfaces** 参数设置为 **all**。

8.2. 用于限制 DOS 攻击的 POSTFIX 配置选项

攻击者可以利用流量处理服务器，或发送触发崩溃的信息，从而实现拒绝服务 (DoS) 攻击。您可以通过在 **/etc/postfix/main.cf** 文件中设置限制来降低此类攻击的风险。您可以更改现有指令的值，或者您可以使用 <directive> **= <** ***value> 格式的自定义值*** 添加新指令。

使用以下指令列表来限制 DoS 攻击：

smtpd_client_connection_rate_limit

限制任何客户端每个时间单元可以进行的最大连接尝试次数。默认值为 **0**，这意味着客户端每次时间可以接收 Postfix 允许的最多连接数量。默认情况下，指令会排除可信网络中的客户端。

anvil_rate_time_unit

定义计算速率限制的时间单位。默认值为 **60** 秒。

smtpd_client_event_limit_exceptions

从连接和速率限制命令中排除客户端。默认情况下，指令会排除可信网络中的客户端。

smtpd_client_message_rate_limit

定义每个时间单位从客户端发送到请求的最大消息数（无论 Postfix 是否实际接受这些消息）。

default_process_limit

定义提供给定服务的默认 Postfix 子进程的最大数。对于 **master.cf** 文件中的特定服务，您可以忽略此规则。默认情况下，值为 **100**。

queue_minfree

定义在队列文件系统中接收邮件所需的最小可用空间量。该指令目前由 Postfix SMTP 服务器使用，以决定是否接受任何邮件。默认情况下，当可用空间量小于 **message_size_limit** 的 1.5 倍时，Postfix SMTP 服务器会拒绝 **MAIL FROM** 命令。要指定一个更高的最小可用空间限制，为 **message_size_limit** 指定一个至少为 **queue_minfree** 1.5 倍的值。默认情况下，**queue_minfree** 值为 **0**。

header_size_limit

定义用于存储消息标头的最大内存量（以字节为单位）。如果标头较大，它会丢弃过量标头。默认情况下，值为 **102400** 字节。

message_size_limit

定义消息的最大大小（以字节为单位），包括信封信息。默认情况下，值为 **10240000** 字节。

8.3. 将 POSTFIX 配置为使用 SASL

Postfix 支持基于简单身份验证和安全层 (SASL) 的 SMTP 身份验证 (AUTH)。SMTP AUTH 是简单邮件传输协议的扩展。目前，Postfix SMTP 服务器通过以下方法支持 SASL 实现：

Dovecot SASL

Postfix SMTP 服务器可以使用 UNIX-domain 套接字或 TCP 套接字与 Dovecot SASL 实施通信。如果 Postfix 和 Dovecot 应用程序在独立的计算机上运行，则使用此方法。

Cyrus SASL

启用后，SMTP 客户端必须使用服务器和客户端支持并接受的身份验证方法与 SMTP 服务器进行身份验证。

先决条件

- **dovecot** 软件包安装在系统中

流程

1. 设置 Dovecot：

   a. 在 **/etc/dovecot/conf.d/10-master.conf** 文件中包括以下行：

```
service auth {
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
}
```

   前面的示例使用 UNIX-domain 套接字进行 Postfix 和 Dovecot 之间的通信。该示例还假定默认 Postfix SMTP 服务器设置，其中包括位于 **/var/spool/postfix/** 目录中的邮件队列，以及在 **postfix** 用户和组下运行的应用。

   b. 可选：将 Dovecot 设置为通过 TCP 侦听 Postfix 验证请求：

```
service auth {
  inet_listener {
      port = port-number
  }
}
```

c. 通过编辑 **/etc/dovecot/conf.d/10-auth.conf** 文件中的 **auth_mechanisms** 参数来指定电子邮件客户端用来通过 Dovecot 进行身份验证的方法：

```
auth_mechanisms = plain login
```

auth_mechanisms 参数支持不同的纯文本和非纯文本身份验证方法。

2. 通过修改 **/etc/postfix/main.cf** 文件来设置 Postfix：

a. 在 Postfix SMTP 服务器上启用 SMTP 身份验证：

```
smtpd_sasl_auth_enable = yes
```

b. 为 SMTP 身份验证启用 Dovecot SASL 实现：

```
smtpd_sasl_type = dovecot
```

c. 提供相对于 Postfix 队列目录的身份验证路径。请注意，使用相对路径可确保无论 Postfix 服务器是否在 **chroot** 中运行，配置都可以正常工作：

```
smtpd_sasl_path = private/auth
```

此步骤使用 UNIX-domain 套接字进行 Postfix 和 Dovecot 之间的通信。

要将 Postfix 配置为在不同机器上查找 Dovecot，如果您使用 TCP 套接字进行通信，请使用类似如下的配置值：

```
smtpd_sasl_path = inet: ip-address : port-number
```

在上例中，将 *ip-address* 替换为 Dovecot 机器的 IP 地址， *port-number* 替换为 Dovecot 的 **/etc/dovecot/conf.d/10-master.conf** 文件中的端口号。

d. 指定 Postfix SMTP 服务器可供客户端使用的 SASL 机制。请注意，您可以为加密和未加密的会话指定不同的机制。

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

以上指令指定在未加密的会话期间，不允许匿名身份验证，且不会允许传输未加密的用户名或密码的机制。对于使用 TLS 的加密会话，只允许非匿名身份验证机制。

其他资源

- Postfix SMTP 服务器策略 - SASL 机制属性
- Postfix 和 Dovecot SASL
- 在 Postfix SMTP 服务器中配置 SASL 身份验证