Red Hat Enterprise Linux 8

ネットワークのセキュリティー保護

セキュリティー保護されたネットワークおよびネットワーク通信の設定

Last Updated: 2024-03-24

Red Hat Enterprise Linux 8 ネットワークのセキュリティー保護

セキュリティー保護されたネットワークおよびネットワーク通信の設定

法律上の通知

概要

ネットワークのセキュリティーを向上させ、データ侵害や侵入のリスクを軽減するためのツールとテクニックを学びます。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 2 台のシステム間で OPENSSH を使用した安全な通信の使用

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間で安全な通信を提供し、ユーザーがリモートでサーバーホストシステムにログインできるようにするプロトコルです。FTP、Telnet などの他のリモート通信プロトコルとは異なり、SSH はログインセッションを暗号化するため、侵入者が接続して暗号化されていないパスワードを入手するのが困難になります。

Red Hat Enterprise Linux には、基本的な **OpenSSH** パッケージ (一般的な **openssh** パッケージ、**openssh-server** パッケージ、および **openssh-clients** パッケージ) が含まれます。**OpenSSH** パッケージには、**OpenSSL** パッケージ (**openssl-libs**) が必要です。このパッケージは、重要な暗号化ライブラリーをいくつかインストールして、暗号化通信を提供する **OpenSSH** を有効にします。

1.1. SSH と OPENSSH

SSH (Secure Shell) は、リモートマシンにログインしてそのマシンでコマンドを実行するプログラムです。SSH プロトコルは、安全でないネットワーク上で、信頼されていないホスト間で安全な通信を提供します。また、X11 接続と任意の TCP/IP ポートを安全なチャンネルで転送することもできます。

SSH プロトコルは、リモートシェルのログインやファイルコピー用に使用する場合に、システム間の通信の傍受や特定ホストの偽装など、セキュリティーの脅威を軽減します。これは、SSH クライアントとサーバーがデジタル署名を使用してそれぞれの ID を確認するためです。さらに、クライアントシステムとサーバーシステムとの間の通信はすべて暗号化されます。

ホストキーは、SSH プロトコルのホストを認証します。ホスト鍵は、OpenSSH の初回インストール時、またはホストの初回起動時に自動的に生成される暗号鍵です。

OpenSSH は、Linux、UNIX、および同様のオペレーティングシステムでサポートされている SSH プロトコルの実装です。OpenSSH クライアントとサーバー両方に必要なコアファイルが含まれます。OpenSSH スイートは、以下のユーザー空間ツールで構成されます。

- **SSH** は、リモートログインプログラム (SSH クライアント) です。
- **sshd** は、OpenSSH SSH デーモンです。
- **scp** は、安全なリモートファイルコピープログラムです。
- **sftp** は、安全なファイル転送プログラムです。
- **ssh-agent** は、秘密鍵をキャッシュする認証エージェントです。
- **ssh-add** は、秘密鍵の ID を **ssh-agent** に追加します。
- **ssh-keygen** が、**ssh** の認証キーを生成、管理、および変換します。
- **ssh-copy-id** は、ローカルの公開鍵をリモート SSH サーバーの **authorized_keys** ファイルに追加するスクリプトです。
- **ssh-keyscan** - SSH パブリックホストキーを収集します。

現在、SSH のバージョンには、バージョン 1 と新しいバージョン 2 の 2 つがあります。RHEL の OpenSSH スイートは、SSH バージョン 2 のみをサポートします。このスイートは、バージョン 1 で知られているエクスプロイトに対して脆弱ではない拡張キー交換アルゴリズムを備えています。

RHEL コア暗号化サブシステムの 1 つである OpenSSH は、システム全体の暗号化ポリシーを使用します。これにより、弱い暗号スイートおよび暗号化アルゴリズムがデフォルト設定で無効になります。ポリシーを変更するには、管理者が **update-crypto-policies** コマンドを使用して設定を調節するか、シス

テム全体の暗号化ポリシーを手動でオプトアウトする必要があります。

OpenSSH スイートは、2 セットの設定ファイルを使用します。1 つはクライアントプログラム (つまり、**ssh**、**scp**、および **sftp**) 用で、もう 1 つはサーバー (**sshd** デーモン) 用です。

システム全体の SSH 設定情報が **/etc/ssh/** ディレクトリーに保存されます。ユーザー固有の SSH 設定情報は、ユーザーのホームディレクトリーの **~/.ssh/** に保存されます。OpenSSH 設定ファイルの詳細なリストは、**sshd (8)** の man ページの **FILES** セクションを参照してください。

関連情報

- **man -k ssh** コマンドを使用してリスト表示される man ページ
- システム全体の暗号化ポリシーの使用

1.2. OPENSSH サーバーの設定および起動

お使いの環境と OpenSSH サーバーの起動に必要となる基本設定には、以下の手順を使用します。デフォルトの RHEL インストールを行うと、**sshd** デーモンがすでに起動し、サーバーのホスト鍵が自動的に作成されることに注意してください。

前提条件

- **openssh-server** パッケージがインストールされている。

手順

1. 現行セッションで **sshd** デーモンを開始し、ブート時に自動的に起動するように設定します。

```
# systemctl start sshd
# systemctl enable sshd
```

2. **/etc/ssh/sshd_config** 設定ファイルの **ListenAddress** ディレクティブに、デフォルトの **0.0.0.0** (IPv4) または **::** (IPv6) とは異なるアドレスを指定し、より低速な動的ネットワーク設定を使用するには、**network-online.target** ターゲットユニットの依存関係を **sshd.service** ユニットファイルに追加します。これを行うには、以下の内容で **/etc/systemd/system/sshd.service.d/local.conf** ファイルを作成します。

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

3. **/etc/ssh/sshd_config** 設定ファイルの OpenSSH サーバーの設定がシナリオの要件を満たしているかどうかを確認します。

4. 必要に応じて、**/etc/issue** ファイルを編集して、クライアント認証を行う前に OpenSSH サーバーに表示される welcome メッセージを変更します。以下に例を示します。

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

Banner オプションが **/etc/ssh/sshd_config** でコメントアウトされておらず、その値に **/etc/issue** が含まれていることを確認します。

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

ログインに成功すると表示されるメッセージを変更するには、サーバーの **/etc/motd** ファイルを編集する必要があります。詳細は、**pam_motd** の man ページを参照してください。

5. **systemd** 設定を再読み込みし、**sshd** を再起動して変更を適用します。

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

検証

1. **sshd** デーモンが実行していることを確認します。

```
# 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. SSH クライアントを使用して SSH サーバーに接続します。

```
# 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:
```

関連情報

- **sshd(8)** および **sshd_config(5)** の man ページ。

1.3. 鍵ベースの認証用の OPENSSH サーバーの設定

システムのセキュリティーを強化するには、OpenSSH サーバーでパスワード認証を無効にして鍵ベースの認証を有効にします。

前提条件

- **openssh-server** パッケージがインストールされている。
- サーバーで **sshd** デーモンが実行している。

手順

1. テキストエディターで **/etc/ssh/sshd_config** 設定を開きます。以下に例を示します。

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

2. **PasswordAuthentication** オプションを **no** に変更します。

```
PasswordAuthentication no
```

新しいデフォルトインストール以外のシステムで **PubkeyAuthentication no** が設定されていないことと、**ChallengeResponseAuthentication** ディレクティブが **no** に設定されていることを確認します。リモートで接続している場合は、コンソールもしくは帯域外アクセスを使用せず、パスワード認証を無効にする前に、鍵ベースのログインプロセスをテストします。

3. NFS がマウントされたホームディレクトリーで鍵ベースの認証を使用するには、SELinux ブール値 **use_nfs_home_dirs** を有効にします。

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

4. **sshd** デーモンを再読み込みし、変更を適用します。

```
# systemctl reload sshd
```

関連情報

- **sshd(8)**、**sshd_config(5)**、および **setsebool(8)** の man ページ。

1.4. SSH 鍵ペアの生成

以下の手順を使用して、ローカルシステムに SSH 鍵ペアを生成し、生成された公開鍵を OpenSSH サーバーにコピーします。サーバーが正しく設定されている場合は、パスワードなしで OpenSSH サーバーにログインできます。

重要

root で次の手順を完了すると、鍵を使用できるのは **root** だけとなります。

手順

1. SSH プロトコルのバージョン 2 用の ECDSA 鍵ペアを生成するには、次のコマンドを実行します。

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/joesec/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joesec/.ssh/id_ecdsa.
```

```
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@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]-----+
```

ssh-keygen コマンドまたは Ed25519 鍵ペアに **-t rsa** オプションを指定して RSA 鍵ペアを生成するには、**ssh-keygen -t ed25519** コマンドを実行します。

2. 公開鍵をリモートマシンにコピーするには、次のコマンドを実行します。

```
$ ssh-copy-id joesec@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
joesec@ssh-server-example.com's password:
...
Number of key(s) added: 1

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

セッションで **ssh-agent** プログラムを使用しない場合は、上記のコマンドで、最後に変更した **~/.ssh/id*.pub** 公開鍵をコピーします (インストールされていない場合)。別の公開キーファイルを指定したり、**ssh-agent** により、メモリーにキャッシュされた鍵よりもファイル内の鍵の方が優先順位を高くするには、**-i** オプションを指定して **ssh-copy-id** コマンドを使用します。

注記

システムを再インストールする際に、生成しておいた鍵ペアを引き続き使用する場合は、**~/.ssh/** ディレクトリーのバックアップを作成します。再インストール後に、このディレクトリーをホームディレクトリーにコピーします。これは、(**root** を含む) システムの全ユーザーで実行できます。

検証

1. パスワードなしで OpenSSH サーバーにログインします。

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

関連情報

- **ssh-keygen (1)** および **ssh-copy-id (1)** の man ページ

1.5. スマートカードに保存された SSH 鍵の使用

Red Hat Enterprise Linux では、OpenSSH クライアントでスマートカードに保存されている RSA 鍵および ECDSA 鍵を使用できるようになりました。この手順に従って、パスワードの代わりにスマートカードを使用した認証を有効にします。

前提条件

- クライアントで、**opensc** パッケージをインストールして、**pcscd** サービスを実行している。

手順

1. PKCS #11 の URI を含む OpenSC PKCS #11 モジュールが提供する鍵のリストを表示し、その出力を **keys.pub** ファイルに保存します。

```
$ 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. リモートサーバー (**example.com**) でスマートカードを使用した認証を有効にするには、公開鍵をリモートサーバーに転送します。前の手順で作成された **keys.pub** で **ssh-copy-id** コマンドを使用します。

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

3. 手順 1 の **ssh-keygen -D** コマンドの出力にある ECDSA 鍵を使用して **example.com** に接続するには、鍵を一意に参照する URI のサブセットのみを使用できます。以下に例を示します。

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

4. ~/**.ssh**/**config** ファイルで同じ URI 文字列を使用して、設定を永続化できます。

```
$ 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] $
```

OpenSSH は **p11-kit-proxy** ラッパーを使用し、OpenSC PKCS #11 モジュールが PKCS#11 キットに登録されているため、以前のコマンドを簡素化できます。

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

PKCS #11 の URI の **id=** の部分を飛ばすと、OpenSSH が、プロキシーモジュールで利用可能な鍵をすべて読み込みます。これにより、必要な入力の量を減らすことができます。

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

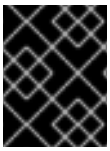
関連情報

- Fedora 28: Better smart card support in OpenSSH
- **p11-kit(8)**、**opensc.conf(5)**、**pcscd(8)**、**ssh(1)**、および **ssh-keygen(1)** の man ページ

1.6. OPENSSH のセキュリティーの強化

以下のヒントは、OpenSSH を使用する際にセキュリティーを高めるのに役に立ちます。OpenSSH 設定ファイル **/etc/ssh/sshd_config** を変更するには、**sshd** デーモンを再読み込みして有効にする必要があることに注意してください。

```
# systemctl reload sshd
```



重要

ほとんどのセキュリティー強化の設定変更により、最新のアルゴリズムまたは暗号スイートに対応していないクライアントとの互換性が低下します。

安全ではない接続プロトコルの無効化

- SSH を本当の意味で有効なものにするため、OpenSSH スイートに置き換えられる安全ではない接続プロトコルを使用しないようにします。このような接続プロトコルを使用すると、ユーザーのパスワード自体は SSH を使用した 1 回のセッションで保護されても、その後に Telnet を使用してログインした時に傍受されてしまうためです。このため、telnet、rsh、rlogin、ftp などの安全ではないプロトコルを無効にすることを検討してください。

鍵ベースの認証の有効化およびパスワードベースの認証の無効化

- 認証用パスワードを無効にして鍵のペアのみを許可すると、攻撃対象領域が減ってユーザーの時間を節約できる可能性があります。クライアントにおいて、**ssh-keygen** ツールを使用して鍵のペアを生成し、**ssh-copy-id** ユーティリティーを使用して OpenSSH サーバーのクライアントから公開鍵をコピーします。OpenSSH サーバーでパスワードベースの認証を無効にするには、**/etc/ssh/sshd_config** の **PasswordAuthentication** オプションを **no** に変更します。

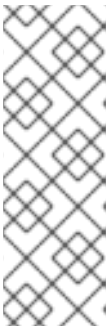
  ```
  PasswordAuthentication no
  ```

鍵のタイプ

- **ssh-keygen** コマンドは、デフォルトで RSA 鍵のペアを生成しますが、**-t** オプションを使用して ECDSA 鍵または Ed25519 鍵を生成するように指定できます。ECDSA (Elliptic Curve Digital Signature Algorithm) は、同等の対称鍵強度で RSA よりも優れたパフォーマンスを提供します。また、短いキーも生成します。Ed25519 公開鍵アルゴリズムは、RSA、DSA、および ECDSA より安全で高速な歪曲エドワーズ曲線の実装です。
  サーバーホストの鍵の RSA、ECDSA、および Ed25519 がない場合は、OpenSSH が自動的に作成します。RHEL でホストの鍵の作成を設定するには、インスタンス化したサービス **sshd-**

keygen@.service を使用します。たとえば、RSA 鍵タイプの自動作成を無効にするには、次のコマンドを実行します。

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

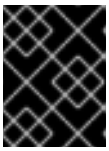


注記

cloud-init が有効になっているイメージでは、**ssh-keygen** ユニットが自動的に無効になります。これは、**ssh-keygen template** サービスが **cloud-init** ツールに干渉し、ホストキーの生成で問題が発生する可能性があるためです。これらの問題を回避するには、**cloud-init** が実行している場合に、**etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** ドロップイン設定ファイルにより **ssh-keygen** ユニットが無効になります。

- SSH 接続の特定の鍵タイプを除外するには、**/etc/ssh/sshd_config** で該当行をコメントアウトして **sshd** サービスを再読み込みします。たとえば、Ed25519 ホストキーだけを許可するには、次のコマンドを実行します。

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



重要

Ed25519 アルゴリズムは FIPS-140 に準拠していないため、OpenSSH は FIPS モードの Ed25519 キーでは機能しません。

デフォルト以外のポート

- デフォルトでは、**sshd** デーモンは TCP ポート 22 をリッスンします。ポートを変更すると、自動ネットワークスキャンに基づく攻撃にシステムがさらされる可能性が減るため、あいまいさによりセキュリティーが向上します。ポートは、**/etc/ssh/sshd_config** 設定ファイルの **Port** ディレクティブを使用して指定できます。
  また、デフォルト以外のポートを使用できるように、デフォルトの SELinux ポリシーも更新する必要があります。そのためには、**policycoreutils-python-utils** パッケージの **semanage** ツールを使用します。

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

さらに、**firewalld** 設定を更新します。

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

前のコマンドの **<port_number>** は、**Port** ディレクティブを使用して指定した新しいポート番号に置き換えます。

root ログインなし

- 特定の使用例で root ユーザーとしてログインする必要がない場合は、**/etc/ssh/sshd_config**

ファイルで **PermitRootLogin** 設定ディレクティブを **no** に設定できます。root ユーザーとしてログインする可能性を無効にすることにより、管理者は、通常のユーザーとしてログインし、root 権限を取得した後に、どのユーザーがどの特権コマンドを実行するかを監査できます。
または、**PermitRootLogin** を **prohibit-password** に設定します。

```
PermitRootLogin prohibit-password
```

これにより、root としてログインしてパスワードを使用する代わりに鍵ベースの認証が使用され、ブルートフォース攻撃を防ぐことでリスクが軽減します。

X セキュリティー拡張機能の使用

- Red Hat Enterprise Linux クライアントの X サーバーは、X セキュリティー拡張を提供しません。そのため、クライアントは X11 転送を使用して信頼できない SSH サーバーに接続するときに別のセキュリティー層を要求できません。ほとんどのアプリケーションは、この拡張機能を有効にしても実行できません。
デフォルトでは、**/etc/ssh/ssh_config.d/05-redhat.conf** ファイルの **ForwardX11Trusted** オプションが **yes** に設定され、**ssh -X remote_machine** コマンド (信頼できないホスト) と **ssh -Y remote_machine** コマンド (信頼できるホスト) には違いがありません。

  シナリオで X11 転送機能を必要としない場合は、**/etc/ssh/sshd_config** 設定ファイルの **X11Forwarding** ディレクティブを **no** に設定します。

特定のユーザー、グループ、またはドメインへのアクセス制限

- **/etc/ssh/sshd_config** 設定ファイルの **AllowUsers** ディレクティブおよび **AllowGroups** ディレクティブを使用すると、特定のユーザー、ドメイン、またはグループのみが OpenSSH サーバーに接続することを許可できます。**AllowUsers** および **AllowGroups** を組み合わせて、アクセスをより正確に制限できます。以下に例を示します。

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

  この設定行は、192.168.1.* サブネットおよび 10.0.0.* のサブネットのシステムの全ユーザーからの接続を許可します (192.168.1.2 アドレスのシステムを除く)。すべてのユーザーは、**example-group** グループに属している必要があります。OpenSSH サーバーは、その他のすべての接続を拒否します。

  OpenSSH サーバーは、**/etc/ssh/sshd_config** 内のすべての Allow および Deny ディレクティブを渡す接続のみを許可します。たとえば、**AllowUsers** ディレクティブに、**AllowGroups** ディレクティブにリストされているグループの一部ではないユーザーがリストされている場合、そのユーザーはログインできません。

  許可リストは、許可されていない新しいユーザーまたはグループもブロックするため、許可リスト (Allow で始まるディレクティブ) の使用は、拒否リスト (Deny で始まるオプション) を使用するよりも安全です。

システム全体の暗号化ポリシーの変更

- OpenSSH は、RHEL のシステム全体の暗号化ポリシーを使用し、デフォルトのシステム全体の暗号化ポリシーレベルは、現在の脅威モデルに安全な設定を提供します。暗号化の設定をより厳格にするには、現在のポリシーレベルを変更します。

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

-

警告

システムがインターネット上で通信する場合、**FUTURE** ポリシーの厳密な設定により、相互運用性の問題が発生する可能性があります。

システム全体の暗号化ポリシーにより、SSH プロトコルの特定の暗号のみを無効にすることもできます。詳細は、セキュリティーの強化 ドキュメントの サブポリシーを使用したシステム全体の暗号化ポリシーのカスタマイズ セクションを参照してください。

OpenSSH サーバーに対するシステム全体の暗号化ポリシーを除外するには、**/etc/sysconfig/sshd** ファイルの **CRYPTO_POLICY=** 変数行のコメントを除外します。この変更後、**/etc/ssh/sshd_config** ファイルの **Ciphers** セクション、**MACs** セクション、**KexAlgoritms** セクション、および **GSSAPIKexAlgorithms** セクションで指定した値は上書きされません。

詳細は、**sshd_config(5)** の man ページを参照してください。

OpenSSH クライアントのシステム全体の暗号化ポリシーをオプトアウトするには、次のいずれかのタスクを実行します。

- 指定のユーザーの場合は、**~/.ssh/config** ファイルのユーザー固有の設定でグローバルの **ssh_config** を上書きします。
- システム全体の場合は、**/etc/ssh/ssh_config.d/** ディレクトリーにあるドロップイン設定ファイルに暗号化ポリシーを指定します。このとき、辞書式順序で **05-redhat.conf** ファイルよりも前に来るように、5 未満の 2 桁の接頭辞と、**.conf** という接尾辞を付けます (例: **04-crypto-policy-override.conf**)。

関連情報

- **sshd_config(5)**、**ssh-keygen(1)**、**crypto-policies(7)**、および **update-crypto-policies(8)** の man ページ
- セキュリティー強化 ドキュメントの システム全体の暗号化ポリシーの使用
- ssh サービスのみに対して特定のアルゴリズムと暗号を無効化する方法 の記事

1.7. SSH ジャンプホストを使用してリモートサーバーに接続

この手順に従って、ジャンプホストとも呼ばれる中間サーバーを介してローカルシステムをリモートサーバーに接続します。

前提条件

- ジャンプホストでローカルシステムからの SSH 接続に対応している。
- リモートサーバーが、ジャンプホストからのみ SSH 接続を受け入れる。

手順

1. ローカルシステムの ~/**.ssh/config** ファイルを編集してジャンプホストを定義します。以下に例を示します。

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

   - **Host** パラメーターは、**ssh** コマンドで使用できるホストの名前またはエイリアスを定義します。値は実際のホスト名と一致可能ですが、任意の文字列にすることもできます。
   - **HostName** パラメーターは、ジャンプホストの実際のホスト名または IP アドレスを設定します。

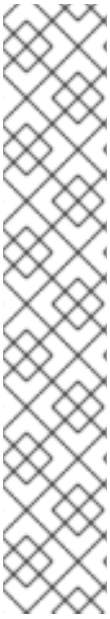
2. **ProxyJump** ディレクティブを使用してリモートサーバーのジャンプ設定を、ローカルシステムの ~/**.ssh/config** ファイルに追加します。以下に例を示します。

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

3. ローカルシステムを使用して、ジャンプサーバー経由でリモートサーバーに接続します。

```
$ ssh remote-server
```

   このコマンドは、設定手順 1 および 2 を省略したときの **ssh -J jump-server1 remote-server** コマンドと同じです。



注記

ジャンプサーバーをさらに指定することもできます。また、完全なホスト名を指定する場合は、設定ファイルへのホスト定義の追加を飛ばすこともできます。以下に例を示します。

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

ジャンプサーバーのユーザー名または SSH ポートが、リモートサーバーの名前およびポートと異なる場合は、上記のコマンドのホスト名のみの表記を変更します。以下に例を示します。

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@ju
mp3.example.com:75 joesec@remote1.example.com:220
```

関連情報

- **ssh_config(5)** および **ssh(1)** の man ページ

1.8. SSH-AGENT を使用して SSH キーでリモートマシンに接続する手順

パスフレーズを SSH 接続を開始するたびに入力しなくて済むようにするには、**ssh-agent** ユーティリティーを使用して SSH 秘密鍵をキャッシュします。秘密鍵とパスフレーズのセキュリティーが確保されます。

前提条件

前提条件

- SSH デーモンが実行中で、ネットワーク経由で到達可能なリモートホストがある。
- リモートホストにログインするための IP アドレスまたはホスト名および認証情報を把握している。
- パスフレーズで SSH キーペアを生成し、公開鍵をリモートマシンに転送している。

手順

1. 必要に応じて、キーを使用してリモートホストに対して認証できることを確認します。
   a. SSH を使用してリモートホストに接続します。

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

   b. 秘密鍵へのアクセス権を付与する鍵の作成時に指定したパスフレーズを入力します。

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

2. **ssh-agent** を起動します。

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

3. **ssh-agent** にキーを追加します。

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

検証

- オプション: SSH を使用してホストマシンにログインします。

  ```
  $ ssh example.user1@198.51.100.1

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

  パスフレーズを入力する必要がないことに注意してください。

1.9. 関連情報

- **sshd(8)**、**ssh(1)**、**scp(1)**、**sftp(1)**、**ssh-keygen(1)**、**ssh-copy-id(1)**、**ssh_config(5)**、**sshd_config(5)**、**update-crypto-policies(8)**、および **crypto-policies(7)** の man ページ
- OpenSSH のホームページ
- 非標準設定でのアプリケーションとサービスの SELinux 設定
- Controlling network traffic using firewalld

第2章 SSH システムロールを使用した安全な通信の設定

管理者は、**sshd** システムロールを使用して SSH サーバーを設定し、 **ssh** システムロールを使用して任意数の RHEL システムに同時に同じ設定の SSH クライアントを Red Hat Ansible Automation Platform で設定できます。

2.1. SSH SERVER のシステムロール変数

sshd システムロール Playbook では、設定と制限に応じて、SSH 設定ファイルのパラメーターを定義できます。

これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じ **sshd_config** ファイルを作成します。

どのような場合でも、ブール値は **sshd** 設定で適切に **yes** と **no** としてレンダリングされます。リストを使用して複数行の設定項目を定義できます。以下に例を示します。

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

レンダリングは以下のようになります。

```
ListenAddress 0.0.0.0
ListenAddress ::
```

sshd システムロールの変数

sshd_enable

false に設定すると、ロールは完全に無効になります。デフォルトは **true** です。

sshd_skip_defaults

true に設定すると、システムロールではデフォルト値が適用されません。代わりに、**sshd** ディクショナリーまたは **sshd_<OptionName>** 変数のいずれかを使用して、設定のデフォルトの完全なセットを指定します。デフォルトは **false** です。

sshd_manage_service

false に設定すると、サービスは管理されません。つまり、サービスは起動時に有効にならず、開始またはリロードされません。Ansible サービスモジュールは現在、AIX の **enabled** をサポートしていないため、コンテナーまたは AIX 内で実行する場合を除き、デフォルトは **true** になります。

sshd_allow_reload

false に設定すると、設定の変更後に **sshd** はリロードされません。これはトラブルシューティングで役立ちます。変更した設定を適用するには、**sshd** を手動で再読み込みします。AIX を除き、デフォルトは **sshd_manage_service** と同じ値になります。AIX では、**sshd_manage_service** のデフォルトは **false** ですが、**sshd_allow_reload** のデフォルトは **true** です。

sshd_install_service

true に設定すると、ロールは **sshd** サービスのサービスファイルをインストールします。これにより、オペレーティングシステムで提供されるファイルが上書きされます。2 番目のインスタンスを設定し、**sshd_service** 変数も変更する場合を除き、**true** に設定しないでください。デフォルトは **false** です。
ロールは、以下の変数でテンプレートとして参照するファイルを使用します。

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

sshd_service

この変数により **sshd** サービス名が変更されます。これは、2 つ目の **sshd** サービスインスタンスを設定するのに役立ちます。

sshd

設定が含まれるディクショナリー。以下に例を示します。

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_config(5) は、**sshd** ディクショナリーのすべてのオプションを一覧表示します。

sshd_<OptionName>

ディクショナリーの代わりに、**sshd_** 接頭辞とオプション名で構成される単純な変数を使用して、オプションを定義できます。簡単な変数は、**sshd** ディクショナリーの値をオーバーライドします。以下に例を示します。

```
sshd_Compression: no
```

sshd_config(5) は、**sshd** のすべてのオプションを一覧表示します。

sshd_manage_firewall

デフォルトのポート **22** 以外のポートを使用している場合は、この変数を **true** に設定します。**true** に設定すると、**sshd** ロールは **firewall** ロールを使用してポートアクセスを自動的に管理します。

注記

sshd_manage_firewall 変数はポートのみを追加できます。ポートは削除できません。ポートを削除するには、**firewall** システムロールを直接使用します。**firewall** システムロールを使用したポート管理の詳細は、Configuring ports by using System Roles を参照してください。

sshd_manage_selinux

デフォルトのポート **22** 以外のポートを使用している場合は、この変数を **true** に設定します。**true** に設定すると、**sshd** ロールは **selinux** ロールを使用してポートアクセスを自動的に管理します。

注記

sshd_manage_selinux 変数はポートのみを追加できます。ポートは削除できません。ポートを削除するには、**selinux** システムロールを直接使用します。

sshd_match and **sshd_match_1** to **sshd_match_9**

ディクショナリーのリスト、または Match セクションのディレクトリーのみ。これらの変数は、**sshd** ディクショナリーで定義されている一致ブロックをオーバーライドしないことに注意してください。すべてのソースは作成された設定ファイルに反映されます。

sshd_backup

false に設定すると、元の **sshd_config** ファイルはバックアップされません。デフォルトは **true** です。

sshd システムロールのセカンダリー変数

これらの変数を使用して、サポートされている各プラットフォームに対応するデフォルトを上書きすることができます。

sshd_packages

この変数を使用して、インストール済みパッケージのデフォルトリストを上書きできます。

sshd_config_owner、sshd_config_group、sshd_config_mode

このロールは、これらの変数を使用して生成する **openssh** 設定ファイルの所有権およびパーミッションを設定できます。

sshd_config_file

このロールが作成した **openssh** サーバー設定を保存するパス。

sshd_config_namespace

この変数のデフォルト値は null です。これは、ロールがシステムのデフォルトを含む設定ファイルの内容全体を定義することを意味します。または、この変数を使用して、他のロールから、またはドロップインディレクトリーをサポートしないシステムの 1 つの Playbook 内の複数の場所から、このロールを呼び出すことができます。**sshd_skip_defaults** 変数は無視され、この場合、システムのデフォルトは使用されません。
この変数が設定されている場合、ロールは指定された namespace の下の既存の設定ファイルの設定スニペットに指定する設定を配置します。シナリオにロールを複数回適用する必要がある場合は、アプリケーションごとに異なる namespace を選択する必要があります。

注記

openssh 設定ファイルの制限は引き続き適用されます。たとえば、設定ファイルで指定した最初のオプションだけが、ほとんどの設定オプションで有効です。

技術的には、ロールは他の一致ブロックが含まれていない限り、スニペットを "Match all" ブロックに配置し、既存の設定ファイル内の以前の一致ブロックに関係なく適用されるようにします。これにより、異なるロール呼び出しから競合しないオプションを設定できます。

sshd_binary

openssh の **sshd** 実行可能ファイルへのパス。

sshd_service

sshd サービスの名前。デフォルトでは、この変数には、ターゲットプラットフォームが使用する **sshd** サービスの名前が含まれます。ロールが **sshd_install_service** 変数を使用する場合は、これを使用してカスタムの **sshd** サービスの名前を設定することもできます。

sshd_verify_hostkeys

デフォルトは **auto** です。**auto** に設定すると、生成された設定ファイルに存在するホストキーがすべてリスト表示され、存在しないパスが生成されます。また、パーミッションおよびファイルの所有者はデフォルト値に設定されます。これは、サービスが最初の試行で開始できることを確認するためにデプロイメント段階でロールが使用される場合に便利です。このチェックを無効にするには、この変数を空のリスト **[]** に設定します。

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

これらの変数を使用して、**sshd_verify_hostkeys** からホストキーの所有権とパーミッションを設定します。

sshd_sysconfig

RHEL 8 以前のバージョンをベースとするシステムでは、この変数は **sshd** サービスの追加の詳細を設定します。**true** に設定すると、このロールは **sshd_sysconfig_override_crypto_policy** および **sshd_sysconfig_use_strong_rng** 変数に基づいて、**/etc/sysconfig/sshd** 設定ファイルも管理します。デフォルトは **false** です。

sshd_sysconfig_override_crypto_policy

RHEL 8 では、これを **true** に設定すると、**sshd** ディクショナリーまたは **sshd_<OptionName>** 形式で次の設定オプションを使用して、システム全体の暗号化ポリシーをオーバーライドできます。

- **Ciphers**
- **MACs**
- **GSSAPIKexAlgorithms**
- **GSSAPIKeyExchange** (FIPS-only)
- **KexAlgorithms**
- **HostKeyAlgorithms**
- **PubkeyAcceptedKeyTypes**
- **CASignatureAlgorithms**
  デフォルトは **false** です。

  RHEL 9 では、この変数は影響を及ぼしません。代わりに、**sshd** ディクショナリーまたは **sshd_<OptionName>** 形式で次の設定オプションを使用して、システム全体の暗号化ポリシーをオーバーライドできます。

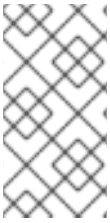
- **Ciphers**
- **MACs**
- **GSSAPIKexAlgorithms**
- **GSSAPIKeyExchange** (FIPS-only)
- **KexAlgorithms**
- **HostKeyAlgorithms**
- **PubkeyAcceptedAlgorithms**
- **HostbasedAcceptedAlgorithms**
- **CASignatureAlgorithms**
- **RequiredRSASize**
  sshd_config_file 変数で定義されたドロップインディレクトリーのカスタム設定ファイルに、これらのオプションを入力する場合は、暗号化ポリシーを含む **/etc/ssh/sshd_config.d/50-redhat.conf** ファイルより辞書順に前にあるファイル名を使用します。

sshd_sysconfig_use_strong_rng

RHEL 8 以前のバージョンをベースとするシステムでは、この変数により、**sshd** が、引数として指定されたバイト数を使用して **openssl** 乱数ジェネレーターを強制的に再シードできます。デフォルトは **0** で、この機能を無効にします。システムにハードウェア乱数ジェネレーターがない場合は、この機能を有効にしないでください。

