



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

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

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書は、管理者が、様々な攻撃からネットワーク、接続されているマシン、およびネットワーク通信のセキュリティーを保護する方法を説明します。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)	4
第1章 2 台のシステム間で OPENSSSH を使用した安全な通信の使用	5
1.1. SSH と OPENSSSH	5
1.2. OPENSSSH サーバーの設定および起動	6
1.3. SSH 認証にパスワードではなく鍵ペアを使用	7
1.4. スマートカードに保存された SSH 鍵の使用	9
1.5. OPENSSSH のセキュリティーの強化	11
1.6. SSH ジャンプホストを使用してリモートサーバーに接続	13
1.7. 関連情報	14
第2章 TLS の計画および実施	16
2.1. SSL プロトコルおよび TLS プロトコル	16
2.2. RHEL 8 における TLS のセキュリティー上の検討事項	16
2.3. アプリケーションで TLS 設定の強化	18
第3章 IPSEC を使用した VPN の設定	21
3.1. IPSEC VPN 実装としての LIBRESWAN	21
3.2. LIBRESWAN のインストール	22
3.3. ホスト間の VPN の作成	22
3.4. サイト間 VPN の設定	23
3.5. リモートアクセスの VPN の設定	24
3.6. メッシュ VPN の設定	25
3.7. LIBRESWAN で使用される認証方法	27
3.8. FIPS 準拠の IPSEC VPN のデプロイメント	28
3.9. パスワードによる IPSEC NSS データベースの保護	31
3.10. 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	32
3.11. 関連情報	33
第4章 MACSEC の設定	34
4.1. MACSEC の概要	34
4.2. NMCLI ツールで MACSEC の使用	34
4.3. WPA_SUPPLICANT で MACSEC の使用	34
4.4. 関連情報	35
第5章 FIREWALLD の使用および設定	36
5.1. FIREWALLD、NFTABLES、または IPTABLES を使用する場合	36
5.2. FIREWALLD の使用	36
5.3. FIREWALL-CONFIG GUI 設定ツールのインストール	38
5.4. FIREWALLD の現在の状況および設定の表示	39
5.5. FIREWALLD の起動	41
5.6. FIREWALLD の停止	41
5.7. ランタイムおよび永続化の設定	41
5.8. 永続的な FIREWALLD 設定の確認	42
5.9. FIREWALLD でネットワークトラフィックの制御	43
5.10. ファイアウォールゾーンでの作業	47
5.11. 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	51
5.12. IP アドレスのマスカレードの設定	53
5.13. ポート転送	54
5.14. ICMP リクエストの管理	56
5.15. FIREWALLD を使用した IP セットの設定および制御	58
5.16. リッチルールの優先度設定	60

5.17. ファイアウォールロックダウンの設定	61
5.18. 拒否されたパケットのログ	64
5.19. 関連情報	65
第6章 NFTABLES の使用	67
6.1. NFTABLES の概要	67
6.2. FIREWALLD、NFTABLES、または IPTABLES を使用する場合	67
6.3. IPTABLES のルールを NFTABLES ルールに変換	68
6.4. NFTABLES スクリプトの作成および実行	68
6.5. NFTABLES ルールセットの表示	73
6.6. NFTABLES テーブルの作成	73
6.7. NFTABLES チェーンの作成	74
6.8. NFTABLES チェーンへのルールの追加	75
6.9. NFTABLES チェーンへのルールの追加	76
6.10. NFTABLES を使用した NAT の設定	76
6.11. NFTABLES コマンドを使用したセットの使用	79
6.12. NFTABLES コマンドにおける決定マップの使用	81
6.13. NFTABLES を使用したポート転送の設定	85
6.14. NFTABLES を使用した接続数の制限	86
6.15. 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	87
6.16. NFTABLES ルールのデバッグ	87
6.17. NFTABLES ルールセットのバックアップおよび復元	90
6.18. 関連情報	91

RED HAT ドキュメントへのフィードバック (英語のみ)

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
 1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上端に **Feedback** ボタンがあることを確認してください。
 2. マウスカーソルで、コメントを追加する部分を強調表示します。
 3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
 4. 表示される手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
 1. [Bugzilla](#) の Web サイトにアクセスします。
 2. Component で **Documentation** を選択します。
 3. **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記入してください。
 4. **Submit Bug** をクリックします。

第1章 2 台のシステム間で OPENSSSH 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 OPENSSSH

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。

OpenSSH は、多数の 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。Red Hat Enterprise Linux 8 の **OpenSSH** スイートは、SSH バージョン 2 のみを使用します。バージョン 1 で既知の不正使用の影響を受けない、強化された鍵交換アルゴリズムを備えています。

OpenSSH は、RHEL コア暗号化サブシステムのいずれかで、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。ポリシーを調整するには、管理者が **update-crypto-policies** コマンドを使用して設定を厳格または緩和するか、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、設定ファイルのセットを 2 つ使用します。クライアントプログラム (つまり

ssh、**scp**、および **sftp** の設定ファイルと、サーバー (**sshd** デーモン) の設定ファイルです。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一覧は、man ページの **sshd (8)** の **FILES** セクションを参照してください。