2.2. SSHD システムロールを使用した OPENSSH サーバーの設定

sshd システムロールを使用して、Ansible Playbook を実行することで複数の SSH サーバーを設定できます。



注記

sshd システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、**sshd** ロールがネームスペース (RHEL 8 以前のバージョン) またはドロップインディレクトリー (RHEL 9) を使用していることを確認してください。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **sshd** システムロールの Playbook の例をコピーします。

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml ~/sshd-playbook.yml
```

2. 以下の例のように、テキストエディターでコピーした Playbook を開きます。

```
# vim ~/sshd-playbook.yml

---
- hosts: all
  tasks:
  - name: Configure sshd to prevent root and password login except from particular subnet
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd:
        # root login and password login is enabled only from a particular subnet
        PermitRootLogin: no
        PasswordAuthentication: no
        Match:
```

```
    - Condition: "Address 192.0.2.0/24"
      PermitRootLogin: yes
      PasswordAuthentication: yes
```

Playbook は、以下のように、マネージドノードを SSH サーバーとして設定します。

- パスワードと **root** ユーザーのログインが無効である
- **192.0.2.0/24** のサブネットからのパスワードおよび **root** ユーザーのログインのみが有効である

設定に合わせて変数を変更できます。詳細は、sshd システムロール変数 を参照してください。

3. Playbook の構文を検証します。

```
# ansible-playbook --syntax-check ~/sshd-playbook.yml
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

4. Playbook を実行します。

```
# ansible-playbook ~/sshd-playbook.yml

...

PLAY RECAP
*************************************************

localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

検証

1. SSH サーバーにログインします。

```
$ ssh user1@10.1.1.1
```

詳細は以下のようになります。

- **user1** は、SSH サーバーのユーザーです。
- **10.1.1.1** は、SSH サーバーの IP アドレスです。

2. SSH サーバーの **sshd_config** ファイルの内容を確認します。

```
$ cat /etc/ssh/sshd_config
…
PasswordAuthentication no
PermitRootLogin no
…
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
…
```

3. **192.0.2.0/24** サブネットから root としてサーバーに接続できることを確認します。

   a. IP アドレスを確認します。

   ```
   $ hostname -I
   192.0.2.1
   ```

   IP アドレスが **192.0.2.1** - **192.0.2.254** 範囲にある場合は、サーバーに接続できます。

   b. **root** でサーバーに接続します。

   ```
   $ ssh root@10.1.1.1
   ```

関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** の man ページ。

2.3. SSH システムロール変数

ssh システムロール Playbook では、設定および制限に応じて、クライアント SSH 設定ファイルのパラメーターを定義できます。

これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じグローバル **ssh_config** ファイルを作成します。

どのような場合でも、ブール値は **ssh** 設定で適切に **yes** または **no** とレンダリングされます。リストを使用して複数行の設定項目を定義できます。以下に例を示します。

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

レンダリングは以下のようになります。

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

注記

設定オプションでは、大文字と小文字が区別されます。

ssh システムロールの変数

ssh_user

システムロールでユーザー固有の設定を変更するように、既存のユーザー名を定義できます。ユーザー固有の設定は、指定したユーザーの **~/.ssh/config** に保存されます。デフォルト値は null で、すべてのユーザーに対するグローバル設定を変更します。

ssh_skip_defaults

デフォルトは **auto** です。**auto** に設定すると、システムロールはシステム全体の設定ファイル **/etc/ssh/ssh_config** を読み取り、そこで定義した RHEL のデフォルトを保持しま

す。**ssh_drop_in_name** 変数を定義してドロップイン設定ファイルを作成すると、**ssh_skip_defaults** 変数が自動的に無効化されます。

ssh_drop_in_name

システム全体のドロップインディレクトリーに置かれたドロップイン設定ファイルの名前を定義します。この名前は、変更する設定ファイルを参照するテンプレート **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** で使用されます。システムがドロップインディレクトリーに対応していない場合、デフォルト値は null です。システムがドロップインディレクトリーに対応している場合、デフォルト値は **00-ansible** です。

警告

システムがドロップインディレクトリーに対応していない場合は、このオプションを設定すると、プレイに失敗します。

推奨される形式は **NN-name** です。**NN** は、設定ファイルの指定に使用する 2 桁の番号で、 **name** はコンテンツまたはファイルの所有者を示す名前になります。

ssh

設定オプションとその値が含まれる dict。

ssh_OptionName

dict の代わりに、**ssh_** 接頭辞とオプション名で設定される単純な変数を使用してオプションを定義できます。簡単な変数は、**ssh** dict の値を上書きします。

ssh_additional_packages

このロールは、一般的なユースケースに必要な **openssh** パッケージおよび **openssh-clients** パッケージを自動的にインストールします。ホストベースの認証用に **openssh-keysign** などの追加のパッケージをインストールする必要がある場合は、この変数で指定できます。

ssh_config_file

ロールが生成した設定ファイルを保存するパス。デフォルト値:

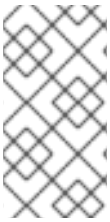
- システムにドロップインディレクトリーがある場合、デフォルト値は **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** テンプレートで定義されます。
- システムにドロップインディレクトリーがない場合、デフォルト値は **/etc/ssh/ssh_config** になります。
- **ssh_user** 変数が定義されている場合、デフォルト値は **~/.ssh/config** になります。

ssh_config_owner, ssh_config_group, ssh_config_mode

作成した設定ファイルの所有者、グループ、およびモード。デフォルトでは、ファイルの所有者は **root:root** で、モードは **0644** です。**ssh_user** が定義されている場合、モードは **0600** で、owner と group は **ssh_user** 変数で指定したユーザー名から派生します。

2.4. SSH システムロールを使用した OPENSSH クライアントの設定

ssh システムロールを使用して、Ansible Playbook を実行して複数の SSH クライアントを設定できます。



注記

ssh システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、**ssh** ロールがドロップインディレクトリー (RHEL 8 から デフォルト) を使用していることを確認してください。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/ssh-clients-playbook.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    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: example.com
            User: user1
      ssh_ForwardX11: no
```

この Playbook は、以下の設定を使用して、マネージドノードで **root** ユーザーの SSH クライアント設定を行います。

- 圧縮が有効になっている。
- ControlMaster multiplexing が **auto** に設定されている。
- **<example.com>** ホストに接続するための **<example>** エイリアスが **<user1>** である。
- **<example>** ホストエイリアスが作成されている。これはユーザー名が **<user1>** の **<example.com>** ホストへの接続を表します。
- X11 転送が無効化されている。

必要に応じて、これらの変数は設定に合わせて変更できます。詳細は、ssh システムロール変数 を参照してください。

2. Playbook の構文を検証します。

```
# ansible-playbook --syntax-check ~/ssh-clients-playbook.yml
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
# ansible-playbook ~/ssh-clients-playbook.yml
```

検証

- テキストエディターで SSH 設定ファイルを開いて、マネージドノードが正しく設定されていることを確認します。以下に例を示します。

```
# vi ~root/.ssh/config
```

上記の Playbook の例の適用後に、設定ファイルの内容は以下のようになるはずです。

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

2.5. 非排他的な設定のための SSHD システムロールの使用

通常、**sshd** システムロールを適用すると、設定全体が上書きされます。これは、たとえば別のシステムロールや Playbook などを使用して、以前に設定を調整している場合に問題を生じる可能性があります。他のオプションをそのまま維持しながら、選択した設定オプションのみに **sshd** システムロールを適用するには、非排他的設定を使用できます。

RHEL 8 以前では、設定スニペットを使用して非排他的設定を適用することができます。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

- 別の RHEL システムロールの Playbook。

手順

1. **sshd_config_namespace** 変数を含む設定スニペットを Playbook に追加します。

```
---
- hosts: all
  tasks:
  - name: <Configure SSHD to accept some useful environment variables>
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd_config_namespace: <my-application>
      sshd:
        # Environment variables to accept
        AcceptEnv:
          LANG
          LS_COLORS
          EDITOR
```

Playbook をインベントリーに適用すると、ロールによって次のスニペットが **/etc/ssh/sshd_config** ファイルに追加されます (まだ存在しない場合)。

```
# 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>
```

検証

- オプション: Playbook の構文を確認します。

```
# ansible-playbook --syntax-check <playbook.yml>
```

関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** の man ページ。

第3章 TLS キーと証明書の作成と管理

TLS (Transport Layer Security) プロトコルを使用して、2 つのシステム間で送信される通信を暗号化できます。この標準は、秘密鍵と公開鍵、デジタル署名、および証明書を用いた非対称暗号化を使用します。

3.1. TLS 証明書

TLS (Transport Layer Security) は、クライアントサーバーアプリケーションが情報を安全に受け渡すことを可能にするプロトコルです。TLS は、公開鍵と秘密鍵のペアのシステムを使用して、クライアントとサーバー間で送信される通信を暗号化します。TLS は、SSL (Secure Sockets Layer) の後継プロトコルです。

TLS は、X.509 証明書を使用して、ホスト名や組織などの ID を、デジタル署名を使用する公開鍵にバインドします。X.509 は、公開鍵証明書の形式を定義する標準です。

セキュアなアプリケーションの認証は、アプリケーションの証明書の公開鍵値の整合性によって異なります。攻撃者が公開鍵を独自の公開鍵に置き換えると、真のアプリケーションになりすまして安全なデータにアクセスできるようになります。この種の攻撃を防ぐには、すべての証明書が証明局 (CA) によって署名されている必要があります。CA は、証明書の公開鍵値の整合性を確認する信頼できるノードです。

CA はデジタル署名を追加して公開鍵に署名し、証明書を発行します。デジタル署名は、CA の秘密鍵でエンコードされたメッセージです。CA の公開鍵は、CA の証明書を配布することによって、アプリケーションで使用できるようになります。アプリケーションは、CA の公開鍵を使用して CA のデジタル署名をデコードして、証明書が有効で署名されていることを確認します。

CA によって署名された証明書を取得するには、公開鍵を生成し、署名のために CA に送信する必要があります。これは、証明書署名要求 (CSR) と呼ばれます。CSR には、証明書の識別名 (DN) も含まれています。どちらのタイプの証明書にも提供できる DN 情報には、国を表す 2 文字の国コード、州または県の完全な名前、市または町、組織の名前、メールアドレスを含めることも、空にすることもできます。現在の商用 CA の多くは、Subject Alternative Name 拡張を好み、CSR の DN を無視します。

RHEL は、TLS 証明書を操作するための 2 つの主要なツールキット (GnuTLS と OpenSSL) を提供します。**openssl** パッケージの **openssl** ユーティリティーを使用して、証明書を作成、読み取り、署名、および検証できます。**gnutls-utils** パッケージで提供される **certtool** ユーティリティーは、異なる構文を使用して同じ操作を行うことができ、特にバックエンドの異なるライブラリーセットを使用できます。

関連情報

- RFC 5280: インターネット X.509 公開キーインフラストラクチャー証明書および証明書失効リスト (CRL) プロファイル
- **openssl (1)**、**x509(1)**、**ca (1)**、**req (1)**、および **certtool (1)** の man ページ

3.2. OPENSSL を使用したプライベート CA の作成

プライベート証明機関 (CA) は、シナリオで内部ネットワーク内のエンティティーを検証する必要がある場合に役立ちます。たとえば、管理下にある CA によって署名された証明書に基づく認証を使用して VPN ゲートウェイを作成する場合、または商用 CA への支払いを希望しない場合は、プライベート CA を使用します。このようなユースケースで証明書に署名するために、プライベート CA は自己署名証明書を使用します。

前提条件

- **sudo** を使用して管理コマンドを入力するため **の root** 権限または権限がある。そのような特権を必要とするコマンドは、**#** でマークされています。

手順

1. CA の秘密鍵を生成します。たとえば、次のコマンドは、256 ビットの楕円曲線デジタル署名アルゴリズム (ECDSA) キーを作成します。

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

キー生成プロセスの時間は、ホストのハードウェアとエントロピー、選択したアルゴリズム、およびキーの長さによって異なります。

2. 前のコマンドで生成された秘密鍵を使用して署名された証明書を作成します。

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

生成された **ca.crt** ファイルは、10 年間、他の証明書の署名に使用できる自己署名 CA 証明書です。プライベート CA の場合、**<Example CA>** を共通名 (CN) として任意の文字列に置き換えることができます。

3. CA の秘密鍵に安全なアクセス許可を設定します。次に例を示します。

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

次のステップ

- 自己署名 CA 証明書をクライアントシステムのトラストアンカーとして使用するには、CA 証明書をクライアントにコピーし、クライアントのシステム全体のトラストストアに **root** として追加します。

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

詳細は、*4章 共通システム証明書の使用* を参照してください。

検証

1. 証明書署名要求 (CSR) を作成し、CA を使用して要求に署名します。CA は、CSR に基づいて証明書を正常に作成する必要があります。次に例を示します。

```
$ 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
```

詳細は、「プライベート CA を使用した OpenSSL での CSR の証明書の発行」 を参照してください。

2. 自己署名 CA に関する基本情報を表示します。

```
$ 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. 秘密鍵の一貫性を確認します。

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

18wRpXoCVC9vcPki7WIT+gnmCI+hRANCAARb9NxIvkaVjFhOoZbGp/HtIQxbM78E
lwbDP0BI624xBJ8gK68ogSaq2x4SdezFdV1gNeKScDcU+Pj2pELldmdF
-----END PRIVATE KEY-----
```

関連情報

- **openssl (1)**、**ca (1)**、**genpkey (1)**、**x509(1)**、および **req (1)** の man ページ

3.3. OPENSSL を使用した TLS サーバー証明書の秘密鍵と CSR の作成

認証局 (CA) からの有効な TLS 証明書がある場合にのみ、TLS 暗号化通信チャネルを使用できます。証明書を取得するには、最初にサーバーの秘密鍵と証明書署名要求 (CSR) を作成する必要があります。

手順

1. 以下のようにサーバーシステムで秘密鍵を生成します。

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

2. オプション: 次の例のように、選択したテキストエディターを使用して、CSR の作成を簡素化する設定ファイルを準備します。

```
$ 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 オプションは、証明書の使用を制限します。

3. 前に作成した秘密鍵を使用して CSR を作成します。

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

-config オプションを省略すると、**req** ユーティリティーは次のような追加情報の入力を求めます。

```
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>
```

次のステップ

- 署名のために選択した CA に CSR を送信します。または、信頼できるネットワーク内での内部使用シナリオでは、署名にプライベート CA を使用します。詳細は、「プライベート CA を使用した OpenSSL での CSR の証明書の発行」 を参照してください。

検証

1. 要求された証明書を CA から取得したら、次の例のように、証明書の中で人間が判読できる部分が要件と一致することを確認します。

```
$ 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
…
```

関連情報

- **openssl (1)**、**x509(1)**、**genpkey (1)**、**req (1)**、および **config (5)** のman ページ

3.4. OPENSSL を使用した TLS クライアント証明書の秘密鍵と CSR の作成

認証局 (CA) からの有効な TLS 証明書がある場合にのみ、TLS 暗号化通信チャネルを使用できます。証明書を取得するには、最初にクライアントの秘密鍵と証明書署名要求 (CSR) を作成する必要があります。

手順

1. 次の例のように、クライアントシステムで秘密鍵を生成します。

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

2. オプション: 次の例のように、選択したテキストエディターを使用して、CSR の作成を簡素化する設定ファイルを準備します。

```
$ 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 オプションは、証明書の使用を制限します。

3. 前に作成した秘密鍵を使用して CSR を作成します。

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

-config オプションを省略すると、**req** ユーティリティーは次のような追加情報の入力を求めます。

```
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>
```

次のステップ

- 署名のために選択した CA に CSR を送信します。または、信頼できるネットワーク内での内部使用シナリオでは、署名にプライベート CA を使用します。詳細は、「プライベート CA を使用した OpenSSL での CSR の証明書の発行」 を参照してください。

検証

1. 次の例のように、証明書の中で人間が判読できる部分が要件と一致することを確認します。

```
$ 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
…
```

関連情報

- **openssl (1)**、**x509(1)**、**genpkey (1)**、**req (1)**、および **config (5)** のman ページ

3.5. プライベート CA を使用した OPENSSL での CSR の証明書の発行

システムが TLS 暗号化通信チャネルを確立できるようにするには、認証局 (CA) が有効な証明書をシステムに提供する必要があります。プライベート CA がある場合は、システムからの証明書署名要求 (CSR) に署名することにより、要求された証明書を作成できます。

前提条件

- プライベート CA が設定済みである。詳細は、「OpenSSL を使用したプライベート CA の作成」 を参照してください。
- CSR を含むファイルがある。CSR の作成例は、「OpenSSL を使用した TLS サーバー証明書の秘密鍵と CSR の作成」 にあります。

手順

1. オプション: 任意のテキストエディターを使用して、次の例のように、証明書に拡張機能を追加するための OpenSSL 設定ファイルを準備します。

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

```
extendedKeyUsage = serverAuth

[client-cert]
extendedKeyUsage = clientAuth
```

2. **x509** ユーティリティーを使用して、CSR に基づいて証明書を作成します。次に例を示します。

```
$ openssl x509 -req -in <server-cert.csr> -CA <ca.crt> -CAkey <ca.key> -CAcreateserial -
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
```

セキュリティーを強化するには、CSR から別の証明書を作成する前に、シリアル番号ファイルを削除してください。こうすることで、シリアル番号が常に無作為になるようにします。カスタムファイル名を指定する **CAserial** オプションを省略した場合、シリアル番号のファイル名は証明書のファイル名と同じですが、その拡張子は **.srl** 拡張子に置き換えられます (前の例では **server-cert.srl**)。

関連情報

- **openssl (1)**、**ca (1)**、および **x509(1)** のman ページ

3.6. GNUTLS を使用したプライベート CA の作成

プライベート証明機関 (CA) は、シナリオで内部ネットワーク内のエンティティーを検証する必要がある場合に役立ちます。たとえば、管理下にある CA によって署名された証明書に基づく認証を使用して VPN ゲートウェイを作成する場合、または商用 CA への支払いを希望しない場合は、プライベート CA を使用します。このようなユースケースで証明書に署名するために、プライベート CA は自己署名証明書を使用します。

前提条件

- **sudo** を使用して管理コマンドを入力するため **の root** 権限または権限がある。そのような特権を必要とするコマンドは、**#** でマークされています。
- システムにはすでに GnuTLS がインストールされている。インストールしていない場合は、次のコマンドを使用できます。

```
$ yum install gnutls-utils
```

手順

1. CA の秘密鍵を生成します。たとえば、次のコマンドは、256 ビットの楕円曲線デジタル署名アルゴリズム (ECDSA) キーを作成します。

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

キー生成プロセスの時間は、ホストのハードウェアとエントロピー、選択したアルゴリズム、およびキーの長さによって異なります。

2. 証明書のテンプレートファイルを作成します。

a. 任意のテキストエディターでファイルを作成します。以下に例を示します。

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

b. ファイルを編集して、必要な証明書の詳細を含めます。

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

3. 手順 1 で生成した秘密鍵を使用して署名された証明書を作成します。
生成された **<ca.crt>** ファイルは、1 年間他の証明書に署名するために使用できる自己署名 CA 証明書です。**<ca.crt>** ファイルは公開鍵 (証明書) です。ロードされたファイル **<ca.key>** が秘密鍵です。このファイルは安全な場所に保管してください。

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

4. CA の秘密鍵に安全なアクセス許可を設定します。次に例を示します。

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

次のステップ

- 自己署名 CA 証明書をクライアントシステムのトラストアンカーとして使用するには、CA 証明書をクライアントにコピーし、クライアントのシステム全体のトラストストアに **root** として追加します。

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

詳細は、*4章 共通システム証明書の使用* を参照してください。

検証

1. 自己署名 CA に関する基本情報を表示します。

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

2. 証明書署名要求 (CSR) を作成し、CA を使用して要求に署名します。CA は、CSR に基づいて証明書を正常に作成する必要があります。次に例を示します。

   a. CA の秘密鍵を生成します。

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

   b. 任意のテキストエディターで新しい設定ファイルを開きます。以下に例を示します。

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

   c. ファイルを編集して、必要な証明書の詳細を含めます。

   ```
   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. 以前に作成した秘密鍵を使用してリクエストを生成します

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

   e. 証明書を生成し、CA の秘密鍵で署名します。

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

   関連情報

   - **certtool(1)** および **trust(1)** の man ページ

3.7. GNUTLS を使用した TLS サーバー証明書の秘密鍵と CSR の作成

証明書を取得するには、最初にサーバーの秘密鍵と証明書署名要求 (CSR) を作成する必要があります。

手順

1. 以下のようにサーバーシステムで秘密鍵を生成します。

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

2. オプション: 次の例のように、選択したテキストエディターを使用して、CSR の作成を簡素化する設定ファイルを準備します。

```
$ 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. 前に作成した秘密鍵を使用して CSR を作成します。

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

--template オプションを省略すると、**certool** ユーティリティーは次のような追加情報の入力を求めます。

```
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>
```

次のステップ

- 署名のために選択した CA に CSR を送信します。または、信頼できるネットワーク内での内部使用シナリオでは、署名にプライベート CA を使用します。詳細は、「プライベート CA を使用した GnuTLS での CSR の証明書の発行」 を参照してください。

検証

1. 要求された証明書を CA から取得したら、次の例のように、証明書の中で人間が判読できる部分が要件と一致することを確認します。

```
$ 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
…
```

関連情報

- **certtool(1)** の man ページ

3.8. GNUTLS を使用した TLS クライアント証明書の秘密鍵と CSR の作成

証明書を取得するには、最初にクライアントの秘密鍵と証明書署名要求 (CSR) を作成する必要があります。

手順

1. 次の例のように、クライアントシステムで秘密鍵を生成します。

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

2. オプション: 次の例のように、選択したテキストエディターを使用して、CSR の作成を簡素化する設定ファイルを準備します。

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

tls_www_client

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

3. 前に作成した秘密鍵を使用して CSR を作成します。

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

--template オプションを省略すると、**certtool** ユーティリティーは次のような追加情報の入力を求めます。

```
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>
```

次のステップ

- 署名のために選択した CA に CSR を送信します。または、信頼できるネットワーク内での内部使用シナリオでは、署名にプライベート CA を使用します。詳細は、「プライベート CA を使用した GnuTLS での CSR の証明書の発行」 を参照してください。

検証

1. 次の例のように、証明書の中で人間が判読できる部分が要件と一致することを確認します。

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

関連情報

- **certtool(1)** の man ページ

3.9. プライベート CA を使用した GNUTLS での CSR の証明書の発行

システムが TLS 暗号化通信チャネルを確立できるようにするには、認証局 (CA) が有効な証明書をシステムに提供する必要があります。プライベート CA がある場合は、システムからの証明書署名要求 (CSR) に署名することにより、要求された証明書を作成できます。

前提条件

- プライベート CA が設定済みである。詳細は、「GnuTLS を使用したプライベート CA の作成」 を参照してください。
- CSR を含むファイルがある。CSR の作成例は、「GnuTLS を使用した TLS サーバー証明書の秘密鍵と CSR の作成」 にあります。

手順

1. オプション: 任意のテキストエディターを使用して、次の例のように、証明書に拡張機能を追加するための GnuTLS 設定ファイルを準備します。

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

2. **certtool** ユーティリティーを使用して、CSR に基づいて証明書を作成します。次に例を示します。

```
$ 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>
```

関連情報

- **certtool(1)** の man ページ

第4章 共通システム証明書の使用

共有システム証明書ストレージは、NSS、GnuTLS、OpenSSL、および Java が、システムの証明書アンカーと、拒否リスト情報を取得するデフォルトソースを共有します。トラストストアには、デフォルトで、Mozilla CA リスト (信頼できるリストおよび信頼できないリスト) が含まれています。システムでは、コア Mozilla CA リストを更新したり、別の証明書リストを選択したりできます。

4.1. システム全体のトラストストア

RHEL では、統合されたシステム全体のトラストストアが **/etc/pki/ca-trust/** ディレクトリーおよび **/usr/share/pki/ca-trust-source/** ディレクトリーに置かれています。**/usr/share/pki/ca-trust-source/** のトラスト設定は、**/etc/pki/ca-trust/** の設定よりも低い優先順位で処理されます。

証明書ファイルは、インストールされているサブディレクトリーによって扱われ方が異なります。たとえば、トラストアンカーは **/usr/share/pki/ca-trust-source/anchors/** または **/etc/pki/ca-trust/source/anchors/** ディレクトリーに属します。

注記

階層暗号化システムでは、トラストアンカーとは、他のパーティーが信頼できると想定する権威あるエンティティーです。X.509 アーキテクチャーでは、ルート証明書はトラストチェーンの元となるトラストアンカーです。チェーンの検証を有効にするには、信頼元がまずトラストアンカーにアクセスできる必要があります。

関連情報

- **update-ca-trust(8)** および **trust(1)** の man ページ

4.2. 新しい証明書の追加

新しいソースでシステムのアプリケーションを確認するには、対応する証明書をシステム全体のストアに追加し、**update-ca-trust** コマンドを使用します。

前提条件

- **ca-certificates** パッケージがシステムにインストールされている。

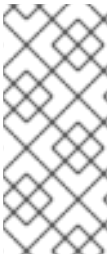
手順

1. システムで信頼されている CA のリストに、シンプルな PEM または DER のファイルフォーマットに含まれる証明書を追加するには、**/usr/share/pki/ca-trust-source/anchors/** ディレクトリーまたは **/etc/pki/ca-trust/source/anchors/** ディレクトリーに証明書ファイルをコピーします。以下に例を示します。

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

2. システム全体のトラストストア設定を更新するには、**update-ca-trust** コマンドを実行します。

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



注記

update-ca-trust を事前に実行しなくても、Firefox ブラウザーは追加された証明書を使用できますが、CA を変更するたびに **update-ca-trust** コマンドを入力してください。Firefox、Chromium および GNOME Web などのブラウザーはファイルをキャッシュするので、ブラウザーのキャッシュをクリアするか、ブラウザーを再起動して、現在のシステム証明書の設定を読み込む必要がある場合があります。

関連情報

- **update-ca-trust(8)** および **trust(1)** の man ページ

4.3. 信頼されているシステム証明書の管理

trust コマンドを使用すると、システム全体の共有トラストストアの証明書を便利な方法で管理できます。

- トラストアンカーのリスト表示、抽出、追加、削除、または変更を行うには、**trust** コマンドを使用します。このコマンドの組み込みヘルプを表示するには、引数を付けずに、または **--help** ディレクティブを付けて実行します。

```
$ 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
```

- すべてのシステムのトラストアンカーおよび証明書のリストを表示するには、**trust list** コマンドを実行します。

```
$ trust list
pkcs11:id=%d2%87%b4%e3%df%37%27%93%55%f6%56%ea%81%e5%36%cc%8c%1e%3
f%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
...
```

- トラストアンカーをシステム全体のトラストストアに保存するには、**trust anchor** サブコマンドを使用し、証明書のパスを指定します。**<path.to/certificate.crt>** を、証明書およびそのファイル名へのパスに置き換えます。

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

- 証明書を削除するには、証明書のパス、または証明書の ID を使用します。

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

関連情報

- **trust** コマンドのすべてのサブコマンドは、以下のような詳細な組み込みヘルプを提供します。

```
$ 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
...
```

関連情報

- **update-ca-trust(8)** および **trust(1)** の man ページ

第5章 TLS の計画および実施

TLS (トランスポート層セキュリティー) は、ネットワーク通信のセキュリティー保護に使用する暗号化プロトコルです。優先する鍵交換プロトコル、認証方法、および暗号化アルゴリズムを設定してシステムのセキュリティー設定を強化する際には、対応するクライアントの範囲が広ければ広いほど、セキュリティーのレベルが低くなることを認識しておく必要があります。反対に、セキュリティー設定を厳密にすると、クライアントとの互換性が制限され、システムからロックアウトされるユーザーが出てくる可能性もあります。可能な限り厳密な設定を目指し、互換性に必要な場合に限り、設定を緩めるようにしてください。

5.1. SSL プロトコルおよび TLS プロトコル

Secure Sockets Layer (SSL) プロトコルは、元々はインターネットを介した安全な通信メカニズムを提供するために、Netscape Corporation により開発されました。その後、このプロトコルは、Internet Engineering Task Force (IETF) により採用され、Transport Layer Security (TLS) に名前が変更になりました。

TLS プロトコルは、アプリケーションプロトコル層と、TCP/IP などの信頼性の高いトランスポート層の間にあります。これは、アプリケーションプロトコルから独立しているため、HTTP、FTP、SMTP など、さまざまなプロトコルの下に階層化できます。

プロトコルのバージョン	推奨される使用方法
SSL v2	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 7 以降、コア暗号ライブラリーから削除されました。
SSL v3	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 8 以降、コア暗号ライブラリーから削除されました。
TLS 1.0	使用は推奨されません。相互運用性を保証した方法では軽減できない既知の問題があり、最新の暗号スイートには対応しません。RHEL 8 では、**LEGACY** システム全体の暗号化ポリシープロファイルでのみ有効です。
TLS 1.1	必要に応じて相互運用性の目的で使用します。最新の暗号スイートには対応しません。RHEL 8 では、**LEGACY** ポリシーでのみ有効になります。
TLS 1.2	最新の AEAD 暗号スイートに対応します。このバージョンは、システム全体のすべての暗号化ポリシーで有効になっていますが、このプロトコルの必須ではない部分に脆弱性があります。また、TLS 1.2 では古いアルゴリズムも使用できます。
TLS 1.3	推奨されるバージョン。TLS 1.3 は、既知の問題があるオプションを取り除き、より多くのネゴシエーションハンドシェイクを暗号化することでプライバシーを強化し、最新の暗号アルゴリズムをより効果的に使用することで速度を速めることができます。TLS 1.3 は、システム全体のすべての暗号化ポリシーでも有効になっています。

関連情報

- IETF: The Transport Layer Security (TLS) Protocol Version 1.3

5.2. RHEL 8 における TLS のセキュリティー上の検討事項

RHEL 8 では、システム全体の暗号化ポリシーにより、暗号化に関する検討事項が大幅に簡素化されています。**DEFAULT** 暗号化ポリシーは TLS 1.2 および 1.3 のみを許可します。システムが以前のバージョンの TLS を使用して接続をネゴシエートできるようにするには、アプリケーション内で次の暗号化ポリシーから除外するか、**update-crypto-policies** コマンドで **LEGACY** ポリシーに切り替える必要があります。詳細は、システム全体の暗号化ポリシーの使用 を参照してください。

大概のデプロイメントは、RHEL 8 に含まれるライブラリーが提供するデフォルト設定で十分に保護されます。TLS 実装は、可能な場合は、安全なアルゴリズムを使用する一方で、レガシーなクライアントまたはサーバーとの間の接続は妨げません。セキュリティーが保護されたアルゴリズムまたはプロトコルに対応しないレガシーなクライアントまたはサーバーの接続が期待できないまたは許可されない場合に、厳密なセキュリティー要件の環境で、強化設定を適用します。

TLS 設定を強化する最も簡単な方法は、**update-crypto-policies --set FUTURE** コマンドを実行して、システム全体の暗号化ポリシーレベルを **FUTURE** に切り替えます。

警告

LEGACY 暗号化ポリシーで無効にされているアルゴリズムは、Red Hat の RHEL 8 セキュリティーのビジョンに準拠しておらず、それらのセキュリティープロパティーは信頼できません。これらのアルゴリズムを再度有効化するのではなく、使用しないようにすることを検討してください。たとえば、古いハードウェアとの相互運用性のためにそれらを再度有効化することを決めた場合は、それらを安全でないものとして扱い、ネットワークの相互作用を個別のネットワークセグメントに分離するなどの追加の保護手段を適用します。パブリックネットワーク全体では使用しないでください。

RHEL システム全体の暗号化ポリシーに従わない場合、またはセットアップに適したカスタム暗号化ポリシーを作成する場合は、カスタム設定で必要なプロトコル、暗号スイート、および鍵の長さについて、以下の推奨事項を使用します。

5.2.1. プロトコル

最新バージョンの TLS は、最高のセキュリティーメカニズムを提供します。古いバージョンの TLS に対応しないといけないような特別な事態がない限り、システムは、TLS バージョン 1.2 以上を使用して接続をネゴシエートできるようにしてください。

RHEL 8 は TLS バージョン 1.3 をサポートしていますが、このプロトコルのすべての機能が RHEL 8 コンポーネントで完全にサポートされているわけではない点に注意してください。たとえば、接続レイテンシーを短縮する 0-RTT (Zero Round Trip Time) 機能は、Apache Web サーバーではまだ完全にはサポートされていません。

5.2.2. 暗号化スイート

旧式で、安全ではない暗号化スイートではなく、最近の、より安全なものを使用してください。暗号化スイートの eNULL および aNULL は、暗号化や認証を提供しないため、常に無効にしてください。RC4 や HMAC-MD5 をベースとした暗号化スイートには深刻な欠陥があるため、可能な場合はこれも無効にしてください。いわゆるエクスポート暗号化スイートも同様です。エクスポート暗号化スイートは意図的に弱くなっているため、侵入が容易になっています。

128 ビット未満のセキュリティーしか提供しない暗号化スイートでは直ちにセキュリティーが保護され

なくなるというわけではありませんが、使用できる期間が短いため考慮すべきではありません。アルゴリズムが 128 ビット以上のセキュリティーを使用している場合は、少なくとも数年間は解読不可能であることが期待されているため、強く推奨されます。3DES 暗号は 168 ビットを使用していると言われていますが、実際に提供されているのは 112 ビットのセキュリティーであることに注意してください。

サーバーの鍵が危険にさらされた場合でも、暗号化したデータの機密性を保証する (完全な) 前方秘匿性 (PFS) に対応する暗号スイートを常に優先します。ここでは、速い RSA 鍵交換は除外されますが、ECDHE および DHE は使用できます。この 2 つを比べると、ECDHE の方が速いため推奨されます。

また、AES-GCM などの AEAD 暗号は、パディングオラクル攻撃の影響は受けないため、CBC モード暗号よりも推奨されます。さらに、多くの場合、特にハードウェアに AES 用の暗号化アクセラレーターがある場合、AES-GCM は CBC モードの AES よりも高速です。

ECDSA 証明書で ECDHE 鍵交換を使用すると、トランザクションは純粋な RSA 鍵交換よりもさらに高速になります。レガシークライアントに対応するため、サーバーには証明書と鍵のペアを 2 つ (新しいクライアント用の ECDSA 鍵と、レガシー用の RSA 鍵) インストールできます。

5.2.3. 公開鍵の長さ

RSA 鍵を使用する際は、SHA-256 以上で署名され、鍵の長さが 3072 ビット以上のものが常に推奨されます (これは、実際に 128 ビットであるセキュリティーに対して十分な大きさです)。

警告

システムのセキュリティー強度は、チェーンの中の最も弱いリンクが示すものと同じになります。たとえば、強力な暗号化だけではすぐれたセキュリティーは保証されません。鍵と証明書も同様に重要で、認証機関 (CA) が鍵の署名に使用するハッシュ機能と鍵もまた重要になります。

関連情報

- System-wide crypto policies in RHEL 8 .
- **update-crypto-policies(8)** の man ページ。

5.3. アプリケーションで TLS 設定の強化

RHEL では、システム全体の暗号化ポリシー は、暗号化ライブラリーを使用するアプリケーションが、既知の安全でないプロトコル、暗号化、またはアルゴリズムを許可しないようにするための便利な方法を提供します。

暗号化設定をカスタマイズして、TLS 関連の設定を強化する場合は、このセクションで説明する暗号化設定オプションを使用して、必要最小量でシステム全体の暗号化ポリシーを上書きできます。

いずれの設定を選択しても、サーバーアプリケーションが **サーバー側が指定した順序** で暗号を利用することを確認し、使用される暗号化スイートの選択がサーバーでの設定順に行われるように設定してください。

5.3.1. TLS を使用するように Apache HTTP サーバーを設定

Apache HTTP Server は、TLS のニーズに **OpenSSL** ライブラリーおよび **NSS** ライブラリーの両方を使用できます。RHEL 8 では、mod_ss パッケージで **mod_ssl** 機能が提供されます。

```
# yum install mod_ssl
```

mod_ssl パッケージは、**/etc/httpd/conf.d/ssl.conf** 設定ファイルをインストールします。これは、**Apache HTTP Server** の TLS 関連の設定を変更するのに使用できます。

httpd-manual パッケージをインストールして、TLS 設定を含む **Apache HTTP Server** の完全ドキュメントを取得します。**/etc/httpd/conf.d/ssl.conf** 設定ファイルで利用可能なディレクティブの詳細は、**/usr/share/httpd/manual/mod/mod_ssl.html** を参照してください。さまざまな設定の例は、**/usr/share/httpd/manual/ssl/ssl_howto.html** ファイルに記載されています。

/etc/httpd/conf.d/ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

SSLProtocol
: このディレクティブを使用して、許可する TLS または SSL のバージョンを指定します。

SSLCipherSuite
: 優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

SSLHonorCipherOrder
: コメントを解除して、このディレクティブを **on** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、TLS 1.2 プロトコルおよび 1.3 プロトコルだけを使用する場合は、以下を実行します。

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

詳細は、Deploying different types of servers の Configuring TLS encryption on an Apache HTTP Server の章を参照してください。

5.3.2. TLS を使用するように Nginx HTTP およびプロキシーサーバーを設定

Nginx で TLS 1.3 サポートを有効にするには、**/etc/nginx/nginx.conf** 設定ファイルの **server** セクションで、**ssl_protocols** オプションに **TLSv1.3** 値を追加します。

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

詳細は、Deploying different types of servers の Adding TLS encryption to an Nginx web server の章を参照してください。

5.3.3. TLS を使用するように Dovecot メールサーバーを設定

Dovecot メールサーバーのインストールが TLS を使用するように設定するには、**/etc/dovecot/conf.d/10-ssl.conf** 設定ファイルを修正します。このファイルで利用可能な基本的な

設定ディレクティブの一部は、**/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt** ファイルで説明されています。このファイルは **Dovecot** の標準インストールに含まれています。

/etc/dovecot/conf.d/10-ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

ssl_protocols

このディレクティブを使用して、許可または無効にする TLS または SSL のバージョンを指定します。

ssl_cipher_list

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

ssl_prefer_server_ciphers

コメントを解除して、このディレクティブを **yes** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、**/etc/dovecot/conf.d/10-ssl.conf** 内の次の行が、TLS 1.1 以降だけを許可します。

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

関連情報

- Deploying different types of servers on RHEL 8
- **config(5)** および **ciphers(1)** の man ページ
- Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
- Mozilla SSL Configuration Generator
- SSL Server Test

第6章 IPSEC を使用した VPN の設定

RHEL 8 では、仮想プライベートネットワーク (VPN) は **IPsec** プロトコルを使用して設定できます。これは、**Libreswan** アプリケーションによりサポートされます。

6.1. IPSEC VPN 実装としての LIBRESWAN

RHEL では、仮想プライベートネットワーク (VPN) は IPsec プロトコルを使用して設定できます。これは、Libreswan アプリケーションによりサポートされます。Libreswan は、Openswan アプリケーションの延長であり、Openswan ドキュメントの多くの例は Libreswan でも利用できます。

VPN の IPsec プロトコルは、IKE (Internet Key Exchange) プロトコルを使用して設定されます。IPsec と IKE は同義語です。IPsec VPN は、IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN、または IKE/IPsec VPN とも呼ばれます。Layer 2 Tunneling Protocol (L2TP) も使用する IPsec VPN のバリアントは、通常、L2TP/IPsec VPN と呼ばれ、**optional** のリポジトリーによって提供される **xl2tpd** パッケージが必要です。

Libreswan は、オープンソースのユーザー空間の IKE 実装です。IKE v1 および v2 は、ユーザーレベルのデーモンとして実装されます。IKE プロトコルも暗号化されています。IPsec プロトコルは Linux カーネルで実装され、Libreswan は、VPN トンネル設定を追加および削除するようにカーネルを設定します。

IKE プロトコルは、UDP ポート 500 および 4500 を使用します。IPsec プロトコルは、以下の 2 つのプロトコルで設定されます。

- 暗号セキュリティーペイロード (ESP) (プロトコル番号が 50)
- 認証ヘッダー (AH) (プロトコル番号 51)

AH プロトコルの使用は推奨されていません。AH のユーザーは、null 暗号化で ESP に移行することが推奨されます。

IPsec プロトコルは、以下の 2 つの操作モードを提供します。

- トンネルモード (デフォルト)
- トランスポートモード

IKE を使用せずに IPsec を使用してカーネルを設定できます。これは、**手動キーリング** と呼ばれます。また、**ip xfrm** コマンドを使用して手動キーを設定できますが、これはセキュリティー上の理由からは強く推奨されません。Libreswan は、Netlink インターフェイスを使用して Linux カーネルと通信します。カーネルはパケットの暗号化と復号化を実行します。

Libreswan は、ネットワークセキュリティーサービス (NSS) 暗号化ライブラリーを使用します。NSS は、**連邦情報処理標準** (**FIPS**) の公開文書 140-2 での使用が認定されています。

重要

Libreswan および Linux カーネルが実装する IKE/IPsec の VPN は、RHEL で使用することが推奨される唯一の VPN 技術です。その他の VPN 技術は、そのリスクを理解せずに使用しないでください。

RHEL では、Libreswan はデフォルトで **システム全体の暗号化ポリシー** に従います。これにより、Libreswan は、デフォルトのプロトコルとして IKEv2 を含む現在の脅威モデルに対して安全な設定を使用するようになります。詳細は、Using system-wide crypto policies を参照してください。

IKE/IPsec はピアツーピアプロトコルであるため、Libreswan では、ソースおよび宛先、またはサーバーおよびクライアントという用語を使用しません。終了点 (ホスト) を参照する場合は、代わりに左と右という用語を使用します。これにより、ほとんどの場合、両方の終了点で同じ設定も使用できます。ただし、管理者は通常、ローカルホストに左を使用し、リモートホストに右を使用します。

leftid と **rightid** オプションは、認証プロセス内の各ホストの識別として機能します。詳細は、man ページの **ipsec.conf(5)** を参照してください。

6.2. LIBRESWAN の認証方法

Libreswan は複数の認証方法をサポートしますが、それぞれは異なるシナリオとなっています。

事前共有キー (PSK)

事前共有キー (PSK) は、最も簡単な認証メソッドです。セキュリティー上の理由から、PSK は 64 文字未満は使用しないでください。FIPS モードでは、PSK は、使用される整合性アルゴリズムに応じて、、最低強度の要件に準拠する必要があります。**authby=secret** 接続を使用して PSK を設定できます。

Raw RSA 鍵

Raw RSA 鍵 は、静的なホスト間またはサブネット間の IPsec 設定で一般的に使用されます。各ホストは、他のすべてのホストのパブリック RSA 鍵を使用して手動で設定され、Libreswan はホストの各ペア間で IPsec トンネルを設定します。この方法は、多数のホストでは適切にスケーリングされません。

ipsec newhostkey コマンドを使用して、ホストで Raw RSA 鍵を生成できます。 **ipsec showhostkey** コマンドを使用して、生成された鍵をリスト表示できます。**leftrsasigkey=** の行は、CKA ID キーを使用する接続設定に必要です。Raw RSA 鍵に **authby=rsasig** 接続オプションを使用します。

X.509 証明書

X.509 証明書 は、共通の IPsec ゲートウェイに接続するホストが含まれる大規模なデプロイメントに一般的に使用されます。中央の **認証局** (CA) は、ホストまたはユーザーの RSA 証明書に署名します。この中央 CA は、個別のホストまたはユーザーの取り消しを含む、信頼のリレーを行います。

たとえば、**openssl** コマンドおよび NSS **certutil** コマンドを使用して X.509 証明書を生成できます。Libreswan は、**leftcert=** 設定オプションの証明書のニックネームを使用して NSS データベースからユーザー証明書を読み取るため、証明書の作成時にニックネームを指定します。

カスタム CA 証明書を使用する場合は、これを Network Security Services(NSS) データベースにインポートする必要があります。**ipsec import** コマンドを使用して、PKCS #12 形式の証明書を Libreswan NSS データベースにインポートできます。

警告

Libreswan は、section 3.1 of RFC 4945 で説明されているように、すべてのピア証明書のサブジェクト代替名 (SAN) としてインターネット鍵 Exchange(IKE) ピア ID を必要とします。**require-id-on-certificated=** オプションを変更してこのチェックを無効にすると、システムが中間者攻撃に対して脆弱になる可能性があります。

SHA-1 および SHA-2 で RSA を使用した X.509 証明書に基づく認証に **authby=rsasig** 接続オプションを使用します。**authby=** を **ecdsa** に設定し、**authby=rsa-sha2** を介した SHA-2 による RSA Probabilistic Signature Scheme (RSASSA-PSS) デジタル署名ベースの認証を設定することにより、

SHA-2 を使用する ECDSA デジタル署名に対してさらに制限することができます。デフォルト値は **authby=rsasig,ecdsa** です。

証明書と **authby=** 署名メソッドが一致する必要があります。これにより、相互運用性が向上し、1 つのデジタル署名システムでの認証が維持されます。

NULL 認証

null 認証 は、認証なしでメッシュの暗号化を取得するために使用されます。これは、パッシブ攻撃は防ぎますが、アクティブ攻撃は防ぎません。ただし、IKEv2 は非対称認証メソッドを許可するため、NULL 認証はインターネットスケールのオポチュニスティック IPsec にも使用できます。このモデルでは、クライアントはサーバーを認証しますが、サーバーはクライアントを認証しません。このモデルは、TLS を使用して Web サイトのセキュリティーを保護するのと似ています。NULL 認証に **authby=null** を使用します。

量子コンピューターに対する保護

上記の認証方法に加えて、**Post-quantum Pre-shared Key** (PPK) メソッドを使用して、量子コンピューターによる潜在的な攻撃から保護することができます。個々のクライアントまたはクライアントグループは、帯域外で設定された事前共有鍵に対応する PPK ID を指定することにより、独自の PPK を使用できます。

事前共有キーで IKEv1 を使用すると、量子攻撃者から保護されます。IKEv2 の再設計は、この保護をネイティブに提供しません。Libreswan は、**Post-quantum Pre-shared Key** (PPK) を使用して、量子攻撃から IKEv2 接続を保護します。

任意の PPK 対応を有効にする場合は、接続定義に **ppk=yes** を追加します。PPK が必要な場合は **ppk=insist** を追加します。次に、各クライアントには、帯域外で通信する (および可能であれば量子攻撃に対して安全な) シークレット値を持つ PPK ID を付与できます。PPK はランダム性において非常に強力で、辞書の単語に基づいていません。PPK ID と PPK データは、次のように **ipsec.secrets** ファイルに保存されます。

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

PPKS オプションは、静的な PPK を参照します。実験的な関数は、ワンタイムパッドに基づいた動的 PPK を使用します。各接続では、ワンタイムパッドの新しい部分が PPK として使用されます。これを使用すると、ファイル内の動的な PPK の部分がゼロで上書きされ、再利用を防ぐことができます。複数のタイムパッドマテリアルが残っていないと、接続は失敗します。詳細は、man ページの **ipsec.secrets(5)** を参照してください。

警告

動的 PPK の実装はサポート対象外のテクノロジープレビューとして提供されます。注意して使用してください。

6.3. LIBRESWAN のインストール

Libreswan IPsec/IKE 実装を通じて VPN を設定する前に、対応するパッケージをインストールし、**ipsec** サービスを開始して、ファイアウォールでサービスを許可する必要があります。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

2. Libreswan を再インストールする場合は、古いデータベースファイルを削除し、新しいデータベースを作成します。

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
# ipsec initnss
```

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

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

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

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

6.4. ホスト間の VPN の作成

raw RSA キーによる認証を使用して、**左** および **右** と呼ばれる 2 つのホスト間に、ホストツーホスト IPsec VPN を作成するように Libreswan を設定できます。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。

手順

1. 各ホストで Raw RSA 鍵ペアを生成します。

```
# ipsec newhostkey
```

2. 前の手順で生成した鍵の **ckaid** を返します。**左** で次のコマンドを実行して、その **ckaid** を使用します。以下に例を示します。

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

上のコマンドの出力により、設定に必要な **leftrsasigkey=** 行が生成されます。次のホスト (**右**) でも同じ操作を行います。

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

3. **/etc/ipsec.d/** ディレクトリーで、新しい **my_host-to-host.conf** ファイルを作成します。上の手順の **ipsec showhostkey** コマンドの出力から、RSA ホストの鍵を新規ファイルに書き込みます。以下に例を示します。

```
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. 鍵をインポートしたら、**ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

5. 接続を読み込みます。

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

6. トンネルを確立します。

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

7. **ipsec** サービスの開始時に自動的にトンネルを開始するには、以下の行を接続定義に追加します。

```
auto=start
```

6.5. サイト間 VPN の設定

2 つのネットワークを結合してサイト間の IPsec VPN を作成する場合は、その 2 つのホスト間の IPsec トンネルを作成します。これにより、ホストは終了点として動作し、1 つまたは複数のサブネットからのトラフィックが通過できるように設定されます。したがって、ホストを、ネットワークのリモート部分にゲートウェイとして見なすことができます。

サイト間の VPN の設定は、設定ファイル内で複数のネットワークまたはサブネットを指定する必要がある点のみが、ホスト間の VPN とは異なります。

前提条件

- ホスト間の VPN が設定されている。

手順

1. ホスト間の VPN の設定が含まれるファイルを、新規ファイルにコピーします。以下に例を示します。

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

2. 上の手順で作成したファイルに、サブネット設定を追加します。以下に例を示します。

```
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
```

6.6. リモートアクセスの VPN の設定

ロードウォーリアーとは、モバイルクライアントと動的に割り当てられた IP アドレスを使用する移動するユーザーのことです。モバイルクライアントは、X.509 証明書を使用して認証します。

以下の例では、**IKEv2** の設定を示しています。**IKEv1** XAUTH プロトコルは使用していません。

サーバー上では以下の設定になります。

```
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
```

ロードウォーリアーのデバイスであるモバイルクライアントでは、上記の設定に多少変更を加えて使用します。

```
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
```

6.7. メッシュ VPN の設定

any-to-any VPN とも呼ばれるメッシュ VPN ネットワークは、全ノードが IPsec を使用して通信するネットワークです。この設定では、IPsec を使用できないノードの例外が許可されます。メッシュの VPN ネットワークは、以下のいずれかの方法で設定できます。

- IPSec を必要とする。
- IPsec を優先するが、平文通信へのフォールバックを可能にする。

ノード間の認証は、X.509 証明書または DNSSEC (DNS Security Extensions) を基にできます。

注記

これらの接続は通常の Libreswan 設定であるため、**オポチュニスティック IPsec** に通常の IKEv2 認証方法を使用できます。ただし、**right=%opportunisticgroup** エントリーで定義されるオポチュニスティック IPsec を除きます。一般的な認証方法は、一般に共有される認証局 (CA) を使用して、X.509 証明書に基づいてホストを相互に認証させる方法です。クラウドデプロイメントでは通常、標準の手順の一部として、クラウド内の各ノードに証明書を発行します。

1 つのホストが侵害されると、グループの PSK シークレットも侵害されるため、PreSharedKey (PSK) 認証は使用しないでください。

NULL 認証を使用すると、認証なしでノード間に暗号化をデプロイできます。これを使用した場合、受動的な攻撃者からのみ保護されます。

以下の手順では、X.509 証明書を使用します。この証明書は、Dogtag Certificate System などの任意の種類の CA 管理システムを使用して生成できます。Dogtag は、各ノードの証明書が PKCS #12 形式 (.p12 ファイル) で利用可能であることを前提としています。これには、秘密鍵、ノード証明書、およびその他のノードの X.509 証明書を検証するのに使用されるルート CA 証明書が含まれます。

各ノードでは、その X.509 証明書を除いて、同じ設定を使用します。これにより、ネットワーク内で既存ノードを再設定せずに、新規ノードを追加できます。PKCS #12 ファイルには分かりやすい名前が必要であるため、名前には node を使用します。これにより、すべてのノードに対して、この名前を参照する設定ファイルが同一になります。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。
- 新しい NSS データベースが初期化されている。

1. すでに古い NSS データベースがある場合は、古いデータベースファイルを削除します。

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
```

2. 次のコマンドを使用して、新しいデータベースを初期化できます。

```
# ipsec initnss
```

手順

1. 各ノードで PKCS #12 ファイルをインポートします。この手順では、PKCS #12 ファイルの生成に使用するパスワードが必要になります。

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

2. **IPsec required** (private)、**IPsec optional** (private-or-clear)、および **No IPsec** (clear) プロファイルに、以下のような 3 つの接続定義を作成します。

```
# 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** 変数にはいくつかのオプションがあります。

ondemand 接続オプションは、IPsec 接続を開始するオポチュニスティック IPsec や、常にアクティブにする必要のない明示的に設定した接続に使用できます。このオプションは、カーネル内にトラップ XFRM ポリシーを設定し、そのポリシーに一致する最初のパケットを受信したときに IPsec 接続を開始できるようにします。

オポチュニスティック IPsec を使用する場合も、明示的に設定した接続を使用する場合も、次のオプションを使用すると、IPsec 接続を効果的に設定および管理できます。

add オプション
: 接続設定をロードし、リモート開始に応答できるように準備します。ただし、接続はローカル側から自動的に開始されません。コマンド **ipsec auto --up** を使用して、IPsec 接続を手動で開始できます。

start オプション
: 接続設定をロードし、リモート開始に応答できるように準備します。さらに、リモートピアへの接続を即座に開始します。このオプションは、永続的かつ常にアクティブな接続に使用できます。

(2) **leftid** 変数と **rightid** 変数は、IPsec トンネル接続の右チャネルと左チャネルを識別します。これらの変数を使用して、ローカル IP アドレスの値、またはローカル証明書のサブジェクト DN を取得できます (設定している場合)。

3 **leftcert** 変数は、使用する NSS データベースのニックネームを定義します。

3. ネットワークの IP アドレスを対応するカテゴリーに追加します。たとえば、すべてのノードが 10.15.0.0/16 ネットワーク内に存在し、すべてのノードで IPsec 暗号化を使用する必要がある場合は、次のコマンドを実行します。

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

4. 特定のノード (10.15.34.0/24 など) を IPsec の有無にかかわらず機能させるには、そのノードを private-or-clear グループに追加します。

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

5. ホストを、10.15.1.2 など、IPsec の機能がない clear グループに定義する場合は、次のコマンドを実行します。

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

/etc/ipsec.d/policies ディレクトリーのファイルは、各新規ノードのテンプレートから作成することも、Puppet または Ansible を使用してプロビジョニングすることもできます。

すべてのノードでは、例外のリストが同じか、異なるトラフィックフローが期待される点に注意してください。したがって、あるノードで IPsec が必要になり、別のノードで IPsec を使用できないために、ノード間の通信ができない場合もあります。

6. ノードを再起動して、設定したメッシュに追加します。

```
# systemctl restart ipsec
```

検証

2 つのノード間に IPsec トンネルを開くことで手順を確認できます。

1. **ping** コマンドを使用して IPsec トンネルを開きます。

```
# ping <nodeYYY>
```

2. インポートされた証明書を含む NSS データベースを表示します。

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

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

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

3. ノードが開いたトンネルを確認します。

```
# 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'
```

関連情報

- **ipsec.conf(5)** man ページ。
- **authby** 変数の詳細は、6.2.Libreswan の認証方法 を参照してください。

6.8. FIPS 準拠の IPSEC VPN のデプロイメント

この手順を使用して、Libreswan に基づく FIPS 準拠の IPsec VPN ソリューションをデプロイします。次の手順では、FIPS モードの Libreswan で使用可能な暗号化アルゴリズムと無効になっている暗号化アルゴリズムを識別することもできます。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

2. Libreswan を再インストールする場合は、古い NSS データベースを削除します。

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
```

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

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

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

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

5. システムを FIPS モードに切り替えます。

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

6. システムを再起動して、カーネルを FIPS モードに切り替えます。

```
# reboot
```

検証

1. Libreswan が FIPS モードで実行していることを確認するには、次のコマンドを実行します。

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

2. または、**systemd** ジャーナルで **ipsec** ユニットのエントリーを確認します。

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

3. FIPS モードで使用可能なアルゴリズムを表示するには、次のコマンドを実行します。

```
# ipsec pluto --selftest 2>&1 | head -11
FIPS Product: YES
FIPS Kernel: YES
FIPS Mode: YES
NSS DB directory: sql:/etc/ipsec.d
Initializing NSS
Opening NSS database "sql:/etc/ipsec.d" read-only
NSS initialized
NSS crypto library initialized
FIPS HMAC integrity support [enabled]
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity verification self-test passed
```

4. FIPS モードで無効化されたアルゴリズムをクエリーするには、次のコマンドを実行します。

```
# 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 SERPENT_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_SSH 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 MODP1024 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. FIPS モードで許可されているすべてのアルゴリズムと暗号のリストを表示するには、次のコマンドを実行します。

```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
{256,192,*128}  aes_ccm, aes_ccm_c
{256,192,*128}  aes_ccm_b
{256,192,*128}  aes_ccm_a
[*192]  3des
{256,192,*128}  aes_gcm, aes_gcm_c
{256,192,*128}  aes_gcm_b
{256,192,*128}  aes_gcm_a
```

```
{256,192,*128}  aesctr
{256,192,*128}  aes
{256,192,*128}  aes_gmac
sha, sha1, sha1_96, hmac_sha1
sha512, sha2_512, sha2_512_256, hmac_sha2_512
sha384, sha2_384, sha2_384_192, hmac_sha2_384
sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
null, dh0
dh14
dh15
dh16
dh17
dh18
ecp_256, ecp256
ecp_384, ecp384
ecp_521, ecp521
```

関連情報

- Using system-wide cryptographic policies .

6.9. パスワードによる IPSEC NSS データベースの保護

デフォルトでは、IPsec サービスは、初回起動時に空のパスワードを使用して Network Security Services (NSS) データベースを作成します。パスワード保護を追加するには、以下の手順を実行します。

注記

以前の RHEL 6.6 リリースでは、NSS 暗号化ライブラリーが FIPS 140-2 Level 2 標準で認定されているため、FIPS 140-2 要件を満たすパスワードで IPsec NSS データベースを保護する必要がありました。RHEL 8 では、この規格の NIST 認定 NSS がこの規格のレベル 1 に認定されており、このステータスではデータベースのパスワード保護は必要ありません。

前提条件

- **/etc/ipsec.d/** ディレクトリーに NSS データベースファイルが含まれます。

手順

1. Libreswan の **NSS** データベースのパスワード保護を有効にします。

```
# certutil -N -d sql:/etc/ipsec.d
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. 前の手順で設定したパスワードを追加した **/etc/ipsec.d/nsspassword** ファイルを作成します。以下に例を示します。

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:MyStrongPasswordHere
```

nsspassword ファイルは以下の構文を使用することに注意してください。

```
token_1_name:the_password
token_2_name:the_password
```

デフォルトの NSS ソフトウェアトークンは **NSS Certificate DB** です。システムが FIPS モードで実行し場合は、トークンの名前が **NSS FIPS 140-2 Certificate DB** になります。

3. 選択したシナリオに応じて、**nsspassword** ファイルの完了後に **ipsec** サービスを起動または再起動します。

```
# systemctl restart ipsec
```

検証

1. NSS データベースに空でないパスワードを追加した後に、**ipsec** サービスが実行中であることを確認します。

```
# 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. 必要に応じて、初期化の成功を示すエントリーが **Journal** ログに含まれていることを確認します。

```
# journalctl -u ipsec
...
pluto[6214]: Initializing NSS using read-write database "sql:/etc/ipsec.d"
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
...
```

関連情報

- **certutil(1)** man ページ。
- Government Standards ナレッジベースアーティクル

6.10. TCP を使用するように IPSEC VPN を設定

Libreswan は、RFC 8229 で説明されているように、IKE パケットおよび IPsec パケットの TCP カプセル化に対応します。この機能により、UDP 経由でトラフィックが転送されないように、IPsec VPN をネットワークに確立し、セキュリティーのペイロード (ESP) を強化できます。フォールバックまたはメ

インの VPN トランスポートプロトコルとして TCP を使用するように VPN サーバーおよびクライアントを設定できます。TCP カプセル化にはパフォーマンスコストが大きくなるため、UDP がシナリオで永続的にブロックされている場合に限り、TCP を主な VPN プロトコルとして使用してください。

前提条件

- リモートアクセス VPN が設定されている。

手順

1. **config setup** セクションの **/etc/ipsec.conf** ファイルに以下のオプションを追加します。

```
listen-tcp=yes
```

2. UDP で最初の試行に失敗した場合に TCP カプセル化をフォールバックオプションとして使用するには、クライアントの接続定義に以下の 2 つのオプションを追加します。

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

または、UDP を永続的にブロックしている場合は、クライアントの接続設定で以下のオプションを使用します。

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

関連情報

- IETF RFC 8229: IKE および IPsec Packets の TCP Encapsulation 。

6.11. IPSEC 接続を高速化するために、ESP ハードウェアオフロードの自動検出と使用を設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、Ethernet で IPsec 接続が加速します。デフォルトでは、Libreswan は、ハードウェアがこの機能に対応しているかどうかを検出するため、ESP ハードウェアのオフロードを有効にします。機能が無効になっているか、明示的に有効になっている場合は、自動検出に戻すことができます。

前提条件

- ネットワークカードは、ESP ハードウェアオフロードに対応します。
- ネットワークドライバーは、ESP ハードウェアのオフロードに対応します。
- IPsec 接続が設定され、動作する。

手順

1. ESP ハードウェアオフロードサポートの自動検出を使用する接続の **/etc/ipsec.d/** ディレクトリーにある Libreswan 設定ファイルを編集します。
2. 接続の設定で **nic-offload** パラメーターが設定されていないことを確認します。

3. **nic-offload** を削除した場合は、**ipsec** を再起動します。

```
# systemctl restart ipsec
```

検証

ネットワークカードが ESP ハードウェアオフロードサポートに対応している場合は、以下の手順に従って結果を検証します。

1. IPsec 接続が使用するイーサネットデバイスの **tx_ipsec** および **rx_ipsec** カウンターを表示します。

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

2. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

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

3. イーサネットデバイスの **tx_ipsec** および **rx_ipsec** カウンターを再度表示します。

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

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- IPsec を使用した VPN の設定

6.12. IPSEC 接続を加速化するためにボンディングでの ESP ハードウェアオフロードの設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、IPsec 接続が加速します。フェイルオーバーの理由でネットワークボンディングを使用する場合、ESP ハードウェアオフロードを設定する要件と手順は、通常のイーサーネットデバイスを使用する要件と手順とは異なります。たとえば、このシナリオでは、ボンディングでオフロードサポートを有効にし、カーネルはボンディングのポートに設定を適用します。

前提条件

- ボンディングのすべてのネットワークカードが、ESP ハードウェアオフロードをサポートしている。
- ネットワークドライバーが、ボンドデバイスで ESP ハードウェアオフロードに対応している。RHEL では、**ixgbe** ドライバーのみがこの機能をサポートします。
- ボンディングが設定されており動作する。
- ボンディングで **active-backup** モードを使用している。ボンディングドライバーは、この機能の他のモードはサポートしていません。

- IPsec 接続が設定され、動作する。

手順

1. ネットワークボンディングで ESP ハードウェアオフロードのサポートを有効にします。

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

このコマンドにより、**bond0** 接続での ESP ハードウェアオフロードのサポートが有効になります。

2. **bond0** 接続を再度アクティブにします。