関連情報

- **man -k ssh** コマンドで一覧表示される **ssh** トピックの man ページ
- [システム全体の暗号化ポリシーの使用](#)

1.2. OPENSSSH サーバーの設定および起動

お使いの環境と **OpenSSH** サーバーを起動するのに必要な基本設定には、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.
```

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、man ページの **pam_motd** を参照してください。

5. **systemd** 設定を再読み込みし、変更を適用します。

```
# systemctl daemon-reload
```

検証手順

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/ssh.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/ssh.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/UzITtku8GtXSz0S1++IPegSy31v3L/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

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

1.3. SSH 認証にパスワードではなく鍵ペアを使用

システムのセキュリティーをさらに強化するには、SSH 鍵のペアを生成し、パスワード認証を無効にすることで鍵ベース認証を強制します。

1.3.1. 鍵ベースの認証用の OpenSSH サーバーの設定

以下の手順に従って、鍵ベースの認証を強制するように OpenSSH サーバーを設定します。

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。

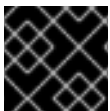
```
# systemctl reload sshd
```

関連情報

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

1.3.2. 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/joeseq/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joeseq/.ssh/id_ecdsa.
Your public key has been saved in /home/joeseq/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joeseq@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++      |
|.. o .oo .     |
```

```

|.. 0. 0 |
|...0.+... |
|0.00.0 +S. |
|.=.+ .0 |
|E.*+ . . . |
|.=.+ +.. 0 |
| . 00*+0. |
+----[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。

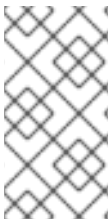
```

$ 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
...
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)。別の公開鍵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。

検証手順

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

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

1.4. スマートカードに保存された SSH 鍵の使用

Red Hat Enterprise Linux 8 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

- PKCS #11 の URI を含む OpenSC PKCS #11 モジュールが提供する鍵の一覧を表示し、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
```

- 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
```

- 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] $
```

- ~/.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] $
```

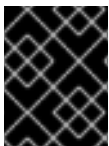
関連情報

- [Fedora 28: Better smart card support in OpenSSH](#)
- man ページの [p11-kit\(8\)](#)
- man ページの [ssh\(1\)](#)
- man ページの [ssh-keygen\(1\)](#)
- man ページの [opensc.conf\(5\)](#)
- man ページの [pcscd\(8\)](#)

1.5. OPENSSSH 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 8 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
```

- 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
```

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、自動化した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 --runtime-to-permanent
```

このコマンドで、**port_number** を、**Port** ディレクティブで指定された新しいポート番号に置き換えます。

root ログインなし

- 特定のユースケースで 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** に設定することを検討してください。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は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。

ホワイトリストは、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、ホワイトリスト (`Allow` で始まるディレクティブ) の使用は、ブラックリスト (`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
```

- **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、`KexAlgorithms` 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。このタスクには、暗号化オプションの設定に関する深い専門知識が必要になることに注意してください。
- 詳細は、『[RHEL 8 セキュリティーの強化](#)』の「[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

関連情報

- `man` ページの `sshd_config(5)`、`ssh-keygen(1)`、`crypto-policies(7)`、および `update-crypto-policies(8)`

1.6. 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

- ジャンプホストが、システムからの SSH 接続を受け入れる。
- 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. `~/.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
```

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 `~/.ssh/config` 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. ジャンプサーバーを使用して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を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@jump3.ex
ample.com:75 joesec@remote1.example.com:220
```

関連情報

- `man` ページの `ssh_config(5)` および `ssh(1)`

1.7. 関連情報

Red Hat Enterprise Linux で **OpenSSH** サーバーおよびクライアントを設定し、接続する方法は、以下の資料を参照してください。

インストールされているドキュメント

- man ページの **sshd(8)** では、利用可能なコマンドラインオプションと、対応している設定ファイルおよびディレクトリーがすべて説明されています。
- man ページの **ssh(1)** には、利用可能なコマンドラインオプションと対応している設定ファイルおよびディレクトリーの一覧が記載されています。
- man ページの **scp(1)** には、**scp** ユーティリティーとその使用方法に関する詳細が記載されています。
- man ページの **sftp(1)** には、**sftp** ユーティリティーとその使用方法に関する詳細が記載されています。
- man ページの **ssh-keygen(1)** には、**ssh-keygen** ユーティリティーを使用して **ssh** が使用する認証鍵を生成、管理、変換する方法を説明します。
- man ページの **ssh-copy-id(1)** では、**ssh-copy-id** スクリプトの使用方法が説明されています。
- man ページの **ssh_config(5)** では、利用可能な SSH クライアント設定オプションが説明されています。
- man ページの **sshd_config(5)** は、利用可能な SSH デーモン設定オプションの詳細な説明があります。
- man ページの **update-crypto-policies(8)** は、システム全体の暗号化ポリシーの管理に関するガイドを提供します。
- man ページの **crypto-policies(7)** は、システム全体の暗号化ポリシーレベルの概要を提供します。