```
# nmcli connection up bond0
```

3. ESP ハードウェアオフロードに使用すべき接続の **/etc/ipsec.d/** ディレクトリーにある Libreswan 設定ファイルを編集し、**nic-offload=yes** ステートメントを接続エントリーに追加します。

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

4. **ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

検証

1. ボンディングのアクティブなポートを表示します。

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

2. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを表示します。

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

3. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

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

4. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを再度表示します。

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

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- ネットワークボンディングの設定
- IPsec を使用した VPN の設定

6.13. システム全体の暗号化ポリシーをオプトアウトする IPSEC 接続の設定

接続向けのシステム全体の暗号化ポリシーのオーバーライド

RHEL のシステム全体の暗号化ポリシーでは、**%default** と呼ばれる特別な接続が作成されます。この接続には、**ikev2** オプション、**esp** オプション、および **ike** オプションのデフォルト値が含まれます。ただし、接続設定ファイルに上記のオプションを指定すると、デフォルト値を上書きできます。

たとえば、次の設定では、AES および SHA-1 または SHA-2 で IKEv1 を使用し、AES-GCM または AES-CBC で IPsec (ESP) を使用する接続が可能です。

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

AES-GCM は IPsec (ESP) および IKEv2 で利用できますが、IKEv1 では利用できません。

全接続向けのシステム全体の暗号化ポリシーの無効化

すべての IPsec 接続のシステム全体の暗号化ポリシーを無効にするには、**/etc/ipsec.conf** ファイルで次の行をコメントアウトします。

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

次に、接続設定ファイルに **ikev2=never** オプションを追加してください。

関連情報

- Using system-wide cryptographic policies .

6.14. IPSEC VPN 設定のトラブルシューティング

IPsec VPN 設定に関連する問題は主に、一般的な理由が原因で発生する可能性が高くなっています。このような問題が発生した場合は、問題の原因が以下のシナリオのいずれかに該当するかを確認して、対応するソリューションを適用します。

基本的な接続のトラブルシューティング

VPN 接続関連の問題の多くは、管理者が不適当な設定オプションを指定してエンドポイントを設定した新しいデプロイメントで発生します。また、互換性のない値が新たに実装された場合に、機能していた設定が突然動作が停止する可能性があります。管理者が設定を変更した場合など、このような結果になることがあります。また、管理者が暗号化アルゴリズムなど、特定のオプションに異なるデフォルト値を使用して、ファームウェアまたはパッケージの更新をインストールした場合などです。

IPsec VPN 接続が確立されていることを確認するには、次のコマンドを実行します。

■

```
# 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
```

出力が空の場合や、エントリーで接続名が表示されない場合など、トンネルが破損します。

接続に問題があることを確認するには、以下を実行します。

1. **vpn.example.com** 接続をもう一度読み込みます。

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

2. 次に、VPN 接続を開始します。

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

ファイアウォール関連の問題

最も一般的な問題は、IPSec エンドポイントの 1 つ、またはエンドポイント間にあるルーターにあるファイアウォールで Internet Key Exchange (IKE) パケットがドロップされるという点が挙げられます。

- IKEv2 の場合には、以下の例のような出力は、ファイアウォールに問題があることを示しています。

```
# 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
...
```

- IKEv1 の場合は、最初のコマンドの出力は以下のようになります。

```
# 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
...
```

IPsec の設定に使用される IKE プロトコルは暗号化されているため、**tcpdump** ツールを使用して、トラブルシューティングできるサブセットは一部のみです。ファイアウォールが IKE パケットまたは IPsec パケットをドロップしている場合は、**tcpdump** ユーティリティーを使用して原因を見つけることができます。ただし、**tcpdump** は IPsec VPN 接続に関する他の問題を診断できません。

- **eth0** インターフェイスで VPN および暗号化データすべてのネゴシエーションを取得するには、次のコマンドを実行します。

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

アルゴリズム、プロトコル、およびポリシーが一致しない場合

VPN 接続では、エンドポイントが IKE アルゴリズム、IPsec アルゴリズム、および IP アドレス範囲に一致する必要があります。不一致が発生した場合には接続は失敗します。以下の方法のいずれかを使用して不一致を特定した場合は、アルゴリズム、プロトコル、またはポリシーを調整して修正します。

- リモートエンドポイントが IKE/IPsec を実行していない場合は、そのパケットを示す ICMP パケットが表示されます。以下に例を示します。

```
# 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)]
...
```

- IKE アルゴリズムが一致しない例:

```
# 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
```

- IPsec アルゴリズムが一致しない例:

```
# 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
```

また、IKE バージョンが一致しないと、リモートエンドポイントが応答なしの状態でリクエストをドロップする可能性がありました。これは、すべての IKE パケットをドロップするファイアウォールと同じです。

- IKEv2 (Traffic Selectors - TS) の IP アドレス範囲が一致しない例:

```
# 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 の IP アドレス範囲で一致しない例:

```
# 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
```

- IKEv1 で PreSharedKeys (PSK) を使用する場合には、どちらでも同じ PSK に配置されなければ、IKE メッセージ全体の読み込みができなくなります。

```
# 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
```

- IKEv2 では、 mismatched-PSK エラーが原因で AUTHENTICATION_FAILED メッセージが表示されます。

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

最大伝送単位 (MTU)

ファイアウォールが IKE または IPSec パケットをブロックする以外で、ネットワークの問題の原因として、暗号化パケットのパケットサイズの増加が最も一般的です。ネットワークハードウェアは、最大伝送単位 (MTU) を超えるパケットを 1500 バイトなどのサイズに断片化します。多くの場合、断片化されたパケットは失われ、パケットの再アセンブルに失敗します。これにより、小さいサイズのパケットを使用する ping テスト時には機能し、他のトラフィックでは失敗するなど、断続的な問題が発生します。このような場合に、SSH セッションを確立できますが、リモートホストに 'ls -al /usr' コマンドに入力した場合など、すぐにターミナルがフリーズします。

この問題を回避するには、トンネル設定ファイルに **mtu=1400** のオプションを追加して、MTU サイズを縮小します。

または、TCP 接続の場合は、MSS 値を変更する iptables ルールを有効にします。

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

各シナリオで上記のコマンドを使用して問題が解決されない場合は、**set-mss** パラメーターで直接サイズを指定します。

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

ネットワークアドレス変換 (NAT)

IPsec ホストが NAT ルーターとしても機能すると、誤ってパケットが再マッピングされる可能性があります。以下の設定例はこの問題について示しています。

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

- アドレスが 172.16.0.1 のシステムには NAT ルールが 1 つあります。

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

アドレスが 10.0.2.33 のシステムがパケットを 192.168.0.1 に送信する場合に、ルーターは IPsec 暗号化を適用する前にソースを 10.0.2.33 から 172.16.0.1 に変換します。

次に、ソースアドレスが 10.0.2.33 のパケットは **conn myvpn** 設定と一致しなくなるので、IPsec ではこのパケットが暗号化されません。

この問題を解決するには、ルーターのターゲット IPsec サブネット範囲の NAT を除外するルールを挿入します。以下に例を示します。

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

カーネル IPsec サブシステムのバグ

たとえば、バグが原因で IKE ユーザー空間と IPsec カーネルの同期が解除される場合など、カーネル IPsec サブシステムに問題が発生する可能性があります。このような問題がないかを確認するには、以下を実行します。

```
$ cat /proc/net/xfrm_stat
XfrmInError               0
XfrmInBufferError         0
...
```

上記のコマンドの出力でゼロ以外の値が表示されると、問題があることを示しています。この問題が発生した場合は、新しい サポートケース を作成し、1 つ前のコマンドの出力と対応する IKE ログを添付してください。

Libreswan のログ

デフォルトでは、Libreswan は **syslog** プロトコルを使用してログに記録します。**journalctl** コマンドを使用して、IPsec に関連するログエントリーを検索できます。ログへの対応するエントリーは **pluto** IKE デーモンにより送信されるため、以下のように、キーワード pluto を検索します。

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

ipsec サービスのライブログを表示するには、次のコマンドを実行します。

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

ロギングのデフォルトレベルで設定問題が解決しない場合は、**/etc/ipsec.conf** ファイルの **config setup** セクションに **plutodebug=all** オプションを追加してデバッグログを有効にします。

デバッグロギングは多くのエントリーを生成し、**journald** サービスまたは **syslogd** サービスレートのいずれかが **syslog** メッセージを制限する可能性があることに注意してください。完全なログを取得するには、ロギングをファイルにリダイレクトします。**/etc/ipsec.conf** を編集し、**config setup** セクションに **logfile=/var/log/pluto.log** を追加します。

関連情報

- ログファイルを使用した問題のトラブルシューティング

- **tcpdump(8)** および **ipsec.conf(5)** の man ページ
- firewalld の使用および設定

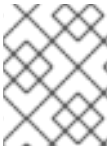
6.15. 関連情報

- **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)**、および **ipsec_rsasigkey(8)** の man ページ
- **/usr/share/doc/libreswan-version/** ディレクトリー
- アップストリームプロジェクトの Web サイト
- The Libreswan プロジェクトの Wiki
- All Libreswan のすべての man ページ
- NIST Special Publication 800-77: Guide to IPsec VPNs

第7章 VPN RHEL システムロールを使用した IPSEC との VPN 接続の設定

vpn システムロールを使用すると、Red Hat Ansible Automation Platform を使用して RHEL システムで VPN 接続を設定できます。これを使用して、ホスト間、ネットワーク間、VPN リモートアクセスサーバー、およびメッシュ設定をセットアップできます。

ホスト間接続の場合、ロールは、必要に応じてキーを生成するなど、デフォルトのパラメーターを使用して、**vpn_connections** のリスト内のホストの各ペア間に VPN トンネルを設定します。または、リストされているすべてのホスト間にオポチュニスティックメッシュ設定を作成するように設定することもできます。このロールは、**hosts** の下にあるホストの名前が Ansible インベントリーで使用されているホストの名前と同じであり、それらの名前を使用してトンネルを設定できることを前提としています。



注記

vpn RHEL システムロールは現在、VPN プロバイダーとして IPsec 実装であ る Libreswan のみをサポートしています。

7.1. VPN システムロールを使用して IPSEC でホスト間 VPN の作成

vpn システムロールを使用して、コントロールノードで Ansible Playbook を実行することにより、ホスト間接続を設定できます。これにより、インベントリーファイルにリストされているすべての管理対象ノードが設定されます。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/vpn-playbook.yml** などの Playbook ファイルを次の内容で作成します。

```
- name: Host to host VPN
  hosts: <managed_node1>, <managed_node2>
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          <managed_node1>:
          <managed_node2>:
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

この Playbook は、システムロールによって自動生成されたキーを使用した事前共有キー認証を

使用して、**<managed_node1>-to-<managed_node2>** の接続を設定します。**vpn_manage_firewall** と **vpn_manage_selinux** は両方とも **true** に設定されているため、**vpn** ロールは **firewall** ロールと **selinux** ロールを使用して、**vpn** ロールが使用するポートを管理します。

2. 必要に応じて、ホストの **vpn_connections** リストに次のセクションを追加して、マネージドホストから、インベントリーファイルに記述されていない外部ホストへの接続を設定します。

```
vpn_connections:
  - hosts:
      <managed_node1>:
      <managed_node2>:
      <external_node>:
        hostname: <192.0.2.2>
```

これにより、2 つの追加の接続 **<managed_node1>-to-<external_node>** および **<managed_node2>-to-<external_node>** が設定されます。

注記

接続はマネージドノードでのみ設定され、外部ノードでは設定されません。

3. 必要に応じて、**vpn_connections** 内の追加セクション (コントロールプレーンやデータプレーンなど) を使用して、管理対象ノードに複数の VPN 接続を指定できます。

```
- name: Multiple VPN
  hosts: <managed_node1>, <managed_node2>
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          <managed_node1>:
            hostname: 192.0.2.0 # IP for the control plane
          <managed_node2>:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          <managed_node1>:
            hostname: 10.0.0.1 # IP for the data plane
          <managed_node2>:
            hostname: 10.0.0.2
```

4. 必要に応じて、設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md** ファイルを参照してください。

5. Playbook の構文を検証します。

```
# ansible-playbook ~/vpn-playbook.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

6. Playbook を実行します。

```
# ansible-playbook ~/vpn-playbook.yml
```

検証

1. マネージドノードで、接続が正常にロードされていることを確認します。

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

<connection_name>を、このノードからの接続の名前 (たとえば、**managed_node1-to-managed_node2**) に置き換えます。

注記

デフォルトでは、ロールは、各システムの観点から作成する接続ごとにわかりやすい名前を生成します。たとえば、**managed_node1** と **managed_node2** との間の接続を作成するときに、**managed_node1** 上のこの接続のわかりやすい名前は **managed_node1-to-managed_node2** ですが、**managed_node2** では、この接続の名前は **managed_node2-to-managed_node1** となります。

2. マネージドノードで、接続が正常に開始されたことを確認します。

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

3. オプション: 接続が正常にロードされない場合は、次のコマンドを入力して手動で接続を追加します。これにより、接続の確立に失敗した理由を示す、より具体的な情報が提供されます。

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

注記

接続のロードおよび開始のプロセスで発生する可能性のあるエラーは、**/var/log/pluto.log** ファイルに報告されます。これらのログは解析が難しいため、代わりに接続を手動で追加して、標準出力からログメッセージを取得してください。

7.2. VPN システムロールを使用して IPSEC でオポチュニスティックメッシュ VPN 接続の作成

vpn システムロールを使用して、コントロールノードで Ansible Playbook を実行することにより、認証に証明書を使用するオポチュニスティックメッシュ VPN 接続を設定できます。これにより、インベントリーファイルにリストされているすべての管理対象ノードが設定されます。

証明書による認証は、Playbook で **auth_method: cert** パラメーターを定義することによって設定されます。**vpn** システムロールは、**/etc/ipsec.d** ディレクトリーで定義されている IPsec ネットワークセキュリティーサービス (NSS) 暗号ライブラリーに必要な証明書が含まれていることを前提としています。デフォルトでは、ノード名が証明書のニックネームとして使用されます。この例では、これは **managed_node1** です。インベントリーで **cert_name** 属性を使用して、さまざまな証明書名を定義できます。

次の手順例では、Ansible Playbook を実行するシステムであるコントロールノードは、両方のマネージドノード (192.0.2.0/24) と同じクラスレスドメイン間ルーティング (CIDR) 番号を共有し、IP アドレスは 192.0.2.7 になります。したがって、コントロールノードは、CIDR 192.0.2.0/24 用に自動的に作成さ

れるプライベートポリシーに該当します。

再生中の SSH 接続の損失を防ぐために、コントロールノードの明確なポリシーがポリシーのリストに含まれています。ポリシーリストには、CIDR がデフォルトと等しい項目もあることに注意してください。これは、この Playbook がデフォルトポリシーのルールを上書きして、private-or-clear ではなく private にするためです。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/mesh-vpn-playbook.yml** などの Playbook ファイルを次の内容で作成します。

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  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
```

注記

vpn_manage_firewall と **vpn_manage_selinux** は両方とも **true** に設定されているため、**vpn** ロールは **firewall** ロールと **selinux** ロールを使用して、**vpn** ロールが使用するポートを管理します。

2. 必要に応じて、設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md** ファイルを参照してください。

3. Playbook の構文を検証します。

```
# ansible-playbook ~/mesh-vpn-playbook.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

4. Playbook を実行します。

```
# ansible-playbook ~/mesh-vpn-playbook.yml
```

7.3. 関連情報

- VPN システムロールで使用するパラメーターの詳細と、**vpn** システムロールに関する追加情報は、**/usr/share/doc/rhel-system-roles/vpn/README.md** ファイルを参照してください。
- **ansible-playbook** コマンドの詳細は、man ページの **ansible-playbook(1)** を参照してください。

第8章 MACSEC を使用した同じ物理ネットワーク内のレイヤー 2 トラフィックの暗号化

MACsec を使用して、2 つのデバイス間の通信を (ポイントツーポイントで) セキュリティー保護できます。たとえば、ブランチオフィスがメトロイーサネット接続を介してセントラルオフィスに接続されている場合、オフィスを接続する 2 つのホストで MACsec を設定して、セキュリティーを強化できます。

Media Access Control Security (MACsec) は、イーサーネットリンクで異なるトラフィックタイプを保護するレイヤー 2 プロトコルです。これには以下が含まれます。

- DHCP (Dynamic Host Configuration Protocol)
- アドレス解決プロトコル (ARP)
- インターネットプロトコルのバージョン 4 / 6 (**IPv4** / **IPv6**)
- TCP や UDP などの IP 経由のトラフィック

MACsec はデフォルトで、LAN 内のすべてのトラフィックを GCM-AES-128 アルゴリズムで暗号化および認証し、事前共有キーを使用して参加者ホスト間の接続を確立します。共有前の鍵を変更する場合は、MACsec を使用するネットワーク内のすべてのホストで NM 設定を更新する必要があります。

MACsec 接続は、親としてイーサネットネットワークカード、VLAN、トンネルデバイスなどのイーサネットデバイスを使用します。暗号化した接続のみを使用して他のホストと通信するように、MACsec デバイスでのみ IP 設定を指定するか、親デバイスに IP 設定を指定することもできます。後者の場合、親デバイスを使用して、暗号化されていない接続と暗号化された接続用の MACsec デバイスで他のホストと通信できます。

MACsec には特別なハードウェアは必要ありません。たとえば、ホストとスイッチの間のトラフィックのみを暗号化する場合を除き、任意のスイッチを使用できます。このシナリオでは、スイッチが MACsec もサポートする必要があります。

つまり、MACsec を設定する方法は 2 つあります。

- ホスト対ホスト
- 他のホストに切り替えるホスト

重要

MACsec は、同じ (物理または仮想) LAN のホスト間でのみ使用することができます。

8.1. NMCLI を使用した MACSEC 接続の設定

nmcli ツールを使用して、MACsec を使用するようにイーサーネットインターフェイスを設定できます。たとえば、イーサネット経由で接続された 2 つのホスト間に MACsec 接続を作成できます。

手順

1. MACsec を設定する最初のホストで:
   - 事前共有鍵の接続アソシエーション鍵 (CAK) と接続アソシエーション鍵名 (CKN) を作成します。

a. 16 バイトの 16 進 CAK を作成します。

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

b. 32 バイトの 16 進 CKN を作成します。

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

2. 両方のホストで、MACsec 接続を介して接続します。

3. MACsec 接続を作成します。

```
# 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
```

前の手順で生成された CAK および CKN を **macsec.mka-cak** および **macsec.mka-ckn** パラメーターで使用します。この値は、MACsec で保護されるネットワーク内のすべてのホストで同じである必要があります。

4. MACsec 接続で IP を設定します。

a. **IPv4** 設定を指定します。たとえば、静的 **IPv4** アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

```
# 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. **IPv6** 設定を指定しますたとえば、静的 **IPv6** アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

```
# 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. 接続をアクティベートします。

```
# nmcli connection up macsec0
```

検証

1. トラフィックが暗号化されていることを確認します。

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

2. オプション: 暗号化されていないトラフィックを表示します。

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

3. MACsec の統計を表示します。

```
# ip macsec show
```

4. integrity-only (encrypt off) および encryption (encrypt on) の各タイプの保護に対して個々のカウンターを表示します。

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

8.2. 関連情報

- MACsec: a different solution to encrypt network traffic ブログ

第9章 FIREWALLD の使用および設定

ファイアウォール は、外部からの不要なトラフィックからマシンを保護する方法です。**ファイアウォールルール** セットを定義することで、ホストマシンへの着信ネットワークトラフィックを制御できます。このようなルールは、着信トラフィックを分類して、拒否または許可するために使用されます。

firewalld は、D-Bus インターフェイスを使用して、動的にカスタマイズできるホストベースのファイアウォールを提供するファイアウォールサービスデーモンです。ルールが変更するたびに、ファイアウォールデーモンを再起動しなくても、ルールの作成、変更、および削除を動的に可能にします。

firewalld は、ゾーンおよびサービスの概念を使用して、トラフィック管理を簡素化します。ゾーンは、事前定義したルールセットです。ネットワークインターフェイスおよびソースをゾーンに割り当てることができます。許可されているトラフィックは、コンピューターが接続するネットワークと、このネットワークが割り当てられているセキュリティーレベルに従います。ファイアウォールサービスは、特定のサービスに着信トラフィックを許可するのに必要なすべての設定を扱う事前定義のルールで、ゾーンに適用されます。

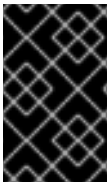
サービスは、ネットワーク接続に 1 つ以上のポートまたはアドレスを使用します。ファイアウォールは、ポートに基づいて接続のフィルターを設定します。サービスに対してネットワークトラフィックを許可するには、そのポートを解放する必要があります。**firewalld** は、明示的に解放されていないポートのトラフィックをすべてブロックします。trusted などのゾーンでは、デフォルトですべてのトラフィックを許可します。

nftables バックエンドを使用した **firewalld** が、**--direct** オプションを使用して、カスタムの **nftables** ルールを **firewalld** に渡すことに対応していないことに注意してください。

9.1. FIREWALLD、NFTABLES、または IPTABLES を使用する場合

以下は、次のユーティリティーのいずれかを使用する必要があるシナリオの概要です。

- **firewalld**: 簡単な firewall のユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。
- **nftables**: **nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。
- **iptables**: Red Hat Enterprise Linux の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。



重要

さまざまなファイアウォール関連サービス (**firewalld**、**nftables**、または **iptables**) が相互に影響を与えないようにするには、RHEL ホストでそのうち 1 つだけを実行し、他のサービスを無効にします。

9.2. ファイアウォールゾーン

firewalld ユーティリティーを使用すると、ネットワーク内のインターフェイスおよびトラフィックに対する信頼レベルに応じて、ネットワークをさまざまなゾーンに分離できます。接続は 1 つのゾーンにしか指定できませんが、そのゾーンは多くのネットワーク接続に使用できます。

firewalld はゾーンに関して厳格な原則に従います。

1. トラフィックは1つのゾーンのみに流入します。
2. トラフィックは1つのゾーンのみから流出します。
3. ゾーンは信頼のレベルを定義します。
4. ゾーン内トラフィック (同じゾーン内) はデフォルトで許可されます。
5. ゾーン間トラフィック (ゾーンからゾーン) はデフォルトで拒否されます。

原則 4 と 5 は原則 3 の結果です。

原則 4 は、ゾーンオプション **--remove-forward** を使用して設定できます。原則 5 は、新しいポリシーを追加することで設定できます。

NetworkManager は、**firewalld** にインターフェイスのゾーンを通知します。次のユーティリティーを使用して、ゾーンをインターフェイスに割り当てることができます。

- **NetworkManager**
- **firewall-config** ユーティリティー
- **firewall-cmd** ユーティリティー
- RHEL Web コンソール

RHEL Web コンソール、**firewall-config**、および **firewall-cmd は、** 適切な **NetworkManager** 設定ファイルのみを編集できます。Web コンソール、**firewall-cmd** または **firewall-config** を使用してインターフェイスのゾーンを変更する場合、リクエストは **NetworkManager** に転送され、**firewalld** では処理されません。

/usr/lib/firewalld/zones/ ディレクトリーには事前定義されたゾーンが保存されており、利用可能なネットワークインターフェイスに即座に適用できます。このファイルは、修正しないと **/etc/firewalld/zones/** ディレクトリーにコピーされません。事前定義したゾーンのデフォルト設定は以下のようになります。

block

- **IPv4** の場合は **icmp-host-prohibited** メッセージ、IPv6 の場合は icmp6-adm-prohibited メッセージで、すべての着信ネットワーク接続が拒否されます。
- システム内から開始したネットワーク接続のみ。

dmz

- パブリックにアクセス可能で、内部ネットワークへのアクセスが制限されている DMZ 内のコンピューター。
- Accept: 選択された着信接続のみ。

drop

適切：着信ネットワークパケットは通知なしでドロップされます。

**許可：発信ネットワーク接続のみ。

external

- マスカレードが有効にされている外部ネットワーク（特にルーター）に適しています。ネットワーク上の他のコンピューターを信頼できない状況。
- Accept: 選択された着信接続のみ。

home

- ネットワーク上の他のコンピューターをほぼ信頼できる自宅の環境。
- Accept: 選択された着信接続のみ。

internal

- ネットワーク上の他のコンピューターをほぼ信頼できる内部ネットワーク。
- Accept: 選択された着信接続のみ。

public

- 適合：ネットワーク上の他のコンピューターを信頼しないパブリックエリア。
- Accept: 選択された着信接続のみ。

trusted

- Accept: すべてのネットワーク接続を許可します。

work

ネットワーク上の他のコンピューターをほぼ信頼できる職場の環境。

- Accept: 選択された着信接続のみ。

このゾーンのいずれかを **デフォルト** ゾーンに設定できます。インターフェイス接続を **NetworkManager** に追加すると、デフォルトゾーンに割り当てられます。インストール時は、**firewalld** のデフォルトゾーンは **public** ゾーンです。デフォルトゾーンは変更できます。

注記

ユーザーがすぐに理解できるように、ネットワークゾーン名は分かりやすい名前にしてください。

セキュリティー問題を回避するために、ニーズおよびリスク評価に合わせて、デフォルトゾーンの設定の見直しを行ったり、不要なサービスを無効にしてください。

関連情報

- **firewalld.zone(5)** の man ページ

9.3. ファイアウォールポリシー

ファイアウォールポリシーは、ネットワークの望ましいセキュリティー状態を指定します。これらのポリシーは、さまざまなタイプのトラフィックに対して実行するルールとアクションの概要を示します。通常、ポリシーには次のタイプのトラフィックに対するルールが含まれます。

- 着信トラフィック
- 送信トラフィック
- 転送トラフィック
- 特定のサービスとアプリケーション
- ネットワークアドレス変換 (NAT)

ファイアウォールポリシーは、ファイアウォールゾーンの概念を使用します。各ゾーンは、許可するトラフィックを決定する特定のファイアウォールルールのセットに関連付けられます。ポリシーは、ステートフルかつ一方向にファイアウォールルールを適用します。つまり、トラフィックの一方向のみを考慮します。**firewalld** のステートフルフィルタリングにより、トラフィックのリターンパスは暗黙的に許可されます。

ポリシーは、イングレスゾーンとエグレスゾーンに関連付けられます。イングレスゾーンは、トラフィックが発生する (受信される) 場所です。エグレスゾーンは、トラフィックが出る (送信される) 場所です。

ポリシーで定義されたファイアウォールのルールは、ファイアウォールゾーンを参照して、複数のネットワークインターフェイス全体に一貫した設定を適用できます。

9.4. ファイアウォールのルール

ファイアウォールのルールを使用して、ネットワークトラフィックを許可またはブロックする特定の設定を実装できます。その結果、ネットワークトラフィックのフローを制御して、システムをセキュリティーの脅威から保護できます。

ファイアウォールのルールは通常、さまざまな属性に基づいて特定の基準を定義します。属性は次のとおりです。

- ソース IP アドレス
- 宛先 IP アドレス
- 転送プロトコル (TCP、UDP など
- ポート
- ネットワークインターフェイス

firewalld ユーティリティーは、ファイアウォールのルールをゾーン (**public**、**internal** など) とポリシーに整理します。各ゾーンには、特定のゾーンに関連付けられたネットワークインターフェイスのトラフィック自由度のレベルを決定する独自のルールセットがあります。

9.5. ゾーンの設定ファイル

firewalld ゾーン設定ファイルには、ゾーンに対する情報があります。これは、XML ファイル形式で、ゾーンの説明、サービス、ポート、プロトコル、icmp-block、マスカレード、転送ポート、およびリッチ言語ルールです。ファイル名は **zone-name.xml** となります。zone-name の長さは 17 文字に制限されます。ゾーンの設定ファイルは、**/usr/lib/firewalld/zones/** ディレクトリーおよび **/etc/firewalld/zones/** ディレクトリーに置かれています。

以下の例は、**TCP** プロトコルまたは **UDP** プロトコルの両方に、1 つのサービス (**SSH**) および 1 つのポート範囲を許可する設定を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>My Zone</short>
  <description>Here you can describe the characteristic features of the zone.</description>
  <service name="ssh"/>
  <port protocol="udp" port="1025-65535"/>
  <port protocol="tcp" port="1025-65535"/>
</zone>
```

関連情報

- **firewalld.zone** man ページ

9.6. 事前定義された FIREWALLD サービス

firewalld サービスは、事前定義されたファイアウォールルールのセットで、特定のアプリケーションまたはネットワークサービスへのアクセスを定義します。各サービスは、次の要素の組み合わせを表します。

- ローカルポート
- ネットワークプロトコル
- 関連するファイアウォールルール
- ソースポートと宛先
- サービスが有効になっている場合に自動的にロードされるファイアウォールヘルパーモジュール

サービスは複数のタスクを一度に実行するため、パケットのフィルタリングを簡素化し、時間を短縮します。たとえば、**firewalld** は次のタスクを一度に実行できます。

- ポートを開く
- ネットワークプロトコルを定義する
- パケット転送を有効にする

サービス設定オプションと、一般的なファイル情報は、man ページの **firewalld.service(5)** で説明されています。サービスは、個々の XML 設定ファイルを使用して指定し、名前は、**service-name.xml** のような形式になります。プロトコル名は、**firewalld** のサービス名またはアプリケーション名よりも優先されます。

次の方法で **firewalld** を設定できます。

- 以下のユーティリティーを使用します。
  - **firewall-config** - グラフィカルユーティリティー
  - **firewall-cmd** - コマンドラインユーティリティー
  - **firewall-offline-cmd** - コマンドラインユーティリティー
- **/etc/firewalld/services/** ディレクトリー内の XML ファイルを編集します。

サービスを追加または変更しない場合、対応する XML ファイルは **/etc/firewalld/services/** に存在しません。**/usr/lib/firewalld/services/** 内のファイルをテンプレートとして使用できます。

関連情報

- **firewalld.service(5)** の man ページ

9.7. ファイアウォールゾーンでの作業

ゾーンは、着信トラフィックをより透過的に管理する概念を表しています。ゾーンはネットワークインターフェイスに接続されているか、ソースアドレスの範囲に割り当てられます。各ゾーンは個別にファイアウォールルールを管理しますが、これにより、複雑なファイアウォール設定を定義してトラフィックに割り当てることができます。

9.7.1. 特定のゾーンのファイアウォール設定をカスタマイズすることによるセキュリティーの強化

ファイアウォール設定を変更し、特定のネットワークインターフェイスまたは接続を特定のファイアウォールゾーンに関連付けることで、ネットワークセキュリティーを強化できます。ゾーンの詳細なルールと制限を定義することで、意図したセキュリティーレベルに基づいて受信トラフィックと送信トラフィックを制御できます。

たとえば、次のような利点が得られます。

- 機密データの保護
- 不正アクセスの防止
- 潜在的なネットワーク脅威の軽減

前提条件

- **firewalld** サービスが実行している。

手順

1. 利用可能なファイアウォールゾーンをリスト表示します。

```
# firewall-cmd --get-zones
```

firewall-cmd --get-zones コマンドは、システムで利用可能なすべてのゾーンを表示し、特定のゾーンの詳細は表示しません。すべてのゾーンの詳細情報を表示するには、**firewall-cmd --list-all-zones** コマンドを使用します。

2. この設定に使用するゾーンを選択します。
3. 選択したゾーンのファイアウォール設定を変更します。たとえば、**SSH** サービスを許可し、**ftp** サービスを削除するには、次のようにします。

```
# firewall-cmd --add-service=ssh --zone=<your_chosen_zone>
# firewall-cmd --remove-service=ftp --zone=<same_chosen_zone>
```

4. ネットワークインターフェイスをファイアウォールゾーンに割り当てます。
   a. 使用可能なネットワークインターフェイスをリスト表示します。

```
# firewall-cmd --get-active-zones
```

ゾーンがアクティブかどうかは、その設定と一致するネットワークインターフェイスまたはソースアドレス範囲の存在によって決定します。デフォルトゾーンは、未分類のトラフィックに対してアクティブですが、ルールに一致するトラフィックがない場合は常にアクティブになるわけではありません。

b. 選択したゾーンにネットワークインターフェイスを割り当てます。

```
# firewall-cmd --zone=<your_chosen_zone> --change-interface=<interface_name> --permanent
```

ネットワークインターフェイスをゾーンに割り当てることは、特定のインターフェイス (物理または仮想) 上のすべてのトラフィックに一貫したファイアウォール設定を適用する場合に適しています。

firewall-cmd コマンドを **--permanent** オプションとともに使用すると、多くの場合、NetworkManager 接続プロファイルが更新され、ファイアウォール設定に対する変更が永続化します。この **firewalld** と NetworkManager の統合により、ネットワークとファイアウォールの設定に一貫性が確保されます。

検証

1. 選択したゾーンの更新後の設定を表示します。

```
# firewall-cmd --zone=<your_chosen_zone> --list-all
```

コマンド出力には、割り当てられたサービス、ネットワークインターフェイス、ネットワーク接続 (ソース) を含むすべてのゾーン設定が表示されます。

9.7.2. デフォルトゾーンの変更

システム管理者は、設定ファイルのネットワークインターフェイスにゾーンを割り当てます。特定のゾーンに割り当てられないインターフェイスは、デフォルトゾーンに割り当てられます。**firewalld** サービスを再起動するたびに、**firewalld** は、デフォルトゾーンの設定を読み込み、それをアクティブにします。他のすべてのゾーンの設定は保存され、すぐに使用できます。

通常、ゾーンは NetworkManager により、NetworkManager 接続プロファイルの **connection.zone** 設定に従って、インターフェイスに割り当てられます。また、再起動後、NetworkManager はこれらのゾーンを "アクティブ化" するための割り当てを管理します。

前提条件

- **firewalld** サービスが実行している。

手順

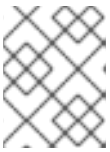
デフォルトゾーンを設定するには、以下を行います。

1. 現在のデフォルトゾーンを表示します。

```
# firewall-cmd --get-default-zone
```

2. 新しいデフォルトゾーンを設定します。

```
# firewall-cmd --set-default-zone <zone_name>
```



注記

この手順では、**--permanent** オプションを使用しなくても、設定は永続化します。

9.7.3. ゾーンへのネットワークインターフェイスの割り当て

複数のゾーンに複数のルールセットを定義して、使用されているインターフェイスのゾーンを変更することで、迅速に設定を変更できます。各インターフェイスに特定のゾーンを設定して、そのゾーンを通過するトラフィックを設定できます。

手順

特定インターフェイスにゾーンを割り当てるには、以下を行います。

1. アクティブゾーン、およびそのゾーンに割り当てられているインターフェイスをリスト表示します。

```
# firewall-cmd --get-active-zones
```

2. 別のゾーンにインターフェイスを割り当てます。

```
# firewall-cmd --zone=zone_name --change-interface=interface_name --permanent
```

9.7.4. nmcli を使用して接続にゾーンを割り当て

nmcli ユーティリティーを使用して、**firewalld** ゾーンを **NetworkManager** 接続に追加できます。

手順

1. ゾーンを **NetworkManager** 接続プロファイルに割り当てます。

```
# nmcli connection modify profile connection.zone zone_name
```

2. 接続をアクティベートします。

```
# nmcli connection up profile
```

9.7.5. 接続プロファイルファイルでネットワーク接続に手動でゾーンを割り当てる

nmcli ユーティリティーを使用して接続プロファイルを変更できない場合は、プロファイルに対応するファイルを手動で編集して、**firewalld** ゾーンを割り当てることができます。

注記

nmcli ユーティリティーを使用して接続プロファイルを変更し、**firewalld** ゾーンを割り当てる方が効率的です。詳細は、ゾーンへのネットワークインターフェイスの割り当てを参照してください。

手順

1. 接続プロファイルへのパスとその形式を決定します。

```
# nmcli -f NAME,FILENAME connection
NAME    FILENAME
enp1s0  /etc/NetworkManager/system-connections/enp1s0.nmconnection
enp7s0  /etc/sysconfig/network-scripts/ifcfg-enp7s0
```

   NetworkManager は、さまざまな接続プロファイル形式に対して個別のディレクトリーとファイル名を使用します。

   - **/etc/NetworkManager/system-connections/<connection_name>.nmconnection** ファイル内のプロファイルは、キーファイル形式を使用します。
   - **/etc/sysconfig/network-scripts/ifcfg-<interface_name>** ファイル内のプロファイルは ifcfg 形式を使用します。

2. 形式に応じて、対応するファイルを更新します。

   - ファイルがキーファイル形式を使用している場合は、**/etc/NetworkManager/system-connections/<connection_name>.nmconnection** ファイルの **[connection]** セクションに **zone=<name>** を追加します。

```
[connection]
...
zone=internal
```

   - ファイルが ifcfg 形式を使用している場合は、**/etc/sysconfig/network-scripts/ifcfg-<interface_name>** ファイルに **ZONE=<name>** を追加します。

```
ZONE=internal
```

3. 接続プロファイルを再読み込みします。

```
# nmcli connection reload
```

4. 接続プロファイルを再度アクティベートします。

```
# nmcli connection up <profile_name>
```

検証

- インターフェイスのゾーンを表示します。以下に例を示します。

```
# firewall-cmd --get-zone-of-interface enp1s0
internal
```

9.7.6. ifcfg ファイルでゾーンをネットワーク接続に手動で割り当て

NetworkManager で接続を管理する場合は、NetworkManager が使用するゾーンを認識する必要があります。すべてのネットワーク接続にゾーンを指定できます。これにより、ポータブルデバイスを使用したコンピューターの場所に従って、様々なファイアウォールを柔軟に設定できるようになります。したがって、ゾーンおよび設定には、会社または自宅など、様々な場所を指定できます。

手順

- 接続のゾーンを設定するには、**/etc/sysconfig/network-scripts/ifcfg-connection_name** ファイルを変更して、この接続にゾーンを割り当てる行を追加します。

```
ZONE=zone_name
```

9.7.7. 新しいゾーンの作成

カスタムゾーンを使用するには、新しいゾーンを作成したり、事前定義したゾーンなどを使用したりします。新しいゾーンには **--permanent** オプションが必要となり、このオプションがなければコマンドは動作しません。

前提条件

- **firewalld** サービスが実行している。

手順

1. 新しいゾーンを作成します。

```
# firewall-cmd --permanent --new-zone=zone-name
```

2. 新しいゾーンを使用可能にします。

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

このコマンドは、すでに実行中のネットワークサービスを中断することなく、最近の変更をファイアウォール設定に適用します。

検証

- 作成したゾーンが永続設定に追加されたかどうかを確認します。

```
# firewall-cmd --get-zones --permanent
```

9.7.8. 着信トラフィックにデフォルトの動作を設定するゾーンターゲットの使用

すべてのゾーンに対して、特に指定されていない着信トラフィックを処理するデフォルト動作を設定できます。そのような動作は、ゾーンのターゲットを設定することで定義されます。4 つのオプションがあります。

- **ACCEPT**: 指定したルールで許可されていないパケットを除いた、すべての着信パケットを許可します。
- **REJECT**: 指定したルールで許可されているパケット以外の着信パケットをすべて拒否します。**firewalld** がパケットを拒否すると、送信元マシンに拒否について通知されます。
- **DROP**: 指定したルールで許可されているパケット以外の着信パケットをすべて破棄します。**firewalld** がパケットを破棄すると、ソースマシンにパケット破棄の通知がされません。
- **default**: **REJECT** と似ていますが、特定のシナリオで特別な意味を持ちます。

前提条件

- **firewalld** サービスが実行している。

手順

ゾーンにターゲットを設定するには、以下を行います。

1. 特定ゾーンに対する情報をリスト表示して、デフォルトゾーンを確認します。

```
# firewall-cmd --zone=zone-name --list-all
```

2. ゾーンに新しいターゲットを設定します。

```
# firewall-cmd --permanent --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

関連情報

- **firewall-cmd(1)** man ページ

9.8. FIREWALLD でネットワークトラフィックの制御

firewalld パッケージは、事前定義された多数のサービスファイルをインストールし、それらをさらに追加したり、カスタマイズしたりできます。さらに、これらのサービス定義を使用して、サービスが使用するプロトコルとポート番号を知らなくても、サービスのポートを開いたり閉じたりできます。

9.8.1. CLI を使用した事前定義サービスによるトラフィックの制御

トラフィックを制御する最も簡単な方法は、事前定義したサービスを **firewalld** に追加する方法です。これにより、必要なすべてのポートが開き、**service definition file** に従ってその他の設定が変更されます。

前提条件

- **firewalld** サービスが実行している。

手順

1. **firewalld** のサービスがまだ許可されていないことを確認します。

```
# firewall-cmd --list-services
ssh dhcpv6-client
```

このコマンドは、デフォルトゾーンで有効になっているサービスをリスト表示します。

2. **firewalld** のすべての事前定義サービスをリスト表示します。

```
# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6
dhcpv6-client dns docker-registry ...
```

このコマンドは、デフォルトゾーンで利用可能なサービスのリストを表示します。

3. **firewalld** が許可するサービスのリストにサービスを追加します。

```
# firewall-cmd --add-service=<service_name>
```

このコマンドは、指定したサービスをデフォルトゾーンに追加します。

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

このコマンドは、これらのランタイムの変更をファイアウォールの永続的な設定に適用します。デフォルトでは、これらの変更はデフォルトゾーンの設定に適用されます。

検証

1. すべての永続的なファイアウォールのルールをリスト表示します。

```
# firewall-cmd --list-all --permanent
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

このコマンドは、デフォルトのファイアウォールゾーン (**public**) の永続的なファイアウォールのルールを含む完全な設定を表示します。

2. **firewalld** サービスの永続的な設定の有効性を確認します。

```
# firewall-cmd --check-config
success
```

永続的な設定が無効な場合、コマンドは詳細を含むエラーを返します。

```
# firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
```

永続的な設定ファイルを手動で検査して設定を確認することもできます。メインの設定ファイルは **/etc/firewalld/firewalld.conf** です。ゾーン固有の設定ファイルは **/etc/firewalld/zones/** ディレクトリーにあり、ポリシーは **/etc/firewalld/policies/** ディレクトリーにあります。

9.8.2. GUI を使用した事前定義サービスによるトラフィックの制御

グラフィカルユーザーインターフェイスを使用して、事前定義されたサービスでネットワークトラフィックを制御できます。Firewall Configuration アプリケーションは、コマンドラインユーティリティーに代わる、アクセスしやすくユーザーフレンドリーな代替手段を提供します。

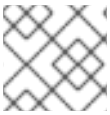
前提条件

- **firewall-config** パッケージがインストールされている。
- **firewalld** サービスが実行している。

手順

1. 事前定義したサービスまたはカスタマイズしたサービスを有効または無効にするには、以下を行います。
   a. **firewall-config** ユーティリティーを起動して、サービスを設定するネットワークゾーンを選択します。
   b. **Zones** タブを選択してから、下の **Services** タブを選択します。
   c. 信頼するサービスのタイプごとにチェックボックスをオンにするか、チェックボックスをオフにして、選択したゾーンのサービスをブロックします。
2. サービスを編集するには、以下を行います。
   a. **firewall-config** ユーティリティーを起動します。
   b. **Configuration** メニューから **Permanent** を選択します。**Services** ウィンドウの下部に、その他のアイコンおよびメニューボタンが表示されます。
   c. 設定するサービスを選択します。

Ports、**Protocols**、**Source Port** のタブでは、選択したサービスのポート、プロトコル、およびソースポートの追加、変更、ならびに削除が可能です。モジュールタブは、**Netfilter** ヘルパーモジュールの設定を行います。**Destination** タブは、特定の送信先アドレスとインターネットプロトコル (**IPv4** または **IPv6**) へのトラフィックが制限できます。

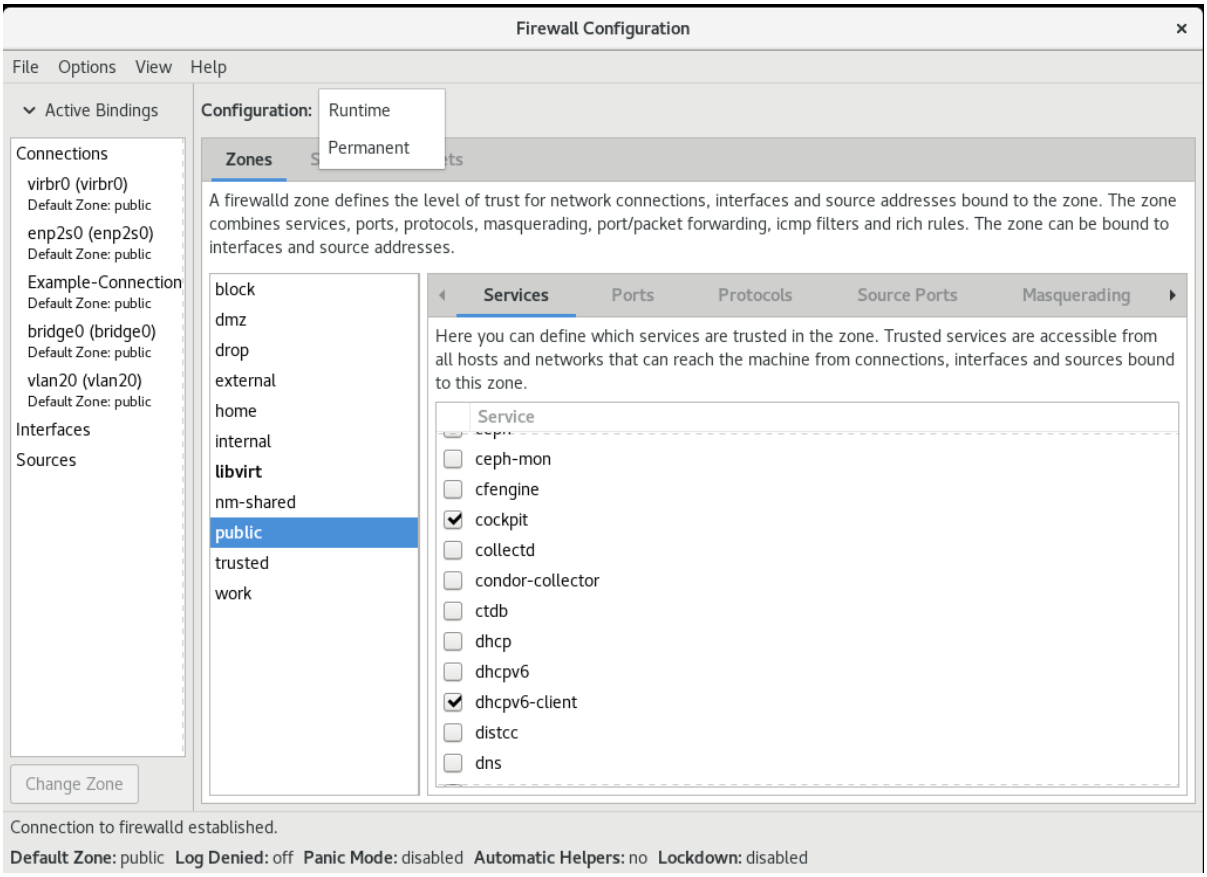


注記

Runtime モードでは、サービス設定を変更できません。

検証

- **Super** キーを押して、アクティビティーの概要に入ります。
- Firewall Configuration ユーティリティーを選択します。
  - コマンドラインで **firewall-config** コマンドを入力して、グラフィカルファイアウォール設定ユーティリティーを起動することもできます。
- ファイアウォールの設定のリストを表示します。



Firewall Configuration ウィンドウが開きます。このコマンドは通常のユーザーとして実行できますが、監理者パスワードが求められる場合もあります。

9.8.3. セキュアな Web サーバーのホストを可能にする firewalld の設定

ポートは、オペレーティングシステムがネットワークトラフィックを受信して区別し、システムサービスに転送できるようにする論理サービスです。このシステムサービスは、ポートをリッスンし、ポートに入るトラフィックを待機するデーモンによって表されます。

通常、システムサービスは、サービスに予約されている標準ポートでリッスンします。**httpd** デーモンは、たとえば、ポート 80 をリッスンします。ただし、システム管理者は、サービス名の代わりにポート番号を直接指定できます。

firewalld サービスを使用して、データをホストするためのセキュアな Web サーバーへのアクセスを設定できます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 現在アクティブなファイアウォールゾーンを確認します。

```
# firewall-cmd --get-active-zones
```

2. HTTPS サービスを適切なゾーンに追加します。

```
# firewall-cmd --zone=<zone_name> --add-service=https --permanent
```

3. ファイアウォール設定を再読み込みします。

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

検証

1. **firewalld** でポートが開いているかどうかを確認します。
   - ポート番号を指定してポートを開いた場合は、次のように入力します。

```
# firewall-cmd --zone=<zone_name> --list-all
```

   - サービス定義を指定してポートを開いた場合は、次のように入力します。

```
# firewall-cmd --zone=<zone_name> --list-services
```

9.8.4. ネットワークのセキュリティーを強化するための不使用または不要なポートの閉鎖

開いているポートが不要になった場合は、**firewalld** ユーティリティーを使用してポートを閉じることができます。

重要

不要なポートをすべて閉じて、潜在的な攻撃対象領域を減らし、不正アクセスや脆弱性悪用のリスクを最小限に抑えてください。

手順

1. 許可されているポートのリストを表示します。

```
# firewall-cmd --list-ports
```

デフォルトでは、このコマンドはデフォルトゾーンで有効になっているポートをリスト表示します。

注記

このコマンドでは、ポートとして開かれているポートのみが表示されます。サービスとして開かれているポートは表示されません。その場合は、**--list-ports** の代わりに **--list-all** オプションの使用を検討してください。

2. 許可されているポートのリストからポートを削除し、着信トラフィックに対して閉じます。

```
# firewall-cmd --remove-port=port-number/port-type
```

このコマンドは、ゾーンからポートを削除します。ゾーンを指定しない場合は、デフォルトゾーンからポートが削除されます。

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

ゾーンを指定しない場合、このコマンドは、ランタイムの変更をデフォルトゾーンの永続的な設定に適用します。

検証

1. アクティブなゾーンをリスト表示し、検査するゾーンを選択します。

```
# firewall-cmd --get-active-zones
```

2. 選択したゾーンで現在開いているポートをリスト表示し、不使用または不要なポートが閉じているかどうかを確認します。

```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

9.8.5. CLI を使用したトラフィックの制御

firewall-cmd コマンドを使用すると、次のことが可能です。

- ネットワークトラフィックの無効化
- ネットワークトラフィックの有効化

その結果、たとえばシステムの防御を強化したり、データのプライバシーを確保したり、ネットワークリソースを最適化したりすることができます。

重要

パニックモードを有効にすると、ネットワークトラフィックがすべて停止します。したがって、そのマシンへの物理アクセスがある場合、またはシリアルコンソールを使用してログインする場合に限り使用してください。

手順

1. ネットワークトラフィックを直ちに無効にするには、パニックモードをオンにします。

```
# firewall-cmd --panic-on
```

2. パニックモードをオフにし、ファイアウォールを永続設定に戻します。パニックモードを無効にするには、次のコマンドを実行します。

```
# firewall-cmd --panic-off
```

検証

- パニックモードを有効または無効にするには、次のコマンドを実行します。

```
# firewall-cmd --query-panic
```

9.8.6. GUI を使用してプロトコルを使用したトラフィックの制御

特定のプロトコルを使用してファイアウォールを経由したトラフィックを許可するには、GUI を使用できます。

前提条件

- **firewall-config** パッケージがインストールされている

手順

1. firewall-config ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側で **Protocols** タブを選択し、**Add** ボタンをクリックします。**Protocol** ウィンドウが開きます。
3. リストからプロトコルを選択するか、**Other Protocol** チェックボックスを選択し、そのフィールドにプロトコルを入力します。

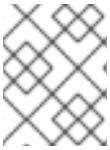
9.9. ゾーンを使用し、ソースに応じた着信トラフィックの管理

ゾーンを使用して、そのソースに基づいて着信トラフィックを管理するゾーンを使用できます。このコンテキストでの着信トラフィックとは、システム宛てのデータ、または **firewalld** を実行しているホストを通過するデータです。ソースは通常、トラフィックの発信元の IP アドレスまたはネットワーク範囲を指します。その結果、着信トラフィックをソートして異なるゾーンに割り当て、そのトラフィックが到達できるサービスを許可または禁止することができます。

ソースアドレスによる一致は、インターフェイス名による一致よりも優先されます。ソースをゾーンに追加すると、ファイアウォールは、インターフェイスベースのルールよりも着信トラフィックに対するソースベースのルールを優先します。これは、着信トラフィックが特定のゾーンに指定されたソースアドレスと一致する場合、トラフィックが通過するインターフェイスに関係なく、ソースアドレスに関連付けられたゾーンによってトラフィックの処理方法が決定されることを意味します。一方、インターフェイスベースのルールは通常、特定のソースベースのルールに一致しないトラフィックのためのフォールバックです。これらのルールは、ソースがゾーンに明示的に関連付けられていないトラフィックに適用されます。これにより、特定のソース定義ゾーンがないトラフィックのデフォルトの動作を定義できます。

9.9.1. ソースの追加

着信トラフィックを特定のゾーンに転送する場合は、そのゾーンにソースを追加します。ソースは、CIDR (Classless Inter-domain Routing) 表記法の IP アドレスまたは IP マスクになります。



注記

ネットワーク範囲が重複している複数のゾーンを追加する場合は、ゾーン名で順序付けされ、最初のゾーンのみが考慮されます。

- 現在のゾーンにソースを設定するには、次のコマンドを実行します。

```
# firewall-cmd --add-source=<source>
```

- 特定ゾーンのソース IP アドレスを設定するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-source=<source>
```

以下の手順は、**信頼される** ゾーンで 192.168.2.15 からのすべての着信トラフィックを許可します。

手順

1. 利用可能なすべてのゾーンをリストします。

```
# firewall-cmd --get-zones
```

2. 永続化モードで、信頼ゾーンにソース IP を追加します。

```
# firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

9.9.2. ソースの削除

ゾーンからソースを削除すると、当該ソースに指定したルールは、そのソースから発信されたトラフィックに適用されなくなります。代わりに、トラフィックは、その発信元のインターフェイスに関連付けられたゾーンのルールと設定にフォールバックするか、デフォルトゾーンに移動します。

手順

1. 必要なゾーンに対して許可されているソースのリストを表示します。

```
# firewall-cmd --zone=zone-name --list-sources
```

2. ゾーンからソースを永続的に削除します。

```
# firewall-cmd --zone=zone-name --remove-source=<source>
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

9.9.3. ソースポートの削除

ソースポートを削除して、送信元ポートに基づいてトラフィックの分類を無効にします。

手順

- ソースポートを削除するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --remove-source-port=<port-name>/<tcp|udp|sctp|dccp>
```

9.9.4. ゾーンおよびソースを使用して特定ドメインのみに対してサービスの許可

特定のネットワークからのトラフィックを許可して、マシンのサービスを使用するには、ゾーンおよびソースを使用します。以下の手順では、他のトラフィックがブロックされている間に **192.0.2.0/24** ネットワークからの HTTP トラフィックのみを許可します。

警告

このシナリオを設定する場合は、**default** のターゲットを持つゾーンを使用します。**192.0.2.0/24** からのトラフィックではネットワーク接続がすべて許可されるため、ターゲットが **ACCEPT** に設定されたゾーンを使用することは、セキュリティー上のリスクになります。

手順

1. 利用可能なすべてのゾーンをリストします。

```
# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

2. IP 範囲を **internal** ゾーンに追加し、ソースから発信されるトラフィックをゾーン経由でルーティングします。

```
# firewall-cmd --zone=internal --add-source=192.0.2.0/24
```

3. **http** サービスを **internal** ゾーンに追加します。

```
# firewall-cmd --zone=internal --add-service=http
```

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

検証

- **internal** ゾーンがアクティブで、サービスが許可されていることを確認します。

```
# firewall-cmd --zone=internal --list-all
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.0.2.0/24
  services: cockpit dhcpv6-client mdns samba-client ssh http
  ...
```

関連情報

- **firewalld.zones(5)** の man ページ

9.10. ゾーン間で転送されるトラフィックのフィルタリング

firewalld を使用すると、異なる **firewalld** ゾーン間のネットワークデータのフローを制御できます。ルールとポリシーを定義することで、これらのゾーン間を移動するトラフィックをどのように許可またはブロックするかを管理できます。

ポリシーオブジェクト機能は、**firewalld** で正引きフィルターと出力フィルターを提供します。**firewalld** を使用して、異なるゾーン間のトラフィックをフィルタリングし、ローカルでホストされている仮想マシンへのアクセスを許可して、ホストを接続できます。

9.10.1. ポリシーオブジェクトとゾーンの関係

ポリシーオブジェクトを使用すると、サービス、ポート、リッチルールなどの firewalld のプリミティブをポリシーに割り当てることができます。ポリシーオブジェクトは、ステートフルおよび一方向の方法でゾーン間を通過するトラフィックに適用することができます。

```
# firewall-cmd --permanent --new-policy myOutputPolicy

# firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST

# firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

HOST および **ANY** は、イングレスゾーンおよびエグレスゾーンのリストで使用されるシンボリックゾーンです。

- **HOST** シンボリックゾーンは、firewalld を実行しているホストから発信されるトラフィック、またはホストへの宛先を持つトラフィックのポリシーを許可します。
- **ANY** シンボリックゾーンは、現行および将来のすべてのゾーンにポリシーを適用します。**ANY** シンボリックゾーンは、すべてのゾーンのワイルドカードとして機能します。

9.10.2. 優先度を使用したポリシーのソート

同じトラフィックセットに複数のポリシーを適用できるため、優先度を使用して、適用される可能性のあるポリシーの優先順位を作成する必要があります。

ポリシーをソートする優先度を設定するには、次のコマンドを実行します。

```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

この例では、**-500** の優先度は低くなりますが、優先度は高くなります。したがって、-500 は、-100 より前に実行されます。

優先度の数値が小さいほど優先度が高く、最初に適用されます。

9.10.3. ポリシーオブジェクトを使用した、ローカルでホストされているコンテナーと、ホストに物理的に接続されているネットワークとの間でのトラフィックのフィルタリング

ポリシーオブジェクト機能を使用すると、ユーザーは Podman ゾーンと firewalld ゾーン間のトラフィックをフィルタリングできます。

注記

Red Hat は、デフォルトではすべてのトラフィックをブロックし、Podman ユーティリティーに必要なサービスを選択して開くことを推奨します。

手順

1. 新しいファイアウォールポリシーを作成します。

```
# firewall-cmd --permanent --new-policy podmanToAny
```

2. Podman から他のゾーンへのすべてのトラフィックをブロックし、Podman で必要なサービスのみを許可します。

```
# firewall-cmd --permanent --policy podmanToAny --set-target REJECT
# firewall-cmd --permanent --policy podmanToAny --add-service dhcp
# firewall-cmd --permanent --policy podmanToAny --add-service dns
# firewall-cmd --permanent --policy podmanToAny --add-service https
```

3. 新しい Podman ゾーンを作成します。

```
# firewall-cmd --permanent --new-zone=podman
```

4. ポリシーのイングレスゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

5. 他のすべてのゾーンのエグレスゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

エグレスゾーンを ANY に設定すると、Podman と他のゾーンの間でフィルタリングすることになります。ホストに対してフィルタリングする場合は、エグレスゾーンを HOST に設定します。

6. firewalld サービスを再起動します。

```
# systemctl restart firewalld
```

検証

- 他のゾーンに対する Podman ファイアウォールポリシーを検証します。

```
# firewall-cmd --info-policy podmanToAny
podmanToAny (active)
  ...
  target: REJECT
  ingress-zones: podman
  egress-zones: ANY
  services: dhcp dns https
  ...
```

9.10.4. ポリシーオブジェクトのデフォルトターゲットの設定

ポリシーには --set-target オプションを指定できます。以下のターゲットを使用できます。

- **ACCEPT** - パケットを受け入れます

- **DROP** - 不要なパケットを破棄します
- **REJECT** - ICMP 応答で不要なパケットを拒否します
- **CONTINUE** (デフォルト) - パケットは、次のポリシーとゾーンのルールに従います。

```
# firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

検証

- ポリシーに関する情報の確認

```
# firewall-cmd --info-policy mypolicy
```

9.10.5. DNAT を使用して HTTPS トラフィックを別のホストに転送する

Web サーバーがプライベート IP アドレスを持つ DMZ で実行されている場合は、宛先ネットワークアドレス変換 (DNAT) を設定して、インターネット上のクライアントがこの Web サーバーに接続できるようにすることができます。この場合、Web サーバーのホスト名はルーターのパブリック IP アドレスに解決されます。クライアントがルーターの定義済みポートへの接続を確立すると、ルーターはパケットを内部 Web サーバーに転送します。

前提条件

- DNS サーバーが、Web サーバーのホスト名をルーターの IP アドレスに解決している。
- 次の設定を把握している。
  - 転送するプライベート IP アドレスおよびポート番号
  - 使用する IP プロトコル
  - パケットをリダイレクトする Web サーバーの宛先 IP アドレスおよびポート

手順

1. ファイアウォールポリシーを作成します。

```
# firewall-cmd --permanent --new-policy <example_policy>
```

ポリシーは、ゾーンとは対照的に、入力、出力、および転送されるトラフィックのパケットフィルタリングを許可します。ローカルで実行されている Web サーバー、コンテナー、または仮想マシン上のエンドポイントにトラフィックを転送するには、このような機能が必要になるため、これは重要です。

2. イングレストラフィックとエグレストラフィックのシンボリックゾーンを設定して、ルーター自体がローカル IP アドレスに接続し、このトラフィックを転送できるようにします。

```
# firewall-cmd --permanent --policy=<example_policy> --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=<example_policy> --add-egress-zone=ANY
```

--add-ingress-zone=HOST オプションは、ローカルで生成され、ローカルホストから送信されるパケットを参照します。**--add-egress-zone=ANY** オプションは、任意のゾーンに向かうトラフィックを参照します。

3. トラフィックを Web サーバーに転送するリッチルールを追加します。

```
# firewall-cmd --permanent --policy=<example_policy> --add-rich-rule='rule
family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp"
to-port="443" to-addr="192.51.100.20"'
```

リッチルールは、ルーターの IP アドレス (192.0.2.1) のポート 443 から Web サーバーの IP アドレス (192.51.100.20) のポート 443 に TCP トラフィックを転送します。

4. ファイアウォール設定ファイルをリロードします。

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

5. カーネルで 127.0.0.0/8 のルーティングを有効にします。

- 変更を永続化するには、次を実行します。

```
# echo "net.ipv4.conf.all.route_localnet=1" > /etc/sysctl.d/90-enable-route-
localnet.conf
```

このコマンドは、**route_localnet** カーネルパラメーターを永続的に設定し、システムの再起動後も設定が確実に保持されるようにします。

- システムを再起動することなく直ちに設定を適用するには、次のコマンドを実行します。

```
# sysctl -p /etc/sysctl.d/90-enable-route-localnet.conf
```

sysctl コマンドは、オンザフライで変更を適用するのに便利ですが、システムを再起動すると設定は元に戻ります。

検証

1. Web サーバーに転送したルーターの IP アドレスおよびポートに接続します。

```
# curl https://192.0.2.1:443
```

2. **net.ipv4.conf.all.route_localnet** カーネルパラメーターがアクティブであることを確認します。

```
# sysctl net.ipv4.conf.all.route_localnet
net.ipv4.conf.all.route_localnet = 1
```

3. **<example_policy>** がアクティブであり、必要な設定 (特にソース IP アドレスとポート、使用するプロトコル、宛先 IP アドレスとポート) が含まれていることを確認します。

```
# firewall-cmd --info-policy=<example_policy>
example_policy (active)
  priority: -1
  target: CONTINUE
  ingress-zones: HOST
  egress-zones: ANY
  services:
  ports:
```

```
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
 rule family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp" to-
port="443" to-addr="192.51.100.20"
```

関連情報

- **firewall-cmd(1)**、**firewalld.policies(5)**、**firewalld.richlanguage(5)**、**sysctl(8)**、および **sysctl.conf(5)** の man ページ
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

9.11. FIREWALLD を使用した NAT の設定

firewalld では、以下のネットワークアドレス変換 (NAT) タイプを設定できます。

- マスカレーディング
- 宛先 NAT (DNAT)
- リダイレクト

9.11.1. ネットワークアドレス変換のタイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレーディング

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダー (ISP) は、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、パケットをローカルマシンの別のポートにリダイレクトする DNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

9.11.2. IP アドレスのマスカレードの設定

システムで IP マスカレードを有効にできます。IP マスカレードは、インターネットにアクセスする際にゲートウェイの向こう側にある個々のマシンを隠します。

手順

1. **external** ゾーンなどで IP マスカレーディングが有効かどうかを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --query-masquerade
```

このコマンドでは、有効な場合は **yes** と出力され、終了ステータスは **0** になります。無効の場合は **no** と出力され、終了ステータスは **1** になります。**zone** を省略すると、デフォルトのゾーンが使用されます。

2. IP マスカレードを有効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --add-masquerade
```

3. この設定を永続化するには、**--permanent** オプションをコマンドに渡します。

4. IP マスカレードを無効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --remove-masquerade
```

この設定を永続化するには、**--permanent** をコマンドラインに渡します。

9.11.3. DNAT を使用した着信 HTTP トラフィックの転送

宛先ネットワークアドレス変換 (DNAT) を使用して、着信トラフィックを 1 つの宛先アドレスおよびポートから別の宛先アドレスおよびポートに転送できます。通常、外部ネットワークインターフェイスからの着信リクエストを特定の内部サーバーまたはサービスにリダイレクトする場合に役立ちます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 次の内容を含む **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルを作成します。

```
net.ipv4.ip_forward=1
```

この設定によって、カーネルでの IP 転送が有効になります。これにより、内部 RHEL サーバーがルーターとして機能し、ネットワークからネットワークへパケットを転送するようになります。

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルから設定をロードします。

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. 着信 HTTP トラフィックを転送します。

```
# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toaddr=198.51.100.10:toport=8080 --permanent
```

上記のコマンドは、次の設定で DNAT ルールを定義します。

- **--zone=public** - DNAT ルールを設定するファイアウォールゾーン。必要なゾーンに合わせて調整できます。
- **--add-forward-port** - ポート転送ルールを追加することを示すオプション。
- **port=80** - 外部宛先ポート。
- **proto=tcp** - TCP トラフィックを転送することを示すプロトコル。
- **toaddr=198.51.100.10** - 宛先 IP アドレス。
- **toport=8080** - 内部サーバーの宛先ポート。
- **--permanent** - 再起動後も DNAT ルールを永続化するオプション。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- 使用したファイアウォールゾーンの DNAT ルールを確認します。

```
# firewall-cmd --list-forward-ports --zone=public
port=80:proto=tcp:toport=8080:toaddr=198.51.100.10
```

あるいは、対応する XML 設定ファイルを表示します。

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="80" protocol="tcp" to-port="8080" to-addr="198.51.100.10"/>
  <forward/>
</zone>
```

関連情報

- ランタイム時のカーネルパラメーターの設定
- **firewall-cmd(1)** man ページ

9.11.4. 非標準ポートからのトラフィックをリダイレクトして、標準ポートで Web サービスにアクセスできるようにする

リダイレクトメカニズムを使用すると、ユーザーが URL でポートを指定しなくても、非標準ポートで内部的に実行される Web サービスにアクセスできるようになります。その結果、URL はよりシンプルになり、ブラウジングエクスペリエンスが向上します。一方で、非標準ポートは依然として内部で、または特定の要件のために使用されます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 次の内容を含む **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルを作成します。

```
net.ipv4.ip_forward=1
```

この設定によって、カーネルでの IP 転送が有効になります。

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルから設定をロードします。

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. NAT リダイレクトルールを作成します。

```
# firewall-cmd --zone=public --add-forward-
port=port=<standard_port>:proto=tcp:toport=<non_standard_port> --permanent
```

上記のコマンドは、次の設定で NAT リダイレクトルールを定義します。

- **--zone=public** - ルールを設定するファイアウォールゾーン。必要なゾーンに合わせて調整できます。
- **--add-forward-port=port=<non_standard_port>** - 着信トラフィックを最初に受信するソースポートを使用したポート転送 (リダイレクト) ルールを追加することを示すオプション。
- **proto=tcp** - TCP トラフィックをリダイレクトすることを示すプロトコル。
- **toport=<standard_port>** - 着信トラフィックがソースポートで受信された後にリダイレクトされる宛先ポート。
- **--permanent** - 再起動後もルールを永続化するオプション。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- 使用したファイアウォールゾーンのリダイレクトルールを確認します。

```
# firewall-cmd --list-forward-ports
port=8080:proto=tcp:toport=80:toaddr=
```

あるいは、対応する XML 設定ファイルを表示します。

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
```

```
  <description>For use in public areas. You do not trust the other computers on networks to
not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="8080" protocol="tcp" to-port="80"/>
  <forward/>
</zone>
```

関連情報

- ランタイム時のカーネルパラメーターの設定
- **firewall-cmd(1)** man ページ

9.12. ICMP リクエストの管理

Internet Control Message Protocol (**ICMP**) は、テスト、トラブルシューティング、診断のために、さまざまなネットワークデバイスによって使用されるサポート対象のプロトコルです。**ICMP** は、システム間でデータを交換するのに使用されていないため、TCP、UDP などの転送プロトコルとは異なります。

ICMP メッセージ (特に **echo-request** および **echo-reply**) を利用して、ネットワークに関する情報を明らかにし、その情報をさまざまな不正行為に悪用することが可能です。したがって、**firewalld** は、ネットワーク情報を保護するため、**ICMP** リクエストを制御できます。

9.12.1. ICMP フィルタリングの設定

ICMP フィルタリングを使用すると、ファイアウォールでシステムへのアクセスを許可または拒否する ICMP のタイプとコードを定義できます。ICMP のタイプとコードは、ICMP メッセージの特定のカテゴリーとサブカテゴリーです。

ICMP フィルタリングは、たとえば次の分野で役立ちます。

- セキュリティーの強化 - 潜在的に有害な ICMP のタイプとコードをブロックして、攻撃対象領域を縮小します。
- ネットワークパフォーマンス - 必要な ICMP タイプのみを許可してネットワークパフォーマンスを最適化し、過剰な ICMP トラフィックによって引き起こされる潜在的なネットワーク輻輳を防ぎます。
- トラブルシューティングの制御 - ネットワークのトラブルシューティングに不可欠な ICMP 機能を維持し、潜在的なセキュリティーリスクとなる ICMP タイプをブロックします。

前提条件

- **firewalld** サービスが実行している。

手順

1. 利用可能な ICMP のタイプとコードをリスト表示します。

```
# firewall-cmd --get-icmptypes
address-unreachable bad-header beyond-scope communication-prohibited destination-
unreachable echo-reply echo-request failed-policy fragmentation-needed host-precedence-
```

```
violation host-prohibited host-redirect host-unknown host-unreachable
...
```

この事前定義されたリストから、許可またはブロックする ICMP のタイプとコードを選択します。

2. 特定の ICMP タイプを次の方法でフィルタリングします。

   - 許可する ICMP タイプ:

     ```
     # firewall-cmd --zone=<target-zone> --remove-icmp-block=echo-request --permanent
     ```

     このコマンドは、エコーリクエスト ICMP タイプに対する既存のブロックルールを削除します。

   - ブロックする ICMP タイプ:

     ```
     # firewall-cmd --zone=<target-zone> --add-icmp-block=redirect --permanent
     ```

     このコマンドは、リダイレクトメッセージ ICMP タイプがファイアウォールによって確実にブロックされるようにします。

3. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- フィルタリングルールが有効であることを確認します。

  ```
  # firewall-cmd --list-icmp-blocks
  redirect
  ```

  コマンド出力には、許可またはブロックした ICMP のタイプとコードが表示されます。

関連情報

- **firewall-cmd(1)** man ページ

9.13. FIREWALLD を使用した IP セットの設定および制御

IP セットは、より柔軟かつ効率的にファイアウォールのルールを管理するために、IP アドレスとネットワークをセットにグループ化する RHEL 機能です。

IP セットは、たとえば次のようなシナリオで役立ちます。

- 大きな IP アドレスリストを処理する場合
- これらの大きな IP アドレスリストに動的更新を実装する場合
- カスタムの IP ベースポリシーを作成して、ネットワークのセキュリティーと制御を強化する場合

警告

Red Hat では、**firewall-cmd** コマンドを使用して IP セットを作成および管理することを推奨します。

9.13.1. IP セットを使用した許可リストの動的更新の設定

ほぼリアルタイムで更新を行うことで、予測不可能な状況でも IP セット内の特定の IP アドレスまたは IP アドレス範囲を柔軟に許可できます。これらの更新は、セキュリティー脅威の検出やネットワーク動作の変化など、さまざまなイベントによってトリガーされます。通常、このようなソリューションでは自動化を活用して手動処理を減らし、素早く状況に対応することでセキュリティーを向上させます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 分かりやすい名前で IP セットを作成します。

```
# firewall-cmd --permanent --new-ipset=allowlist --type=hash:ip
```

この **allowlist** という新しい IP セットには、ファイアウォールで許可する IP アドレスが含まれています。

2. IP セットに動的更新を追加します。

```
# firewall-cmd --permanent --ipset=allowlist --add-entry=198.51.100.10
```

この設定により、新しく追加した、ネットワークトラフィックを渡すことがファイアウォールにより許可される IP アドレスで、**allowlist** の IP セットが更新されます。

3. 先に作成した IP セットを参照するファイアウォールのルールを作成します。

```
# firewall-cmd --permanent --zone=public --add-source=ipset:allowlist
```

このルールがない場合、IP セットはネットワークトラフィックに影響を与えません。デフォルトのファイアウォールポリシーが優先されます。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

1. すべての IP セットをリスト表示します。

```
# firewall-cmd --get-ipsets
allowlist
```

2. アクティブなルールをリスト表示します。

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s1
  sources: ipset:allowlist
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  ...
```

コマンドライン出力の **sources** セクションでは、どのトラフィックの発信元 (ホスト名、インターフェイス、IP セット、サブネットなど) が、特定のファイアウォールゾーンへのアクセスを許可または拒否されているかについての洞察が得られます。上記の場合、**allowlist** IP セットに含まれる IP アドレスが、**public** ゾーンのファイアウォールを通してトラフィックを渡すことが許可されています。

3. IP セットの内容を調べます。

```
# cat /etc/firewalld/ipsets/allowlist.xml
<?xml version="1.0" encoding="utf-8"?>
<ipset type="hash:ip">
  <entry>198.51.100.10</entry>
</ipset>
```

次のステップ

- スクリプトまたはセキュリティーユーティリティーを使用して脅威インテリジェンスのフィードを取得し、それに応じて **allowlist** を自動的に更新します。

関連情報

- **firewall-cmd(1)** man ページ

9.14. リッチルールの優先度設定

デフォルトでは、リッチルールはルールアクションに基づいて設定されます。たとえば、**許可** ルールよりも **拒否** ルールが優先されます。リッチルールで **priority** パラメーターを使用すると、管理者はリッチルールとその実行順序をきめ細かく制御できます。**priority** パラメーターを使用すると、ルールはまず優先度の値によって昇順にソートされます。多くのルールが同じ **priority** を持つ場合、ルールの順序はルールアクションによって決まります。アクションも同じである場合、順序は定義されない可能性があります。

9.14.1. priority パラメーターを異なるチェーンにルールを整理する方法

リッチルールの **priority** パラメーターは、**-32768** から **32767** までの任意の数値に設定でき、数値が小さいほど優先度が高くなります。

firewalld サービスは、優先度の値に基づいて、ルールを異なるチェーンに整理します。

- 優先度が 0 未満 - ルールは **_pre** 接尾辞が付いたチェーンにリダイレクトされます。

- 優先度が 0 を超える - ルールは **_post** 接尾辞が付いたチェーンにリダイレクトされます。
- 優先度が 0 - アクションに基づいて、ルールは、**_log**、**_deny**、または **_allow** のアクションを使用してチェーンにリダイレクトされます。

このサブチェーンでは、**firewalld** は優先度の値に基づいてルールを分類します。

9.14.2. リッチルールの優先度の設定

以下は、**priority** パラメーターを使用して、他のルールで許可または拒否されていないすべてのトラフィックをログに記録するリッチルールを作成する方法を示しています。このルールを使用して、予期しないトラフィックにフラグを付けることができます。

手順

- 優先度が非常に低いルールを追加して、他のルールと一致していないすべてのトラフィックをログに記録します。

```
# firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"'
```

このコマンドでは、ログエントリーの数を、毎分 **5** に制限します。

検証

- 前の手順のコマンドで作成した **nftables** ルールを表示します。

```
# nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
  chain filter_IN_public_post {
    log prefix "UNEXPECTED: " limit rate 5/minute
  }
}
```

9.15. ファイアウォールロックダウンの設定

ローカルのアプリケーションやサービスは、**root** で実行していれば、ファイアウォール設定を変更できます (たとえば **libvirt**)。管理者は、この機能を使用してファイアウォール設定をロックし、すべてのアプリケーションでファイアウォール変更を要求できなくするか、ロックダウンの許可リストに追加されたアプリケーションのみがファイアウォール変更を要求できるようにすることが可能になります。ロックダウン設定はデフォルトで無効になっています。これを有効にすると、ローカルのアプリケーションやサービスによるファイアウォールへの望ましくない設定変更を確実に防ぐことができます。

9.15.1. CLI を使用したロックダウンの設定

コマンドラインでロックダウン機能を有効または無効にすることができます。

手順

1. ロックダウンが有効かどうかをクエリーするには、以下を実行します。

```
# firewall-cmd --query-lockdown
```

2. 次のいずれかの方法でロックダウン設定を管理します。

   - ロックダウンを有効にする場合:

     ```
     # firewall-cmd --lockdown-on
     ```

   - ロックダウンを無効にする場合:

     ```
     # firewall-cmd --lockdown-off
     ```

9.15.2. ロックダウン許可リスト設定ファイルの概要

デフォルトの許可リスト設定ファイルには、**NetworkManager** コンテキストと、**libvirt** のデフォルトコンテキストが含まれます。リストには、ユーザー ID (0) もあります。

許可リスト設定ファイルは **/etc/firewalld/** ディレクトリーに保存されます。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
  <command name="/usr/bin/python3 -s /usr/bin/firewall-config"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

以下の許可リスト設定ファイルの例では、**firewall-cmd** ユーティリティーのコマンドと、ユーザー ID が **815** である **user** のコマンドをすべて有効にしています。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <command name="/usr/libexec/platform-python -s /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

この例では、**user id** と **user name** の両方が使用されていますが、実際にはどちらか一方のオプションだけが必要です。Python はインタープリターとしてコマンドラインに追加されています。

Red Hat Enterprise Linux では、すべてのユーティリティーが **/usr/bin/** ディレクトリーに格納されており、**/bin/** ディレクトリーは **/usr/bin/** ディレクトリーへのシンボリックリンクとなります。つまり、**root** で **firewall-cmd** のパスを実行すると **/bin/firewall-cmd** に対して解決しますが、**/usr/bin/firewall-cmd** が使用できるようになっています。新たなスクリプトは、すべて新しい格納場所を使用する必要があります。ただし、**root** で実行するスクリプトが **/bin/firewall-cmd** へのパスを使用するようになっているのであれば、これまでは **root** 以外のユーザーにのみ使用されていた **/usr/bin/firewall-cmd** パスに加え、このコマンドのパスも許可リストに追加する必要があります。

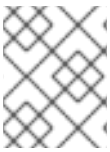
コマンドの名前属性の最後にある * は、その名前で始まるすべてのコマンドが一致することを意味します。* がなければ、コマンドと引数が完全に一致する必要があります。

9.16. FIREWALLD ゾーン内の異なるインターフェイスまたはソース間でのトラフィック転送の有効化

ゾーン内転送は、**firewalld** ゾーン内のインターフェイスまたはソース間のトラフィック転送を可能にする **firewalld** 機能です。

9.16.1. ゾーン内転送と、デフォルトのターゲットが ACCEPT に設定されているゾーンの違い

ゾーン内転送を有効にすると、1 つの **firewalld** ゾーン内のトラフィックは、あるインターフェイスまたはソースから別のインターフェイスまたはソースに流れることができます。ゾーンは、インターフェイスおよびソースの信頼レベルを指定します。信頼レベルが同じ場合、トラフィックは同じゾーン内に留まります。



注記

firewalld のデフォルトゾーンでゾーン内転送を有効にすると、現在のデフォルトゾーンに追加されたインターフェイスおよびソースにのみ適用されます。

firewalld は、異なるゾーンを使用して着信トラフィックと送信トラフィックを管理します。各ゾーンには独自のルールと動作のセットがあります。たとえば、**trusted** ゾーンでは、転送されたトラフィックがデフォルトですべて許可されます。

他のゾーンでは、異なるデフォルト動作を設定できます。標準ゾーンでは、ゾーンのターゲットが **default** に設定されている場合、転送されたトラフィックは通常デフォルトで破棄されます。

ゾーン内の異なるインターフェイスまたはソース間でトラフィックを転送する方法を制御するには、ゾーンのターゲットを理解し、それに応じてゾーンのターゲットを設定する必要があります。

9.16.2. ゾーン内転送を使用したイーサネットと Wi-Fi ネットワーク間でのトラフィックの転送

ゾーン内転送を使用して、同じ **firewalld** ゾーン内のインターフェイスとソース間のトラフィックを転送することができます。この機能には次の利点があります。

- 有線デバイスと無線デバイスの間のシームレスな接続性 (**enp1s0** に接続されたイーサネットネットワークと **wlp0s20** に接続された Wi-Fi ネットワークの間でトラフィックを転送可能)
- 柔軟な作業環境のサポート
- ネットワーク内の複数のデバイスまたはユーザーがアクセスして使用できる共有リソース (プリンター、データベース、ネットワーク接続ストレージなど)
- 効率的な内部ネットワーク (スムーズな通信、レイテンシーの短縮、リソースへのアクセス性など)

この機能は、個々の **firewalld** ゾーンに対して有効にすることができます。

手順

1. カーネルでパケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2. ゾーン内転送を有効にするインターフェイスが **internal** ゾーンにのみ割り当てられていることを確認します。

```
# firewall-cmd --get-active-zones
```

3. 現在、インターフェイスが **internal** 以外のゾーンに割り当てられている場合は、以下のように再割り当てします。

```
# firewall-cmd --zone=internal --change-interface=interface_name --permanent
```

4. **enp1s0** および **wlp0s20** インターフェイスを **internal** ゾーンに追加します。

```
# firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5. ゾーン内転送を有効にします。

```
# firewall-cmd --zone=internal --add-forward
```

検証

以下の検証手順では、**nmap-ncat** パッケージが両方のホストにインストールされている必要があります。

1. ゾーン転送を有効にしたホストの **enp1s0** インターフェイスと同じネットワーク上にあるホストにログインします。
2. **ncat** で echo サービスを起動し、接続をテストします。

```
# ncat -e /usr/bin/cat -l 12345
```

3. **wlp0s20** インターフェイスと同じネットワークにあるホストにログインします。
4. **enp1s0** と同じネットワークにあるホスト上で実行している echo サーバーに接続します。

```
# ncat <other_host> 12345
```

5. 何かを入力して **Enter** を押します。テキストが返送されることを確認します。

関連情報

- **firewalld.zones(5)** の man ページ

9.17. RHEL システムロールを使用した FIREWALLD の設定

firewall システムロールを使用すると、一度に複数のクライアントに **firewalld** サービスを設定できます。この解決策は以下のとおりです。

- 入力設定が効率的なインターフェイスを提供する。
- 目的の **firewalld** パラメーターを 1 か所で保持する。

コントロールノードで **firewall** ロールを実行すると、システムロールは **firewalld** パラメーターをマネージドノードに即座に適用し、再起動後も維持されます。

9.17.1. RHEL システムロール firewall の概要

RHEL システムロールは、Ansible 自動化ユーティリティーのコンテンツセットです。このコンテンツは、Ansible 自動化ユーティリティーとともに、複数のシステムをリモートで管理するための一貫した設定インターフェイスを提供します。

firewalld サービスの自動設定に、RHEL システムロールからの **rhel-system-roles.firewall** ロールが導入されました。**rhel-system-roles** パッケージには、このシステムロールと参考ドキュメントも含まれます。

firewalld パラメーターを自動化された方法で 1 つ以上のシステムに適用するには、Playbook で **firewall** システムロール変数を使用します。Playbook は、テキストベースの YAML 形式で記述された 1 つ以上のプレイのリストです。

インベントリーファイルを使用して、Ansible が設定するシステムセットを定義できます。

firewall ロールを使用すると、以下のような異なる **firewalld** パラメーターを設定できます。

- ゾーン。
- パケットが許可されるサービス。
- ポートへのトラフィックアクセスの付与、拒否、または削除。
- ゾーンのポートまたはポート範囲の転送。

関連情報

- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリーの **README.md** ファイルおよび **README.html** ファイル
- Playbook の使用
- インベントリーの構築方法

9.17.2. RHEL システムロールを使用した firewalld 設定のリセット

firewall RHEL システムロールを使用すると、**firewalld** 設定をデフォルトの状態にリセットできます。**previous:replaced** パラメーターを変数リストに追加すると、システムロールは既存のユーザー定義の設定をすべて削除し、**firewalld** をデフォルトにリセットします。**previous:replaced** パラメーターを他の設定と組み合わせると、**firewall** ロールは新しい設定を適用する前に既存の設定をすべて削除します。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/reset-firewalld.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
  - name: Reset firewalld
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - previous: replaced
```

2. Playbook の構文を検証します。

```
# ansible-playbook ~/configure-ethernet-device-with-ethtoolcoalesce-settings.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
# ansible-playbook ~/reset-firewalld.yml
```

検証

- 管理対象ノードで **root** として次のコマンドを実行し、すべてのゾーンを確認します。

```
# firewall-cmd --list-all-zones
```

関連情報

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

9.17.3. RHEL システムロールを使用して、firewalld の着信トラフィックをあるローカルポートから別のローカルポートに転送する

firewall ロールを使用すると、複数の管理対象ホストで設定が永続化されるので **firewalld** パラメーターをリモートで設定できます。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。

- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/port_forwarding.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Forward incoming traffic on port 8080 to 443
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. Playbook の構文を検証します。

```
# ansible-playbook ~/port_forwarding.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
# ansible-playbook ~/port_forwarding.yml
```

検証

- 管理対象ホストで、**firewalld** 設定を表示します。

```
# firewall-cmd --list-forward-ports
```

関連情報

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

9.17.4. RHEL システムロールを使用した firewalld でのポートの管理

RHEL **firewall** システムロールを使用すると、着信トラフィックに対してローカルファイアウォールでポートを開くか、閉じて、再起動後に新しい設定を永続化できます。たとえば、HTTPS サービスの着信トラフィックを許可するようにデフォルトゾーンを設定できます。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している

- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/opening-a-port.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Allow incoming HTTPS traffic to the local host
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - port: 443/tcp
          service: http
          state: enabled
          runtime: true
          permanent: true
```

permanent: true オプションを使用すると、再起動後も新しい設定が維持されます。

2. Playbook の構文を検証します。

```
# ansible-playbook ~/opening-a-port.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
# ansible-playbook ~/opening-a-port.yml
```

検証

- 管理対象ノードで、**HTTPS** サービスに関連付けられた **443/tcp** ポートが開いていることを確認します。

```
# firewall-cmd --list-ports
443/tcp
```

関連情報

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

9.17.5. RHEL システムロールを使用した firewalld DMZ ゾーンの設定

システム管理者は、**firewall** システムロールを使用して、**enp1s0** インターフェイスで **dmz** ゾーンを設定し、ゾーンへの **HTTPS** トラフィックを許可できます。これにより、外部ユーザーが Web サーバーにアクセスできるようにします。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントには、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/configuring-a-dmz.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - zone: dmz
          interface: enp1s0
          service: https
          state: enabled
          runtime: true
          permanent: true
```

2. Playbook の構文を検証します。

```
# ansible-playbook ~/configuring-a-dmz.yml --syntax-check
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
# ansible-playbook ~/configuring-a-dmz.yml
```

検証

- 管理ノードで、**dmz** ゾーンに関する詳細情報を表示します。

```
# firewall-cmd --zone=dmz --list-all
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
```

関連情報

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

第10章 NFTABLES の使用

nftables フレームワークはパケットを分類し、**iptables**、**ip6tables**、**arptables**、**ebtables**、および **ipset** ユーティリティーの後継です。利便性、機能、パフォーマンスにおいて、以前のパケットフィルタリングツールに多くの改良が追加されました。以下に例を示します。

- 線形処理の代わりに組み込みルックアップテーブルを使用
- **IPv4** プロトコルおよび **IPv6** プロトコルに対する 1 つのフレームワーク
- 完全ルールセットのフェッチ、更新、および保存を行わず、すべてアトミックに適用されるルール
- ルールセットにおけるデバッグおよびトレースへの対応 (**nftrace**) およびトレースイベントの監視 (**nft** ツール)
- より統一されたコンパクトな構文、プロトコル固有の拡張なし
- サードパーティーのアプリケーション用 Netlink API

nftables フレームワークは、テーブルを使用してチェーンを保存します。このチェーンには、アクションを実行する個々のルールが含まれます。**nft** ユーティリティーは、以前のパケットフィルタリングフレームワークのツールをすべて置き換えます。**libmnl** ライブラリーを介して、**nftables** Netlink API との低レベルの対話に **libnftnl** ライブラリーを使用できます。

ルールセット変更が適用されていることを表示するには、**nft list ruleset** コマンドを使用します。これらのユーティリティーはテーブル、チェーン、ルール、セット、およびその他のオブジェクトを **nftables** ルールセットに追加するため、**nft flush ruleset** コマンドなどの **nftables** ルールセット操作は、**iptables** コマンドを使用してインストールされたルールセットに影響を与える可能性があることに注意してください。

10.1. IPTABLES から NFTABLES への移行

ファイアウォール設定が依然として **iptables** ルールを使用している場合は、**iptables** ルールを **nftables** に移行できます。

10.1.1. firewalld、nftables、または iptables を使用する場合

以下は、次のユーティリティーのいずれかを使用する必要があるシナリオの概要です。

- **firewalld**: 簡単な firewall のユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。
- **nftables**: **nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。
- **iptables**: Red Hat Enterprise Linux の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

さまざまなファイアウォール関連サービス (**firewalld**、**nftables**、または **iptables**) が相互に影響を与えないようにするには、RHEL ホストでそのうち 1 つだけを実行し、他のサービスを無効にします。

10.1.2. iptables および ip6tables ルールセットの nftables への変換

iptables-restore-translate ユーティリティーおよび **ip6tables-restore-translate** ユーティリティーを使用して、**iptables** および **ip6tables** ルールセットを **nftables** に変換します。

前提条件

- **nftables** パッケージおよび **iptables** パッケージがインストールされている。
- システムに **iptables** ルールおよび **ip6tables** ルールが設定されている。

手順

1. **iptables** ルールおよび **ip6tables** ルールをファイルに書き込みます。

```
# iptables-save >/root/iptables.dump
# ip6tables-save >/root/ip6tables.dump
```

2. ダンプファイルを **nftables** 命令に変換します。

```
# iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-from-iptables.nft
# ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-from-ip6tables.nft
```

3. 必要に応じて、生成された **nftables** ルールを手動で更新して、確認します。
4. **nftables** サービスが生成されたファイルをロードできるようにするには、以下を **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5. **iptables** サービスを停止し、無効にします。

```
# systemctl disable --now iptables
```

カスタムスクリプトを使用して **iptables** ルールを読み込んだ場合は、スクリプトが自動的に開始されなくなったことを確認し、再起動してすべてのテーブルをフラッシュします。

6. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

- **nftables** ルールセットを表示します。

```
# nft list ruleset
```

関連情報

- システムの起動時に nftables ルールの自動読み込み

10.1.3. 単一の iptables および ip6tables ルールセットの nftables への変換

Red Hat Enterprise Linux は、**iptables** ルールまたは **ip6tables** ルールを、**nftables** で同等のルールに変換する **iptables-translate** ユーティリティーおよび **ip6tables-translate** ユーティリティーを提供します。

前提条件

- **nftables** パッケージがインストールされている。

手順

- 以下のように、**iptables** または **ip6tables** の代わりに **iptables-translate** ユーティリティーまたは **ip6tables-translate** ユーティリティーを使用して、対応する **nftables** ルールを表示します。

```
# iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

拡張機能によっては変換機能がない場合もあります。このような場合には、ユーティリティーは、以下のように、前に **#** 記号が付いた未変換ルールを出力します。

```
# iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

関連情報

- **iptables-translate --help**

10.1.4. 一般的な iptables コマンドと nftables コマンドの比較

以下は、一般的な **iptables** コマンドと **nftables** コマンドの比較です。

- すべてのルールをリスト表示します。

iptables	nftables
iptables-save	**nft list ruleset**

- 特定のテーブルおよびチェーンをリスト表示します。

iptables	nftables
iptables -L	**nft list table ip filter**

iptables	nftables
iptables -L INPUT	**nft list chain ip filter INPUT**
iptables -t nat -L PREROUTING	**nft list chain ip nat PREROUTING**

nft コマンドは、テーブルおよびチェーンを事前に作成しません。これらは、ユーザーが手動で作成した場合にのみ存在します。

firewalld によって生成されたルールの一覧表示:

```
# nft list table inet firewalld
# nft list table ip firewalld
# nft list table ip6 firewalld
```

10.2. NFTABLES スクリプトの作成および実行

nftables フレームワークを使用する主な利点は、スクリプトの実行がアトミックであることです。つまり、システムがスクリプト全体を適用するか、エラーが発生した場合には実行を阻止することを意味します。これにより、ファイアウォールは常に一貫した状態になります。

さらに、**nftables** スクリプト環境を使用すると、次のことができます。

- コメントの追加
- 変数の定義
- 他のルールセットファイルの組み込み

nftables パッケージをインストールすると、Red Hat Enterprise Linux が自動的に ***.nft** スクリプトを **/etc/nftables/** ディレクトリーに作成します。このスクリプトは、さまざまな目的でテーブルと空のチェーンを作成するコマンドが含まれます。

10.2.1. 対応している nftables スクリプトの形式

nftables スクリプト環境では、次の形式でスクリプトを記述できます。

- **nft list ruleset** コマンドと同じ形式でルールセットが表示されます。

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
  chain example_chain {
    # Chain for incoming packets that drops all packets that
    # are not explicitly allowed by any rule in this chain
    type filter hook input priority 0; policy drop;

    # Accept connections to port 22 (ssh)
```

```
    tcp dport ssh accept
  }
}
```

- **nft** コマンドと同じ構文:

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table

# Create a chain for incoming packets that drops all packets
# that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0 ; policy drop ; }

# Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

10.2.2. nftables スクリプトの実行

nftables スクリプトは、**nft** ユーティリティーに渡すか、スクリプトを直接実行することで実行できます。

手順

- **nftables** スクリプトを **nft** ユーティリティーに渡して実行するには、次のコマンドを実行します。

```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

- **nftables** スクリプトを直接実行するには、次のコマンドを実行します。

  a. 1 回だけ実行する場合:

  i. スクリプトが以下のシバンシーケンスで始まることを確認します。

```
#!/usr/sbin/nft -f
```

重要

-f パラメーターを指定しないと、**nft** ユーティリティーはスクリプトを読み取らず、**Error: syntax error, unexpected newline, expecting string** を表示します。

  ii. 必要に応じて、スクリプトの所有者を **root** に設定します。

```
# chown root /etc/nftables/<example_firewall_script>.nft
```

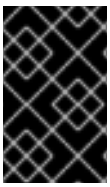
  iii. 所有者のスクリプトを実行ファイルに変更します。

```
# chmod u+x /etc/nftables/<example_firewall_script>.nft
```

b. スクリプトを実行します。

```
# /etc/nftables/<example_firewall_script>.nft
```

出力が表示されない場合は、システムがスクリプトを正常に実行します。



重要

nft はスクリプトを正常に実行しますが、ルールの配置やパラメーター不足、またはスクリプト内のその他の問題により、ファイアウォールが期待通りの動作を起こさない可能性があります。

関連情報

- **chown(1)** の man ページ
- **chmod(1)** の man ページ
- システムの起動時に nftables ルールの自動読み込み

10.2.3. nftables スクリプトでコメントの使用

nftables スクリプト環境は、**#** 文字の右側から行末までのすべてをコメントとして解釈します。

コメントは、行の先頭またはコマンドの横から開始できます。

```
...
# Flush the rule set
flush ruleset

add table inet example_table  # Create a table
...
```

10.2.4. nftables スクリプトでの変数の使用

nftables スクリプトで変数を定義するには、**define** キーワードを使用します。シングル値および匿名セットを変数に保存できます。より複雑なシナリオの場合は、セットまたは決定マップを使用します。

値を 1 つ持つ変数

以下の例は、値が **enp1s0** の **INET_DEV** という名前の変数を定義します。

```
define INET_DEV = enp1s0
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

匿名セットを含む変数

以下の例では、匿名セットを含む変数を定義します。

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```

注記

中括弧は、変数がセットを表していることを示すため、ルールで使用する場合は、特別なセマンティクスを持ちます。

関連情報

- nftables コマンドでのセットの使用
- nftables コマンドにおける決定マップの使用

10.2.5. nftables スクリプトへのファイルの追加

nftables スクリプト環境では、**include** ステートメントを使用して他のスクリプトを含めることができます。

絶対パスまたは相対パスのないファイル名のみを指定すると、**nftables** には、デフォルトの検索パスのファイルが含まれます。これは、Red Hat Enterprise Linux では **/etc** に設定されています。

例10.1 デフォルト検索ディレクトリーからのファイルを含む

デフォルトの検索ディレクトリーからファイルを指定するには、次のコマンドを実行します。

```
include "example.nft"
```

例10.2 ディレクトリーの *.nft ファイルをすべて含む

***.nft** で終わるすべてのファイルを **/etc/nftables/rulesets/** ディレクトリーに保存するには、次のコマンドを実行します。

```
include "/etc/nftables/rulesets/*.nft"
```

include ステートメントは、ドットで始まるファイルに一致しないことに注意してください。

関連情報

- **nft(8)** の man ページの **Include files** セクション

10.2.6. システムの起動時に nftables ルールの自動読み込み

systemd サービス **nftables** は、**/etc/sysconfig/nftables.conf** ファイルに含まれるファイアウォールスクリプトを読み込みます。

前提条件

- **nftables** スクリプトは、**/etc/nftables/** ディレクトリーに保存されます。

手順

1. **/etc/sysconfig/nftables.conf** ファイルを編集します。
   - **nftables** パッケージのインストールで **/etc/nftables/** に作成された ***.nft** スクリプトを変更した場合は、これらのスクリプトの **include** ステートメントのコメントを解除します。
   - 新しいスクリプトを作成した場合は、**include** ステートメントを追加してこれらのスクリプトを含めます。たとえば、**nftables** サービスの起動時に **/etc/nftables/example.nft** スクリプトを読み込むには、以下を追加します。

     ```
     include "/etc/nftables/_example_.nft"
     ```

2. オプション: **nftables** サービスを開始して、システムを再起動せずにファイアウォールルールを読み込みます。

   ```
   # systemctl start nftables
   ```

3. **nftables** サービスを有効にします。

   ```
   # systemctl enable nftables
   ```

関連情報

- 対応している nftables スクリプトの形式

10.3. NFTABLES テーブル、チェーン、およびルールの作成および管理

nftables ルールセットを表示して管理できます。

10.3.1. nftables テーブルの基本

nftables のテーブルは、チェーン、ルール、セットなどのオブジェクトを含む名前空間です。

各テーブルにはアドレスファミリーが割り当てられている必要があります。アドレスファミリーは、このテーブルが処理するパケットタイプを定義します。テーブルを作成する際に、以下のいずれかのアドレスファミリーを設定できます。

- **ip** - IPv4 パケットのみと一致します。アドレスファミリーを指定しないと、これがデフォルトになります。
- **ip6** - IPv6 パケットのみと一致します。
- **inet** - IPv4 パケットと IPv6 パケットの両方と一致します。
- **arp**: IPv4 アドレス解決プロトコル (ARP) パケットと一致します。

- **bridge**: ブリッジデバイスを通過するパケットに一致します。
- **netdev**: ingress からのパケットに一致します。

テーブルを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
}
```

- シェルスクリプトで、以下を使用します。

```
nft add table <table_address_family> <table_name>
```

10.3.2. nftables チェーンの基本

テーブルは、ルールのコンテナーであるチェーンで構成されます。次の 2 つのルールタイプが存在します。

- **ベースチェーン**: ネットワークスタックからのパケットのエントリーポイントとしてベースチェーンを使用できます。
- **通常のチェーン**: **jump** ターゲットとして通常のチェーンを使用し、ルールをより適切に整理できます。

ベースチェーンをテーブルに追加する場合に使用する形式は、ファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority>
    policy <policy> ;
  }
}
```

- シェルスクリプトで、以下を使用します。

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook <hook> priority <priority> \; policy <policy> \; }
```

シェルがセミコロンをコマンドの最後として解釈しないようにするには、セミコロンの前にエスケープ文字 \ を配置します。

どちらの例でも、**ベースチェーン** を作成します。**通常のチェーン** を作成する場合、中括弧内にパラメーターを設定しないでください。

チェーンタイプ
チェーンタイプとそれらを使用できるアドレスファミリーとフックの概要を以下に示します。

型	アドレスファミリー	フック	説明
filter	all	all	標準のチェーンタイプ
nat	**ip**、**ip6**、**inet**	**prerouting**、**input**、**output**、**postrouting**	このタイプのチェーンは、接続追跡エントリーに基づいてネイティブアドレス変換を実行します。最初のパケットのみがこのチェーンタイプをトラバースします。
ルート	**ip**、**ip6**	**出力 (output)**	このチェーンタイプを通過する許可済みパケットは、IP ヘッダーの関連部分が変更された場合に、新しいルートルックアップを引き起こします。

チェーンの優先度
priority パラメーターは、パケットが同じフック値を持つチェーンを通過する順序を指定します。このパラメーターは、整数値に設定することも、標準の priority 名を使用することもできます。

以下のマトリックスは、標準的な priority 名とその数値の概要、それらを使用できるファミリーおよびフックの概要です。

テキストの値	数値	アドレスファミリー	フック
raw	**-300**	**ip**、**ip6**、**inet**	all
mangle	**-150**	**ip**、**ip6**、**inet**	all
dstnat	**-100**	**ip**、**ip6**、**inet**	**prerouting**
	-300	**bridge**	**prerouting**
filter	**0**	**ip**、**ip6**、**inet**、**arp**、**netdev**	all
	-200	**bridge**	all
security	**50**	**ip**、**ip6**、**inet**	all
srcnat	**100**	**ip**、**ip6**、**inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**出力 (output)**

チェーンポリシー
チェーンポリシーは、このチェーンのルールでアクションが指定されていない場合に、**nftables** がパケットを受け入れるかドロップするかを定義します。チェーンには、以下のいずれかのポリシーを設定できます。

- **accept** (デフォルト)
- **drop**

10.3.3. nftables ルールの基本

ルールは、このルールを含むチェーンを渡すパケットに対して実行するアクションを定義します。ルールに一致する式も含まれる場合、**nftables** は、以前の式がすべて適用されている場合にのみアクションを実行します。

チェーンにルールを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority> ; policy <policy> ;
      <rule>
  }
}
```

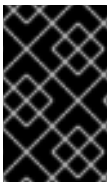
- シェルスクリプトで、以下を使用します。

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

このシェルコマンドは、チェーンの最後に新しいルールを追加します。チェーンの先頭にルールを追加する場合は、**nft add** の代わりに **nft insert** コマンドを使用します。

10.3.4. nft コマンドを使用したテーブル、チェーン、ルールの管理

コマンドラインまたはシェルスクリプトで **nftables** ファイアウォールを管理するには、**nft** ユーティリティーを使用します。



重要

この手順のコマンドは通常のワークフローを表しておらず、最適化されていません。この手順では、**nft** コマンドを使用して、一般的なテーブル、チェーン、およびルールを管理する方法を説明します。

手順

1. テーブルが IPv4 パケットと IPv6 パケットの両方を処理できるように、**inet** アドレスファミリーを使用して **nftables_svc** という名前のテーブルを作成します。

```
# nft add table inet nftables_svc
```

2. 受信ネットワークトラフィックを処理する **INPUT** という名前のベースチェーンを **inet nftables_svc** テーブルに追加します。

```
# nft add chain inet nftables_svc INPUT { type filter hook input priority filter \; policy accept \; }
```

シェルがセミコロンをコマンドの最後として解釈しないようにするには、\ 文字を使用してセミコロンをエスケープします。

3. **INPUT** チェーンにルールを追加します。たとえば、**INPUT** チェーンの最後のルールとして、ポート 22 および 443 で着信 TCP トラフィックを許可し、Internet Control Message Protocol (ICMP)ポートに到達できないメッセージで他の着信トラフィックを拒否します。

```
# nft add rule inet nftables_svc INPUT tcp dport 22 accept
# nft add rule inet nftables_svc INPUT tcp dport 443 accept
# nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

ここで示されたように **nft add rule** コマンドを実行すると、**nft** はコマンド実行と同じ順序でルールをチェーンに追加します。

4. ハンドルを含む現在のルールセットを表示します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

5. ハンドル 3 で既存ルールの前にルールを挿入します。たとえば、ポート 636 で TCP トラフィックを許可するルールを挿入するには、以下を入力します。

```
# nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6. ハンドル 3 で、既存ルールの後ろにルールを追加します。たとえば、ポート 80 で TCP トラフィックを許可するルールを追加するには、以下を入力します。

```
# nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7. ハンドルでルールセットを再表示します。後で追加したルールが指定の位置に追加されていることを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    tcp dport 80 accept # handle 6
    reject # handle 4
  }
}
```

8. ハンドル 6 でルールを削除します。

```
# nft delete rule inet nftables_svc INPUT handle 6
```

ルールを削除するには、ハンドルを指定する必要があります。

9. ルールセットを表示し、削除されたルールがもう存在しないことを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

10. **INPUT** チェーンから残りのルールをすべて削除します。

```
# nft flush chain inet nftables_svc INPUT
```

11. ルールセットを表示し、**INPUT** チェーンが空であることを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
    type filter hook input priority filter; policy accept
  }
}
```

12. **INPUT** チェーンを削除します。

```
# nft delete chain inet nftables_svc INPUT
```

このコマンドを使用して、まだルールが含まれているチェーンを削除することもできます。

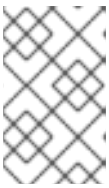
13. ルールセットを表示し、**INPUT** チェーンが削除されたことを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
}
```

14. **nftables_svc** テーブルを削除します。

```
# nft delete table inet nftables_svc
```

このコマンドを使用して、まだルールが含まれているテーブルを削除することもできます。



注記

ルールセット全体を削除するには、個別のコマンドですべてのルール、チェーン、およびテーブルを手動で削除するのではなく、**nft flush ruleset** コマンドを使用します。

関連情報

nft(8) の man ページ

10.4. NFTABLES を使用した NAT の設定

nftables を使用すると、以下のネットワークアドレス変換 (NAT) タイプを設定できます。

- マスカレーディング
- ソース NAT (SNAT)
- 宛先 NAT (DNAT)
- リダイレクト

重要

iifname パラメーターおよび **oifname** パラメーターでは実インターフェイス名のみを使用でき、代替名 (**altname**) には対応していません。

10.4.1. NAT タイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダー (ISP) は、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードと SNAT は互いに非常に似ています。相違点は次のとおりです。

- マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。
- SNAT は、パケットのソース IP アドレスを指定された IP に設定し、出力インターフェイスの IP アドレスを動的に検索しません。そのため、SNAT の方がマスカレードよりも高速です。出力インターフェイスが固定 IP アドレスを使用する場合は、SNAT を使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、チェーンフックに応じてパケットをローカルマシンにリダイレクトする DNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

10.4.2. nftables を使用したマスカレードの設定

マスカレードを使用すると、ルーターは、インターフェイスを介して送信されるパケットのソース IP を、インターフェイスの IP アドレスに動的に変更できます。これは、インターフェイスに新しい IP が割り当てられている場合に、**nftables** はソース IP の置き換え時に新しい IP を自動的に使用することを意味します。

ens3 インターフェイスを介してホストから出るパケットの送信元 IP を、**ens3** で設定された IP に置き換えます。

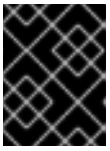
手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

prerouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、着信パケット返信に一致するようにこのチェーンが必要になります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **postrouting** チェーンに、**ens3** インターフェイスの出力パケットに一致するルールを追加します。

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

10.4.3. nftables を使用したソース NAT の設定

ルーターでは、ソース NAT (SNAT) を使用して、インターフェイスを介して特定の IP アドレスに送信するパケットの IP を変更できます。次に、ルーターは送信パケットのソース IP を置き換えます。

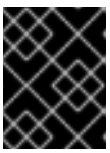
手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **ens3** を介した発信パケットのソース IP を **192.0.2.1** に置き換えるルールを **postrouting** チェーンに追加します。

```
# nft add rule nat postrouting oifname "ens3" snat to 192.0.2.1
```

関連情報

- 特定のローカルポートで着信パケットを別のホストに転送

10.4.4. nftables を使用した宛先 NAT の設定

宛先 NAT (DNAT)を使用すると、ルーター上のトラフィックをインターネットから直接アクセスできないホストにリダイレクトできます。

たとえば、DNAT を使用すると、ルーターはポート **80** および **443** に送信された受信トラフィックを、IP アドレス **192.0.2.1** の Web サーバーにリダイレクトします。

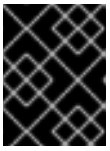
手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **prerouting** チェーンに、ルーターの **ens3** インターフェイスのポート **80** および **443** への受信トラフィックを、IP アドレス **192.0.2.1** の Web サーバーにリダイレクトするルールを追加します。

```
# nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4. 環境に応じて、SNAT ルールまたはマスカレードルールを追加して、Web サーバーから返されるパケットのソースアドレスを送信者に変更します。

   a. **ens3** インターフェイスが動的 IP アドレスを使用している場合は、マスカレードルールを追加します。

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

   b. **ens3** インターフェイスが静的 IP アドレスを使用する場合は、SNAT ルールを追加します。たとえば、**ens3** が IP アドレス **198.51.100.1** を使用している場合は、以下のようになります。

```
# nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

関連情報

- NAT タイプ

10.4.5. nftables を使用したリダイレクトの設定

redirect 機能は、チェーンフックに応じてパケットをローカルマシンにリダイレクトする宛先ネットワークアドレス変換 (DNAT) の特殊なケースです。

たとえば、ローカルホストのポート **22** に送信された着信および転送されたトラフィックを **2222** ポートにリダイレクトすることができます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに **prerouting** チェーンを追加します。

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **22** ポートの着信トラフィックを **2222** ポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule nat prerouting tcp dport 22 redirect to 2222
```

関連情報

- NAT タイプ

10.4.6. nftables を使用したフローテーブルの設定

nftables ユーティリティーは、**netfilter** フレームワークを使用してネットワークトラフィックにネットワークアドレス変換 (NAT) を提供し、高速パス機能ベースの **flowtable** メカニズムを提供してパケット転送を高速化します。

フローテーブルメカニズムには次の機能があります。

- 接続追跡を使用して、従来のパケット転送パスをバイパスします。
- 従来のパケット処理をバイパスすることで、ルーティングテーブルの再参照を回避します。
- TCP および UDP プロトコルでのみ動作します。
- ハードウェアに依存しないソフトウェア高速パスです。

手順

1. **inet** ファミリーの **example-table** テーブルを追加します。

```
# nft add table inet <example-table>
```

2. 優先度タイプとして **ingress** フックと **filter** を含む **example-flowtable** フローテーブルを追加します。

```
# nft add flowtable inet <example-table> <example-flowtable> { hook ingress priority filter \; devices = { enp1s0, enp7s0 } \; }
```

3. **example-forwardchain** フローをパケット処理テーブルからフローテーブルに追加します。

```
# nft add chain inet <example-table> <example-forwardchain> { type filter hook forward priority filter \; }
```

このコマンドは、**forward** フックと **filter** 優先度を備えた **filter** タイプのフローテーブルを追加します。

4. **established** 接続追跡状態を含むルールを追加して、**example-flowtable** フローをオフロードします。

```
# nft add rule inet <example-table> <example-forwardchain> ct state established flow add @<example-flowtable>
```

検証

- **example-table** のプロパティーを確認します。

```
# nft list table inet <example-table>
table inet example-table {
    flowtable example-flowtable {
      hook ingress priority filter
          devices = { enp1s0, enp7s0 }
    }

    chain example-forwardchain {
type filter hook forward priority filter; policy accept;
ct state established flow add @example-flowtable
   }
}
```

関連情報

- **nft(8)** の man ページ

10.5. NFTABLES コマンドでのセットの使用

nftables フレームワークは、セットをネイティブに対応します。たとえば、ルールが複数の IP アドレス、ポート番号、インターフェイス、またはその他の一致基準に一致する必要がある場合など、セットを使用できます。

10.5.1. nftables での匿名セットの使用

匿名セットには、ルールで直接使用する **{ 22, 80, 443 }** などの中括弧で囲まれたコンマ区切りの値が含まれます。IP アドレスやその他の一致基準にも匿名セットを使用できます。

匿名セットの欠点は、セットを変更する場合はルールを置き換える必要があることです。動的なソリューションの場合は、nftables で名前付きセットの使用 で説明されているように名前付きセットを使用します。

前提条件

- **inet** ファミリーに **example_chain** チェーンおよび **example_table** テーブルがある。

手順

1. たとえば、ポート **22**、**80**、および **443** に着信トラフィックを許可するルールを、**example_table** の **example_chain** に追加するには、次のコマンドを実行します。

```
# nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2. オプション: **example_table** ですべてのチェーンとそのルールを表示します。

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport { ssh, http, https } accept
  }
}
```

10.5.2. nftables で名前付きセットの使用

nftables フレームワークは、変更可能な名前付きセットに対応します。名前付きセットは、テーブル内の複数のルールで使用できる要素のリストまたは範囲です。匿名セットに対する別の利点として、セットを使用するルールを置き換えることなく、名前付きセットを更新できます。

名前付きセットを作成する場合は、セットに含まれる要素のタイプを指定する必要があります。以下のタイプを設定できます。

- **192.0.2.1** や **192.0.2.0/24** など、IPv4 アドレスまたは範囲を含むセットの場合は **ipv4_addr**。
- **2001:db8:1::1** や **2001:db8:1::1/64** など、IPv6 アドレスまたは範囲を含むセットの場合は **ipv6_addr**。
- **52:54:00:6b:66:42** など、メディアアクセス制御 (MAC) アドレスの一覧を含むセットの場合は **ether_addr**。
- **tcp** など、インターネットプロトコルタイプの一覧が含まれるセットの場合は **inet_proto**。
- **ssh** など、インターネットサービスの一覧を含むセットの場合は **inet_service**。
- パケットマークの一覧を含むセットの場合は **mark**。パケットマークは、任意の 32 ビットの正の整数値 (**0** から **2147483647**) にすることができます。

前提条件

- **example_chain** チェーンと **example_table** テーブルが存在する。

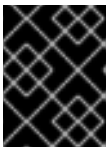
手順

1. 空のファイルを作成します。以下の例では、IPv4 アドレスのセットを作成します。
   - 複数の IPv4 アドレスを格納することができるセットを作成するには、次のコマンドを実行します。

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; }
     ```

   - IPv4 アドレス範囲を保存できるセットを作成するには、次のコマンドを実行します。

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
     ```

     重要

     シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

2. オプション: セットを使用するルールを作成します。たとえば、次のコマンドは、**example_set** の IPv4 アドレスからのパケットをすべて破棄するルールを、**example_table** の **example_chain** に追加します。

   ```
   # nft add rule inet example_table example_chain ip saddr @example_set drop
   ```

   example_set が空のままなので、ルールには現在影響がありません。

3. IPv4 アドレスを **example_set** に追加します。
   - 個々の IPv4 アドレスを保存するセットを作成する場合は、次のコマンドを実行します。

     ```
     # nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
     ```

   - IPv4 範囲を保存するセットを作成する場合は、次のコマンドを実行します。

     ```
     # nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
     ```

     IP アドレス範囲を指定する場合は、上記の例の **192.0.2.0/24** のように、CIDR (Classless Inter-Domain Routing) 表記を使用することもできます。

10.5.3. 関連情報

- **nft(8)** の man ページの **Sets** セクション

10.6. NFTABLES コマンドにおける決定マップの使用

ディクショナリーとしても知られている決定マップにより、**nft** は一致基準をアクションにマッピングすることで、パケット情報に基づいてアクションを実行できます。

10.6.1. nftables での匿名マップの使用

匿名マップは、ルールで直接使用する **{ match_criteria : action }** ステートメントです。ステートメントには、複数のコンマ区切りマッピングを含めることができます。

匿名マップの欠点は、マップを変更する場合には、ルールを置き換える必要があることです。動的なソリューションの場合は、nftables での名前付きマップの使用 で説明されているように名前付きマップを使用します。

たとえば、匿名マップを使用して、IPv4 プロトコルおよび IPv6 プロトコルの TCP パケットと UDP パケットの両方を異なるチェーンにルーティングし、着信 TCP パケットと UDP パケットを個別にカウントできます。

手順

1. 新しいテーブルを作成します。

```
# nft add table inet example_table
```

2. **example_table** に **tcp_packets** チェーンを作成します。

```
# nft add chain inet example_table tcp_packets
```

3. このチェーンのトラフィックをカウントする **tcp_packets** にルールを追加します。

```
# nft add rule inet example_table tcp_packets counter
```

4. **example_table** で **udp_packets** チェーンを作成します。

```
# nft add chain inet example_table udp_packets
```

5. このチェーンのトラフィックをカウントする **udp_packets** にルールを追加します。

```
# nft add rule inet example_table udp_packets counter
```

6. 着信トラフィックのチェーンを作成します。たとえば、**example_table** に、着信トラフィックをフィルタリングする **incoming_traffic** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7. 匿名マップを持つルールを **incoming_traffic** に追加します。

```
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

匿名マップはパケットを区別し、プロトコルに基づいて別のカウンターチェーンに送信します。

8. トラフィックカウンターの一覧を表示する場合は、**example_table** を表示します。

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
```

```
    counter packets 36379 bytes 2103816
  }

  chain udp_packets {
    counter packets 10 bytes 1559
  }

  chain incoming_traffic {
    type filter hook input priority filter; policy accept;
    ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
  }
}
```

tcp_packets チェーンおよび **udp_packets** チェーンのカウンターは、受信パケットとバイトの両方を表示します。

10.6.2. nftables での名前付きマップの使用

nftables フレームワークは、名前付きマップに対応します。テーブルの複数のルールでこのマップを使用できます。匿名マップに対する別の利点は、名前付きマップを使用するルールを置き換えることなく、名前付きマップを更新できることです。

名前付きマップを作成する場合は、要素のタイプを指定する必要があります。

- 一致する部分に **192.0.2.1** などの IPv4 アドレスが含まれるマップの場合は **ipv4_addr**。
- 一致する部分に **2001:db8:1::1** などの IPv6 アドレスが含まれるマップの場合は **ipv6_addr**。
- **52:54:00:6b:66:42** などのメディアアクセス制御 (MAC) アドレスを含むマップの場合は **ether_addr**。
- 一致する部分に **tcp** などのインターネットプロトコルタイプが含まれるマップの場合は **inet_proto**。
- 一致する部分に **ssh** や **22** などのインターネットサービス名のポート番号が含まれるマップの場合は **inet_service**。
- 一致する部分にパケットマークが含まれるマップの場合は **mark**。パケットマークは、任意の正の 32 ビットの整数値 (**0** ~ **2147483647**) にできます。
- 一致する部分にカウンターの値が含まれるマップの場合は **counter**。カウンター値は、正の値の 64 ビットであれば任意の値にすることができます。
- 一致する部分にクォータ値が含まれるマップの場合は **quota**。クォータの値は、64 ビットの整数値にできます。

たとえば、送信元 IP アドレスに基づいて着信パケットを許可または拒否できます。名前付きマップを使用すると、このシナリオを設定するのに必要なルールは 1 つだけで、IP アドレスとアクションがマップに動的に保存されます。

手順

1. テーブルを作成します。たとえば、IPv4 パケットを処理する **example_table** という名前のテーブルを作成するには、次のコマンドを実行します。

```
# nft add table ip example_table
```

2. チェーンを作成します。たとえば、**example_table** に、**example_chain** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```

重要

シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

3. 空のマップを作成します。たとえば、IPv4 アドレスのマッピングを作成するには、次のコマンドを実行します。

```
# nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4. マップを使用するルールを作成します。たとえば、次のコマンドは、両方とも **example_map** で定義されている IPv4 アドレスにアクションを適用するルールを、 **example_table** の **example_chain** に追加します。

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. IPv4 アドレスと対応するアクションを **example_map** に追加します。

```
# nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

以下の例では、IPv4 アドレスのアクションへのマッピングを定義します。上記で作成したルールと組み合わせて、ファイアウォールは **192.0.2.1** からのパケットを許可し、**192.0.2.2** からのパケットを破棄します。

6. オプション: 別の IP アドレスおよび action ステートメントを追加してマップを拡張します。

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. オプション: マップからエントリーを削除します。

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. オプション: ルールセットを表示します。

```
# nft list ruleset
table ip example_table {
  map example_map {
    type ipv4_addr : verdict
    elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
  }

  chain example_chain {
    type filter hook input priority filter; policy accept;
    ip saddr vmap @example_map
  }
}
```

10.6.3. 関連情報

- **nft(8)** の man ページの **Maps** セクション

10.7. 例: NFTABLES スクリプトを使用した LAN および DMZ の保護

RHEL ルーターで **nftables** フレームワークを使用して、内部 LAN 内のネットワーククライアントと DMZ の Web サーバーを、インターネットやその他のネットワークからの不正アクセスから保護するファイアウォールスクリプトを作成およびインストールします。

重要

この例はデモ目的専用で、特定の要件があるシナリオを説明しています。

ファイアウォールスクリプトは、ネットワークインフラストラクチャーとセキュリティー要件に大きく依存します。この例を使用して、独自の環境用のスクリプトを作成する際に **nftables** ファイアウォールの概念を理解してください。

10.7.1. ネットワークの状態

この例のネットワークは、以下の条件下にあります。

- ルーターは以下のネットワークに接続されています。
  - インターフェイス **enp1s0**を介したインターネット
  - インターフェイス **enp7s0**を介した内部 LAN
  - **enp8s0**までの DMZ
- ルーターのインターネットインターフェイスには、静的 IPv4 アドレス (**203.0.113.1**) と IPv6 アドレス (**2001:db8:a::1**) の両方が割り当てられています。
- 内部 LAN のクライアントは **10.0.0.0/24** の範囲のプライベート IPv4 アドレスのみを使用します。その結果、LAN からインターネットへのトラフィックには、送信元ネットワークアドレス変換 (SNAT) が必要です。
- 内部 LAN の管理者用 PC は、IP アドレス **10.0.0.100** および **10.0.0.200** を使用します。
- DMZ は、**198.51.100.0/24** および **2001:db8:b::/56** の範囲のパブリック IP アドレスを使用します。
- DMZ の Web サーバーは、IP アドレス **198.51.100.5** および **2001:db8:b::5** を使用します。
- ルーターは、LAN および DMZ 内のホストのキャッシング DNS サーバーとして機能します。

10.7.2. ファイアウォールスクリプトのセキュリティー要件

以下は、サンプルネットワークにおける **nftables** ファイアウォールの要件です。

- ルーターは以下を実行できる必要があります。
  - DNS クエリーを再帰的に解決します。
  - ループバックインターフェイスですべての接続を実行します。

- 内部 LAN のクライアントは以下を実行できる必要があります。
  - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
  - DMZ の HTTPS サーバーにアクセスします。
  - インターネット上の任意の HTTPS サーバーにアクセスします。
- 管理者用の PC は、SSH を使用してルーターと DMZ 内のすべてのサーバーにアクセスできる必要があります。
- DMZ の Web サーバーは以下を実行できる必要があります。
  - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
  - インターネット上の HTTPS サーバーにアクセスして更新をダウンロードします。
- インターネット上のホストは以下を実行できる必要があります。
  - DMZ の HTTPS サーバーにアクセスします。
- さらに、以下のセキュリティー要件が存在します。
  - 明示的に許可されていない接続の試行はドロップする必要があります。
  - ドロップされたパケットはログに記録する必要があります。

10.7.3. ドロップされたパケットをファイルにロギングするための設定

デフォルトでは、**systemd** は、ドロップされたパケットなどのカーネルメッセージをジャーナルに記録します。さらに、このようなエントリーを別のファイルに記録するように **rsyslog** サービスを設定することもできます。ログファイルが無限に大きくならないようにするために、ローテーションポリシーを設定します。

前提条件

- **rsyslog** パッケージがインストールされている。
- **rsyslog** サービスが実行されている。

手順

1. 以下の内容で **/etc/rsyslog.d/nftables.conf** ファイルを作成します。

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

   この設定を使用すると、**rsyslog** サービスはドロップされたパケットを **/var/log/messages** ではなく **/var/log/nftables.log** ファイルに記録します。

2. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

3. サイズが 10 MB を超える場合は、以下の内容で **/etc/logrotate.d/nftables** ファイルを作成し、**/var/log/nftables.log** をローテーションします。

```
/var/log/nftables.log {
  size +10M
  maxage 30
  sharedscripts
  postrotate
    /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
  endscript
}
```

maxage 30 設定は、次のローテーション操作中に **logrotate** が 30 日経過したローテーション済みログを削除することを定義します。

関連情報

- **rsyslog.conf(5)** の man ページ
- **logrotate(8)** の man ページ

10.7.4. nftables スクリプトの作成とアクティブ化

この例は、RHEL ルーターで実行され、DMZ の内部 LAN および Web サーバーのクライアントを保護する **nftables** ファイアウォールスクリプトです。この例で使用されているネットワークとファイアウォールの要件について、詳しくはファイアウォールスクリプトの ネットワークの状態 および ファイアウォールスクリプトのセキュリティー要件 を参照してください。

警告

この **nftables** ファイアウォールスクリプトは、デモ専用です。お使いの環境やセキュリティー要件に適応させて使用してください。

前提条件

- ネットワークは、ネットワークの状態 で説明されているとおりに設定されます。

手順

1. 以下の内容で **/etc/nftables/firewall.nft** スクリプトを作成します。

```
# Remove all rules
flush ruleset


# Table for both IPv4 and IPv6 rules
table inet nftables_svc {

  # Define variables for the interface name
  define INET_DEV = enp1s0
  define LAN_DEV  = enp7s0
  define DMZ_DEV  = enp8s0
```

```
# Set with the IPv4 addresses of admin PCs
set admin_pc_ipv4 {
  type ipv4_addr
  elements = { 10.0.0.100, 10.0.0.200 }
}


# Chain for incoming trafic. Default policy: drop
chain INPUT {
  type filter hook input priority filter
  policy drop

  # Accept packets in established and related state, drop invalid packets
  ct state vmap { established:accept, related:accept, invalid:drop }

  # Accept incoming traffic on loopback interface
  iifname lo accept

  # Allow request from LAN and DMZ to local DNS server
  iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

  # Allow admins PCs to access the router using SSH
  iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

  # Last action: Log blocked packets
  # (packets that were not accepted in previous rules in this chain)
  log prefix "nft drop IN : "
}


# Chain for outgoing traffic. Default policy: drop
chain OUTPUT {
  type filter hook output priority filter
  policy drop

  # Accept packets in established and related state, drop invalid packets
  ct state vmap { established:accept, related:accept, invalid:drop }

  # Accept outgoing traffic on loopback interface
  oifname lo accept

  # Allow local DNS server to recursively resolve queries
  oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

  # Last action: Log blocked packets
  log prefix "nft drop OUT: "
}


# Chain for forwarding traffic. Default policy: drop
chain FORWARD {
  type filter hook forward priority filter
  policy drop

  # Accept packets in established and related state, drop invalid packets
```

```
    ct state vmap { established:accept, related:accept, invalid:drop }

    # IPv4 access from LAN and internet to the HTTPS server in the DMZ
    iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp dport
443 accept

    # IPv6 access from internet to the HTTPS server in the DMZ
    iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443 accept

    # Access from LAN and DMZ to HTTPS servers on the internet
    iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

    # Last action: Log blocked packets
    log prefix "nft drop FWD: "
  }


  # Postrouting chain to handle SNAT
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;

    # SNAT for IPv4 traffic from LAN to internet
    iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
  }
}
```

2. **/etc/nftables/firewall.nft** スクリプトを **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/firewall.nft"
```

3. IPv4 転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

1. オプション: **nftables** ルールセットを確認します。

```
# nft list ruleset
...
```

2. ファイアウォールが阻止するアクセスの実行を試みます。たとえば、DMZ から SSH を使用してルーターにアクセスします。

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. ロギング設定に応じて、以下を検索します。

- ブロックされたパケットの **systemd** ジャーナル:

```
# journalctl -k -g "nft drop"
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

- ブロックされたパケットの **/var/log/nftables.log** ファイル:

```
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

10.8. NFTABLES を使用したポート転送の設定

ポート転送を使用すると、管理者は特定の宛先ポートに送信されたパケットを、別のローカルまたはリモートポートに転送できます。

たとえば、Web サーバーにパブリック IP アドレスがない場合は、ファイアウォールの **80** ポートおよび **443** ポートの着信パケットを Web サーバーに転送するファイアウォールのポート転送ルールを設定できます。このファイアウォールルールを使用すると、インターネットのユーザーは、ファイアウォールの IP またはホスト名を使用して Web サーバーにアクセスできます。

10.8.1. 着信パケットの別のローカルポートへの転送

nftables を使用してパケットを転送できます。たとえば、ポート **8022** の着信 IPv4 パケットを、ローカルシステムのポート **22** に転送できます。

手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

```
# nft add table ip nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
```

注記

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **8022** ポートの着信パケットを、ローカルポート **22** にリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

10.8.2. 特定のローカルポートで着信パケットを別のホストに転送

宛先ネットワークアドレス変換 (DNAT) ルールを使用して、ローカルポートの着信パケットをリモートホストに転送できます。これにより、インターネット上のユーザーは、プライベート IP アドレスを持つホストで実行しているサービスにアクセスできるようになります。

たとえば、ローカルポート **443** の着信 IPv4 パケットを、IP アドレス **192.0.2.1** を持つリモートシステムの同じポート番号に転送できます。

前提条件

- パケットを転送するシステムに **root** ユーザーとしてログインしている。

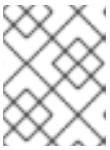
手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

```
# nft add table ip nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain ip nat postrouting { type nat hook postrouting priority 100 \; }
```



注記

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **443** ポートの着信パケットを **192.0.2.1** 上の同じポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 出力トラフィックをマスカレードするルールを **postrouting** チェーンに追加します。

```
# nft add rule ip nat postrouting daddr 192.0.2.1 masquerade
```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

10.9. NFTABLES を使用した接続の量の制限

nftables を使用して、接続の数を制限したり、一定の数の接続の確立を試みる IP アドレスをブロックして、システムリソースを過剰に使用されないようにします。

10.9.1. nftables を使用した接続数の制限

nft ユーティリティーの **ct count** パラメーターを使用すると、管理者は接続数を制限することができます。

前提条件

- **example_table** にベースの **example_chain** が存在する。

手順

1. IPv4 アドレスの動的セットを作成します。

```
# nft add set inet example_table example_meter { type ipv4_addr\; flags dynamic \;}
```

2. IPv4 アドレスから SSH ポート (22) への 2 つの同時接続のみを許可し、同じ IP からのすべての接続を拒否するルールを追加します。

```
# nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

3. オプション: 前の手順で作成したセットを表示します。

```
# nft list set inet example_table example_meter
table inet example_table {
  meter example_meter {
    type ipv4_addr
    size 65535
    elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2  }
  }
}
```

elements エントリーは、現時点でルールに一致するアドレスを表示します。この例では、**elements** は、SSH ポートへのアクティブな接続がある IP アドレスを一覧表示します。出力には、アクティブな接続の数を表示しないため、接続が拒否された場合は表示されないことに注意してください。

10.9.2.1 分以内に新しい着信 TCP 接続を 11 個以上試行する IP アドレスのブロック

1 分以内に 11 個以上の IPv4 TCP 接続を確立しているホストを一時的にブロックできます。

手順

1. **ip** アドレスファミリーを使用して **filter** テーブルを作成します。

```
# nft add table ip filter
```

2. **input** チェーンを **filter** テーブルに追加します。

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. 1 分以内に 10 を超える TCP 接続を確立しようとするソースアドレスからのすべてのパケットを破棄するルールを追加します。

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip saddr timeout 5m limit rate over 10/minute } drop
```

timeout 5m パラメーターは、**nftables** が、メーターが古いエントリーで一杯にならないように、5 分後にエントリーを自動的に削除することを定義します。

検証

- メーターのコンテンツを表示するには、以下のコマンドを実行します。

```
# nft list meter ip filter ratemeter
table ip filter {
  meter ratemeter {
    type ipv4_addr
    size 65535
    flags dynamic,timeout
    elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
  }
}
```

10.10. NFTABLES ルールのデバッグ

nftables フレームワークは、管理者がルールをデバッグし、パケットがそれに一致するかどうかを確認するためのさまざまなオプションを提供します。

10.10.1. カウンターによるルールの作成

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- 既存のルールにカウンターを追加する手順の詳細は、**Configuring and managing networking** の Adding a counter to an existing rule を参照してください。

前提条件

- ルールを追加するチェーンが存在する。

手順

1. **counter** パラメーターで新しいルールをチェーンに追加します。以下の例では、ポート 22 で TCP トラフィックを許可し、このルールに一致するパケットとトラフィックをカウントするカウンターを使用するルールを追加します。

```
# nft add rule inet example_table example_chain tcp dport 22 counter accept
```

2. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

10.10.2. 既存のルールへのカウンターの追加

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- カウンターで新しいルールを追加する手順の詳細は、**Configuring and managing networking** の Creating a rule with the counter を参照してください。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールの代わりに、**counter** パラメーターを使用してカウンターを追加します。以下の例は、前の手順で表示したルールの代わりに、カウンターを追加します。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter accept
```

3. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

10.10.3. 既存のルールに一致するパケットの監視

nftables のトレース機能と、**nft monitor** コマンドを組み合わせることにより、管理者はルールに一致するパケットを表示できます。このルールに一致するパケットを監視するために、ルールのトレースを有効にできます。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールを置き換えてトレース機能を追加しますが、**meta nftrace set 1** パラメーターを使用します。以下の例は、前の手順で表示したルールの代わりに、トレースを有効にします。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
```

3. **nft monitor** コマンドを使用して、トレースを表示します。以下の例は、コマンドの出力をフィルタリングして、**inet example_table example_chain** が含まれるエントリーのみを表示します。

```
# nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr 52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728 tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1 accept (verdict accept)
...
```

警告

nft monitor コマンドは、トレースが有効になっているルールの数と、一致するトラフィックの量に応じて、大量の出力を表示できます。**grep** などのユーティリティーを使用して出力をフィルタリングします。

10.11. NFTABLES ルールセットのバックアップおよび復元

nftables ルールをファイルにバックアップし、後で復元できます。また、管理者はルールが含まれるファイルを使用して、たとえばルールを別のサーバーに転送できます。

10.11.1. ファイルへの nftables ルールセットのバックアップ

nft ユーティリティーを使用して、**nftables** ルールセットをファイルにバックアップできます。

手順

- **nftables** ルールのバックアップを作成するには、次のコマンドを実行します。
  - **nft list ruleset** 形式で生成された形式の場合:

```
# nft list ruleset > file.nft
```

  - JSON 形式の場合は、以下のようになります。

```
# nft -j list ruleset > file.json
```

10.11.2. ファイルからの nftables ルールセットの復元

ファイルから **nftables** ルールセットを復元できます。

手順

- **nftables** ルールを復元するには、以下を行います。
  - 復元するファイルが、**nft list ruleset** が生成した形式であるか、**nft** コマンドを直接含んでいる場合は、以下のコマンドを実行します。

```
# nft -f file.nft
```

  - 復元するファイルが JSON 形式の場合は、次のコマンドを実行します。

```
# nft -j -f file.json
```

10.12. 関連情報

- Using nftables in Red Hat Enterprise Linux 8
- What comes after iptables?Its successor, of course: nftables
- Firewalld: The Future is nftables

第11章 ネットワークサービスのセキュリティー保護

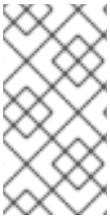
Red Hat Enterprise Linux 8 は、さまざまな種類のネットワークサーバーをサポートしています。RHEL 9 ネットワークサービスを使用すると、システムのセキュリティーが DoS 攻撃 (Denial of Service)、DDoS 攻撃 (Distributed Denial of Service)、スクリプト脆弱性攻撃、バッファーオーバーフロー攻撃など、さまざまな種類の攻撃のリスクにさらされる可能性があります。

攻撃に対するシステムのセキュリティーを強化するには、使用しているアクティブなネットワークサービスを監視することが重要です。たとえば、ネットワークサービスがマシンで実行されている場合に、そのデーモンはネットワークポートでの接続をリッスンするのでセキュリティーが低下する可能性があります。ネットワークに対する攻撃に対する公開を制限するには、未使用のすべてのサービスをオフにする必要があります。

11.1. RPCBIND サービスのセキュリティー保護

rpcbind サービスは、Network Information Service (NIS) や Network File System (NFS) などの Remote Procedure Calls (RPC) サービス用の動的ポート割り当てデーモンです。その認証メカニズムは弱く、制御するサービスに幅広いポート範囲を割り当てる可能性があるため、**rpcbind** をセキュア化することが重要です。

すべてのネットワークへのアクセスを制限し、サーバーのファイアウォールルールを使用して特定の例外を定義することにより、**rpcbind** のセキュリティーを確保できます。



注記

- **NFSv2** および **NFSv3** サーバーでは **rpcbind** サービスが必要です。
- **NFSv4** では、**rpcbind** サービスがネットワークをリッスンする必要がありません。

前提条件

- **rpcbind** パッケージがインストールされている。
- **FirewallD** パッケージがインストールされ、サービスが実行されている。

手順

1. 次に、ファイアウォールルールを追加します。
   - TCP 接続を制限し、**111** ポート経由の **192.168.0.0/24** ホストからのパッケージだけを受け入れます。

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

   - TCP 接続を制限し、**111** ポート経由のローカルホストからのパッケージだけを受け入れます。

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

   - UDP 接続を制限し、**111** ポート経由の **192.168.0.0/24** ホストからのパッケージだけを受け入れます。

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

ファイアウォール設定を永続化するには、ファイアウォールルールを追加するときに **--permanent** オプションを使用します。

2. ファイアウォールをリロードして、新しいルールを適用します。

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

検証手順

- ファイアウォールルールをリストします。

```
# 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
```

関連情報

- **NFSv4-only** サーバーの詳細は、Configuring an NFSv4-only server セクションを参照してください。
- firewalld の使用および設定

11.2. RPC.MOUNTD サービスのセキュリティー保護

rpc.mountd デーモンは、NFS マウントプロトコルのサーバー側を実装します。NFS マウントプロトコルは、NFS バージョン 2 (RFC 1904) および NFS バージョン 3 (RFC 1813) で使用されます。

rpc.mountd サービスは、サーバーにファイアウォールルールを追加することでセキュリティー保護できます。すべてのネットワークへのアクセスを制限し、ファイアウォールルールを使用して特定の例外を定義できます。

前提条件

- **rpc.mountd** パッケージがインストールされている。
- **Firewalld** パッケージがインストールされ、サービスが実行されている。

手順

1. 以下のように、サーバーにファイアウォールルールを追加します。
   - **192.168.0.0/24** ホストからの **mountd** 接続を許可します。

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

   - ローカルホストからの **mountd** 接続を受け入れます。

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

ファイアウォール設定を永続化するには、ファイアウォールルールを追加するときに **--permanent** オプションを使用します。

2. ファイアウォールをリロードして、新しいルールを適用します。

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

検証手順

- ファイアウォールルールをリストします。

```
# 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
```

関連情報

- firewalld の使用および設定

11.3. NFS サービスの保護

Kerberos を使用してすべてのファイルシステム操作を認証および暗号化して、ネットワークファイルシステムバージョン 4 (NFSv4) のセキュリティーを保護できます。ネットワークアドレス変換 (NAT) またはファイアウォールで NFSv4 を使用する場合に、**/etc/default/nfs** ファイルを変更することで委譲をオフにできます。委譲は、サーバーがファイルの管理をクライアントに委譲する手法です。対照的に、NFSv2 および NFSv3 ではファイルのロックとマウントに Kerberos は使用されません。

NFS サービスは、すべてのバージョンの NFS で TCP を使用してトラフィックを送信します。このサービスは、**RPCSEC_GSS** カーネルモジュールの一部として Kerberos ユーザーおよびグループ認証をサポートします。

NFS を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムのように操作できるようになります。集約サーバーのリソースを統合して、ファイルシステムを共有するときに **/etc/nfsmount.conf** ファイルの NFS マウントオプションをさらにカスタマイズできます。

11.3.1. NFS サーバーのセキュリティーを保護するエクスポートオプション

NFS サーバーは、**/etc/exports** ファイル内のどのファイルシステムにどのファイルシステムをエクスポートするかなど、ディレクトリーとホストのリスト構造を決定します。

警告

エクスポートファイルの構文に余分なスペースがあると、設定が大幅に変更される可能性があります。

以下の例では、**/tmp/nfs/** ディレクトリーは **bob.example.com** ホストと共有され、読み取りおよび書き込みのパーミッションを持ちます。

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

以下の例は上記と同じになりますが、同じディレクトリーを読み取り専用パーミッションで **bob.example.com** ホストに共有し、ホスト名の後の 1 つのスペース文字が原因で読み取りと書き込み権限で **すべてのユーザー** に共有します。

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

showmount -e <hostname> コマンドを入力すると、システム上の共有ディレクトリーを確認できます。

/etc/exports ファイルで次のエクスポートオプションを使用します。

警告

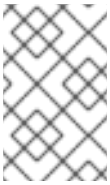
ファイルシステムのサブディレクトリーのエクスポートは安全ではないため、ファイルシステム全体をエクスポートします。攻撃者は、部分的にエクスポートされたファイルシステムで、エクスポートされていない部分にアクセスする可能性があります。

ro

ro オプションを使用して、NFS ボリュームを読み取り専用としてエクスポートします。

rw

rw オプションを使用して、NFS ボリュームに対する読み取りおよび書き込み要求の両方を許可します。書き込みアクセスが許可されると攻撃のリスクが高まるため、このオプションは注意して使用してください。



注記

シナリオとして **rw** オプションを使用してディレクトリーをマウントする必要がある場合は、起こりうるリスクを軽減するために、すべてのユーザーがディレクトリーを書き込み可能にしないようにしてください。

root_squash

root_squash オプションを使用して、**uid/gid** 0 からの要求を匿名の **uid/gid** にマッピングします。これは、**bin** ユーザーや **staff** グループなど、同様に機密である可能性の高い他の **uid** または **gid** には適用されません。

no_root_squash

root squashing をオフにするには、**no_root_squash** オプションを使用します。デフォルトでは、NFS 共有は **root** ユーザーを、非特権ユーザーである **nobody** ユーザーに変更します。これにより、**root** が作成したすべてのファイルの所有者が **nobody** に変更され、**setuid** ビットが設定されたプログラムのアップロードができなくなります。**no_root_squash** オプションを使用すると、リモートの root ユーザーは共有ファイルシステムの任意のファイルを変更し、他のユーザーに対してアプリケーションが Trojans に感染した状態のままにします。

secure

secure オプションを使用して、エクスポートを予約ポートに制限します。デフォルトでは、サーバーは予約済みポートからのクライアント通信のみを許可します。ただし、多くのネットワークで、クライアント上で **root** ユーザーになるのは簡単です。そのため、サーバーで予約されたポートからの通信が特権であると仮定することは安全ではありません。そのため、予約ポートの制限は効果が限定的です。Kerberos、ファイアウォール、および特定クライアントへのエクスポートを制限することに依存すると良いでしょう。

また、NFS サーバーをエクスポートする際に、以下のベストプラクティスを考慮してください。

- 一部のアプリケーションでは、パスワードをプレーンテキストまたは弱い暗号化形式で保存するため、ホームディレクトリーをエクスポートすることはリスクがあります。アプリケーションコードを確認して改善することで、リスクを軽減できます。
- 一部のユーザーは SSH キーにパスワードを設定していないため、この場合もホームディレクトリーによるリスクが発生します。パスワードの使用を強制するか、Kerberos を使用することで、これらのリスクを軽減できます。
- NFS エクスポートを必要なクライアントのみに制限します。NFS サーバーで **showmount -e** コマンドを使用して、サーバーのエクスポート内容を確認します。特に必要のないものはエクスポートしないでください。
- 攻撃のリスクを減らすために、不要なユーザーがサーバーにログインできないようにしてください。サーバーにアクセスできるユーザーを定期的に確認してください。

関連情報

- Secure NFS with Kerberos when using Red Hat Identity Management
- **exports(5)** および **nfs(5)** の man ページ

11.3.2. NFS クライアントのセキュリティーを保護するマウントオプション

mount コマンドに次のオプションを渡すと、NFS ベースのクライアントのセキュリティーを強化できます。

nosuid

nosuid オプションを使用して **set-user-identifier** または **set-group-identifier** ビットを無効にします。これにより、リモートユーザーが **setuid** プログラムを実行してより高い特権を取得するのを防ぎ、**setuid** オプションの反対となるこのオプションを使用できます。

noexec

noexec オプションを使用して、クライアント上の実行可能なファイルをすべて無効にします。これを使用して、ユーザーが共有ファイルシステムに配置されたファイルを誤って実行するのを防ぎます。

nodev

nodev オプションを使用して、クライアントがデバイスファイルをハードウェアデバイスとして処理するのを防ぎます。

resvport

resvport オプションを使用して、通信を予約済みポートに制限し、特権送信元ポートを使用してサーバーと通信できます。予約済みポートは、**root** ユーザーなどの特権ユーザーおよびプロセス用に予約されています。

秒

NFS サーバーの **sec** オプションを使用して、マウントポイント上のファイルにアクセスするための RPCGSS セキュリティーフレーバーを選択します。有効なセキュリティーフレーバーは、**none**、**sys**、**krb5**、**krb5i**、および **krb5p** です。

重要

krb5-libs パッケージが提供する MIT Kerberos ライブラリーは、新しいデプロイメントで Data Encryption Standard (DES) アルゴリズムに対応しなくなりました。DES は、セキュリティーと互換性の理由から、Kerberos ライブラリーでは非推奨であり、デフォルトで無効になっています。互換性の理由でご使用の環境で DES が必要な場合を除き、DES の代わりに新しくより安全なアルゴリズムを使用してください。

関連情報

- 一般的な NFS マウントオプション

11.3.3. ファイアウォールでの NFS のセキュリティー保護

NFS サーバーでファイアウォールを保護するには、必要なポートのみを開いてください。他のサービスには NFS 接続ポート番号を使用しないでください。

前提条件

- **nfs-utils** パッケージがインストールされている。
- **Firewalld** パッケージがインストールされ、実行されている。

手順

- NFSv4 では、ファイアウォールは TCP ポート **2049** を開く必要があります。
- NFSv3 では、**2049** で 4 つのポートを追加で開きます。

1. **rpcbind** サービスは NFS ポートを動的に割り当て、ファイアウォールルールの作成時に問題が発生する可能性があります。このプロセスを簡素化するには、**/etc/nfs.conf** ファイルを使用して、使用するポートを指定します。

   a. **mountd** セクションの **mountd** (**rpc.mountd**) の TCP および UDP ポートを **port= <value>** 形式で設定します。

   b. **statd** セクションの **statd** (**rpc.statd**) の TCP および UDP ポートを **port= <value>** 形式で設定します。

2. **/etc/nfs.conf** ファイルで NFS ロックマネージャー (**nlockmgr**) の TCP および UDP ポートを設定します。

a. **lockd** セクションの **nlockmgr** (**rpc.statd**) の TCP ポートを **port=value** 形式で設定します。または、**/etc/modprobe.d/lockd.conf** ファイルの **nlm_tcpport** オプションを使用することもできます。

b. **lockd** セクションの **nlockmgr** (**rpc.statd**) の UDP ポートを **udp-port=value** 形式で設定します。または、**/etc/modprobe.d/lockd.conf** ファイルの **nlm_udpport** オプションを使用することもできます。

検証手順

- NFS サーバー上のアクティブなポートと RPC プログラムをリスト表示します。

```
$ rpcinfo -p
```

関連情報

- Secure NFS with Kerberos when using Red Hat Identity Management
- **exports(5)** および **nfs(5)** の man ページ

11.4. FTP サービスのセキュリティー保護

ファイル転送プロトコル (FTP) を使用して、ネットワーク経由でファイルを転送できます。ユーザー認証を含むサーバーとの FTP トランザクションがすべて暗号化されているわけではないため、サーバーが安全に設定されていることを確認する必要があります。

RHEL 8 は、2 つの FTP サーバーを提供します。

- Red Hat Content Accelerator (tux): FTP 機能を持つカーネルスペースの Web サーバー。
- Very Secure FTP Daemon (vsftpd): スタンドアロンの、セキュリティー指向の FTP サービスの実装。

vsftpd FTP サービスをセットアップするためのセキュリティーガイドラインを以下に示します。

11.4.1. FTP グリーティングバナーのセキュリティー保護

ユーザーが FTP サービスに接続すると、FTP はグリーティングバナーを表示します。このバナーにはデフォルトで、バージョン情報が含まれており、攻撃者がシステムの弱点を特定するのに役立つ場合があります。デフォルトのバナーを変更することで、攻撃者がこの情報にアクセスできないようにします。

/etc/banners/ftp.msg ファイルを編集して、単一行のメッセージを直接含めるか、複数行のメッセージを含めることができる別のファイルを参照して、カスタムバナーを定義できます。

手順

- 1 行のメッセージを定義するには、次のオプションを **/etc/vsftpd/vsftpd.conf** ファイルに追加します。

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

- 別のファイルでメッセージを定義するには以下を実行します。

- バナーメッセージを含む **.msg** ファイルを作成します。(例: **/etc/vendors/ftp .msg**)

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

  複数のバナーの管理を簡素化するには、すべてのバナーを **/etc/vendors/** ディレクトリーに配置します。

- バナーファイルへのパスを **/etc/vsftpd/vsftpd.conf** ファイルの **banner_file** オプションに追加します。

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

検証

- 変更されたバナーを表示します。

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

11.4.2. FTP での匿名アクセスとアップロードの防止

デフォルトでは、**vsftpd** パッケージをインストールすると、**/var/ftp/** ディレクトリーと、ディレクトリーに対する読み取り専用権限を持つ匿名ユーザー用のディレクトリーツリーが作成されます。匿名ユーザーはデータにアクセスできるため、これらのディレクトリーに機密データを保存しないでください。

システムのセキュリティーを強化するために、匿名ユーザーが特定のディレクトリーにファイルをアップロードできるが、データは読み取れないように、FTP サーバーを設定できます。次の手順では、匿名ユーザーが **root** ユーザー所有のディレクトリーにファイルをアップロードできるが変更できないようにする必要があります。

手順

- **/var/ftp/pub/** ディレクトリーに書き込み専用ディレクトリーを作成します。

  ```
  # 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
  ```

- **/etc/vsftpd/vsftpd.conf** ファイルに以下の行を追加します。

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

- オプション: システムで SELinux が有効で Enforcing に設定されている場合には、SELinux ブール属性 **allow_ftpd_anon_write** および **allow_ftpd_full_access** を有効にします。

警告

匿名ユーザーによるディレクトリーの読み取りと書き込みを許可すると、盗まれたソフトウェアのリポジトリーになってしまう可能性があります。

11.4.3. FTP のユーザーアカウントのセキュリティー保護

FTP は、認証のために安全でないネットワークを介して暗号化されていないユーザー名とパスワードを送信します。システムユーザーが自分のユーザーアカウントからサーバーにアクセスできないようにして、FTP のセキュリティーを向上させることができます。

以下の手順のうち、お使いの設定に該当するものをできるだけ多く実行してください。

手順

- **/etc/vsftpd/vsftpd.conf** ファイルに次の行を追加して、**vsftpd** サーバーのすべてのユーザーアカウントを無効にします。

```
local_enable=NO
```

- **/etc/pam.d/vsftpd** PAM 設定ファイルにユーザー名を追加して、特定のアカウントまたは特定のアカウントグループ (**root** ユーザーや **sudo** 権限を持つユーザーなど) の FTP アクセスを無効にします。
- **/etc/vsftpd/ftpusers** ファイルにユーザー名を追加して、ユーザーアカウントを無効にします。

11.4.4. 関連情報

- **ftpd_selinux(8)** の man ページ

11.5. HTTP サーバーのセキュリティー保護

11.5.1. httpd.conf のセキュリティー強化

/etc/httpd/conf/httpd.conf ファイルでセキュリティーオプションを設定して、Apache HTTP のセキュリティーを強化できます。

システムで実行されているすべてのスクリプトが正しく機能することを常に確認してから、本番環境に移行してください。

root ユーザーのみが、スクリプトまたは Common Gateway Interface (CGI) を含むディレクトリーへの書き込み権限を持っていることを確認してください。ディレクトリーの所有権を書き込み権限を持つ **root** ユーザーに変更するには、次のコマンドを入力します。

```
# chown root directory-name
# chmod 755 directory-name
```

/etc/httpd/conf/httpd.conf ファイルで、次のオプションを設定できます。

FollowSymLinks

このディレクティブはデフォルトで有効になっており、ディレクトリー内のシンボリックリンクをたどります。

Indexes

このディレクティブはデフォルトで有効になっています。訪問者がサーバー上のファイルを閲覧できないようにするには、このディレクティブを削除してください。

UserDir

このディレクティブは、システム上にユーザーアカウントが存在することを確認できるため、デフォルトでは無効になっています。**/root/** 以外のすべてのユーザーディレクトリーのユーザーディレクトリーブラウジングをアクティブにするには、**UserDir enabled** と **UserDir disabled** の root ディレクティブを使用します。無効化されたアカウントのリストにユーザーを追加するには、**UserDir disabled** 行にスペースで区切られたユーザーのリストを追加します。

ServerTokens

このディレクティブは、クライアントに送り返されるサーバー応答ヘッダーフィールドを制御します。以下のパラメーターを使用するとログの出力をカスタマイズできます。

ServerTokens Full

以下のように、Web サーバーのバージョン番号、サーバーのオペレーティングシステムの詳細、インストールされている Apache モジュールなど、利用可能なすべての情報を提供します。

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

ServerTokens Full-Release

以下のように、利用可能なすべての情報をリリースバージョンとともに提供します。

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

ServerTokens Prod / ServerTokens ProductOnly

以下のように、Web サーバー名を提供します。

```
Apache
```

ServerTokens Major

以下のように、Web サーバーのメジャーリリースバージョンを提供します。

```
Apache/2
```

ServerTokens Minor

以下のように、Web サーバーのマイナーリリースバージョンを提供します。

```
Apache/2.4
```

ServerTokens Min / ServerTokens Minimal

以下のように、Web サーバーの最小リリースバージョンを提供します。

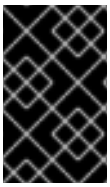
```
Apache/2.4.37
```

ServerTokens OS

以下のように、Web サーバーのリリースバージョンとオペレーティングシステムを提供します。

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

ServerTokens Prod オプションを使用して、攻撃者がシステムに関する貴重な情報を入手するリスクを軽減します。



重要

IncludesNoExec ディレクティブを削除しないでください。デフォルトでは、Server-Side Includes (SSI) モジュールは、コマンドを実行できません。これを変更すると、攻撃者がシステムにコマンドを入力できるようになる可能性があります。

httpd モジュールの削除

httpd モジュールを削除して、HTTP サーバーの機能を制限できます。これを行うには、**/etc/httpd/conf.modules.d/** または **/etc/httpd/conf.d/** ディレクトリーの設定ファイルを編集します。たとえば、プロキシーモジュールを削除するためには、以下のコマンドを実行します。

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

関連情報

- Apache HTTP サーバー
- Apache HTTP サーバーの SELinux ポリシーのカスタマイズ

11.5.2. Nginx サーバー設定のセキュリティー保護

Nginx は、高性能の HTTP およびプロキシーサーバーです。次の設定オプションを使用して、Nginx 設定を強化できます。

手順

- バージョン文字列を無効にするには、**server_tokens** 設定オプションを変更します。

  ```
  server_tokens off;
  ```

  このオプションは、サーバーのバージョン番号などの追加の情報表示を停止します。以下のようにこの設定では、Nginx によって処理されるすべての要求のサーバー名のみが表示されます。

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

- 特定の **/etc/nginx/** conf ファイルに、特定の既知の Web アプリケーションの脆弱性を軽減するセキュリティーヘッダーを追加します。
  - たとえば、**X-Frame-Options** ヘッダーオプションは、Nginx が提供するコンテンツのフレーム化がされないように、ドメイン外のページを拒否して、クリックジャッキング攻撃を軽減します。

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

- たとえば、**x-content-type** ヘッダーは、特定の古いブラウザーでの MIME タイプのスニッフィングを防ぎます。

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

- また、**X-XSS-Protection** ヘッダーは、クロスサイトスクリプティング (XSS) フィルタリングを有効にし、Nginx での応答に含まれる可能性がある、悪意のあるコンテンツをブラウザーがレンダリングしないようにします。

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

- たとえば、一般に公開されるサービスを制限し、訪問者からのサービスと受け入れを制限できます。

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

スニペットは、**GET** と **HEAD** を除くすべてのメソッドへのアクセスを制限します。

- 以下のように、HTTP メソッドを無効にできます。

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

- Nginx Web サーバーによって提供されるデータを保護するように SSL を設定できます。これは、HTTPS 経由でのみ提供することを検討してください。さらに、Mozilla SSL Configuration Generator を使用して、Nginx サーバーで SSL を有効にするための安全な設定プロファイルを生成できます。生成された設定により、既知の脆弱なプロトコル (SSLv2 や SSLv3 など)、暗号、ハッシュアルゴリズム (3DES や MD5 など) が確実に無効化されます。また、SSL サーバーテストを使用して、設定した内容が最新のセキュリティー要件を満たしていることを確認できます。

関連情報

- Mozilla SSL Configuration Generator
- SSL Server Test

11.6. 認証されたローカルユーザーへのアクセスを制限することによる POSTGRESQL のセキュリティー保護

PostgreSQL は、オブジェクトリレーショナルデータベース管理システム (DBMS) です。Red Hat Enterprise Linux では、PostgreSQL は **postgresql-server** パッケージによって提供されます。

クライアント認証を設定して、攻撃のリスクを減らすことができます。データベースクラスターのデータディレクトリーに保存されている **pg_hba.conf** 設定ファイルは、クライアント認証を制御します。手順に従って、ホストベースの認証用に PostgreSQL を設定します。

手順

1. PostgreSQL をインストールします。

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

2. 次のいずれかのオプションを使用して、データベースストレージ領域を初期化します。

   a. **initdb** ユーティリティーの使用:

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

   -D オプションを指定した **initdb** コマンドを実行すると、指定したディレクトリーがまだ存在しない場合は作成します (例: **/home/postgresql/db1/**)。このディレクトリーには、データベースに保存されているすべてのデータと、クライアント認証設定ファイルが含まれています。

   b. **postgresql-setup** スクリプトの使用:

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

   デフォルトでは、スクリプトは **/var/lib/pgsql/data/** ディレクトリーを使用します。このスクリプトは、基本的なデータベースクラスター管理でシステム管理者を支援します。

3. 認証されたローカルユーザーが自分のユーザー名でデータベースにアクセスできるようにするには、**pg_hba.conf** ファイルの以下の行を変更します。

```
local   all             all                                     trust
```

   これは、データベースユーザーを作成し、ローカルユーザーを作成しないレイヤー型アプリケーションを使用する場合に、問題となることがあります。システム上のすべてのユーザー名を明示的に制御しない場合は、**pg_hba.conf** ファイルから **local** の行を削除してください。

4. データベースを再起動して、変更を適用します。

```
# systemctl restart postgresql
```

   前のコマンドはデータベースを更新し、設定ファイルの構文も検証します。

11.7. MEMCACHED サービスのセキュリティー保護

Memcached は、オープンソースの高性能分散メモリーオブジェクトキャッシングシステムです。データベースの負荷を軽減して、動的 Web アプリケーションのパフォーマンスを向上させることができます。

Memcached は、データベース呼び出し、API 呼び出し、またはページレンダリングの結果から、文字列やオブジェクトなどの任意のデータの小さなチャンクを格納するメモリー内のキーと値のストアです。Memcached を使用すると、十分に活用されていない領域から、より多くのメモリーを必要とするアプリケーションにメモリーを割り当てることができます。

2018 年に、パブリックインターネットに公開されている Memcached サーバーを悪用することによる DDoS 増幅攻撃の脆弱性が発見されました。これらの攻撃は、トランスポートに UDP プロトコルを使用する Memcached 通信を利用します。この攻撃は増幅率が高いため、効果的です。数百バイトのサイズの要求は、数メガバイトまたは数百メガバイトのサイズの応答を生成することができます。

ほとんどの場合、**memcached** サービスはパブリックインターネットに公開する必要はありません。このような弱点には、リモートの攻撃者が memcached に保存されている情報を漏洩または変更できるなど、独自のセキュリティー問題があります。

このセクションに従って、DDoS 攻撃の可能性に対して Memcached サービスを使用してシステムを強化します。

11.7.1. DDoS に対する Memcached の強化

セキュリティーリスクを軽減するために、以下の手順のうち、お使いの設定に該当するものをできるだけ多く実行してください。

手順

- LAN にファイアウォールを設定してください。Memcached サーバーにローカルネットワークだけでアクセスできるようにする必要がある場合は、**memcached** サービスで使用されるポートに外部トラフィックをルーティングしないでください。たとえば、許可されたポートのリストからデフォルトのポート **11211** を削除します。

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

- アプリケーションと同じマシンで単一の memcached サーバーを使用する場合、ローカルホストトラフィックのみをリッスンするように **memcached** を設定します。**/etc/sysconfig/memcached** ファイルの **OPTIONS** 値を変更します。

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

- Simple Authentication and Security Layer (SASL) 認証を有効にします。

  1. **/etc/sasl2/memcached.conf** ファイルで、以下のように修正または追加します。

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

  2. SASL データベースにアカウントを追加します。

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

  3. **memcached** のユーザーとグループがデータベースにアクセスできることを確認します。

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

  4. **/etc/sysconfig/memcached** ファイルの **OPTIONS** パラメーターに **-S** 値を追加して、Memcached で SASL サポートを有効にします。

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

  5. Memcached サーバーを再起動して、変更を適用します。

  ```
  # systemctl restart memcached
  ```

  6. SASL データベースで作成したユーザー名とパスワードを、お使いのアプリケーションの Memcached クライアント設定に追加します。

- memcached クライアントとサーバー間の通信を TLS で暗号化します。

  1. **/etc/sysconfig/memcached** ファイルの **OPTIONS** パラメーターに **-Z** 値を追加して、TLS を使用した Memcached クライアントとサーバー間の暗号化通信を有効にします。

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

  2. **-o ssl_chain_cert** オプションを使用して、証明書チェーンファイルパスを PEM 形式で追加します。

  3. **-o ssl_key** オプションを使用して、秘密鍵ファイルのパスを追加します。