オンラインドキュメント

- [OpenSSH Home Page](#) - その他のドキュメントやFAQ、メーリングリストへのリンク、バグレポートなどの役立つリソースが掲載されています。
- [「非標準設定でアプリケーションとサービスのSELinuxの設定」](#) - Enforcing モードのSELinuxを使用した非標準設定のOpenSSHに類似の手順を適用できます。
- [「firewalldを使用したネットワークトラフィックの制御」](#) - SSHポートの変更後に **firewalld** 設定の更新に関するガイダンスを提供します。

第2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。

2.1. 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。 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。 LEGACY ポリシーでのみ有効です。
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](#)

2.2. 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。

RHEL システム全体の暗号化ポリシーに従わない場合は、カスタム設定上好ましいプロトコル、暗号スイート、および鍵の長さについて、以下の推奨事項を参照してください。

2.2.1. 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 が TLS バージョン 1.3 に対応していても、RHEL 8 コンポーネントが、このプロトコルのすべての機能に対応しているわけではないことに注意してください。たとえば、現時点では、Apache または Nginx の Web サーバーは、接続レイテンシーを短縮する 0-RTT (Zero Round Trip Time) 機能に完全に対応していません。

2.2.2. 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。

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。

2.2.3. 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](#).
- man ページの `update-crypto-policies(8)`

2.3. アプリケーションで TLS 設定の強化

Red Hat Enterprise Linux 8 では、「システム全体の暗号化ポリシー」では、暗号化ライブラリーを使用したアプリケーションが、安全でないことが知られているプロトコル、暗号、またはアルゴリズムを許可しないようにする便利な方法が紹介されています。

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強制的に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での設定順に行われるように設定してください。

2.3.1. Apache HTTP サーバー の設定

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。Red Hat Enterprise Linux 8 では、`mod_ssl` 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。各種設定の例は `/usr/share/httpd/manual/ssl/ssl_howto.html` で確認できます。

`/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
```

2.3.2. Nginx HTTP およびプロキシサーバーの設定

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
    ....
}
```

2.3.3. Dovecot メールサーバーの設定

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](https://www.dovecot.org/docs/10-ssl.conf) 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

TLS 設定と関連トピックの詳細は、以下に挙げるリソースを参照してください。

- man ページの **config(5)** で、`/etc/ssl/openssl.conf` 設定ファイルの形式が説明されています。
- man ページの **ciphers(1)** には、利用可能な **OpenSSL** キーワードおよび暗号文字列の一覧が記載されています。
- [Recommendations for Secure Use of Transport Layer Security \(TLS\) and Datagram Transport Layer Security \(DTLS\)](#)
- [「Mozilla SSL Configuration Generator」](#) は、既知の脆弱なプロトコル、暗号、およびハッシュアルゴリズムを無効にする安全な設定で、**Apache** または **Nginx** に対して設定ファイルを作成するのに役に立ちます。
- [SSL サーバーテスト](#) は、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します。

第3章 IPSEC を使用した VPN の設定

Red Hat Enterprise Linux 8 では、**Libreswan** アプリケーションで対応している **IPsec** プロトコルを使用して、仮想プライベートネットワーク (VPN) を設定できます。

3.1. IPSEC VPN 実装としての LIBRESWAN

Red Hat Enterprise Linux 8 では、VPN (Virtual Private Network) は、**Libreswan** アプリケーションで対応している **IPsec** プロトコルを使用して設定できます。**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** と相互変更できます。

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。レベル 2 **L2TP** (Level 2 Tunneling Protocol) も使用する IPsec VPN のバリエーションは、通常は L2TP/IPsec VPN と呼ばれます。これには、Optional チャンネルの **xl2tpd** アプリケーションが必要です。

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 つのプロトコルで構成されます。

- 暗号セキュリティペイロード (**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上の理由からは強く推奨されません。**Libreswan** では、**netlink** を使用する Linux カーネルで相互作用が行われます。Linux カーネルでパケットの暗号化と復号が行われます。

Libreswan は、ネットワークセキュリティサービス (**NSS**) 暗号化ライブラリーを使用します。**Libreswan** および **NSS** はともに、**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、Red Hat Enterprise Linux 8 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。

Red Hat Enterprise Linux 8 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「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。

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。

3.2. LIBRESWAN のインストール

この手順では、**Libreswan** IPsec/IKE VPN 実装をインストールおよび起動を行う手順を説明します。

前提条件

- **AppStream** リポジトリ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
```

3.3. ホスト間の VPN の作成

Libreswan が、左 および右 と呼ばれる 2 台のホスト間で、ホスト間の **IPsec** VPN を作成するように設定するには、両方のホストで次のコマンドを実行します。

手順

1. 各ホストで RSA 鍵ペアを生成します。

```
# ipsec newhostkey --output /etc/ipsec.d/hostkey.secrets
```

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+hOafUZDICQmXFrije/oZm [...] W2n417C/4urYHQkCvulQ==
  rightid=@east
  right=192.1.2.45
  rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
  authby=rsasig
```

4. **ipsec** を起動します。

```
# ipsec setup start
```

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
```

3.4. 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+hOafUZDICQmXFrije/oZm [...] W2n417C/4urYHQkCvulQ==
  rightid=@east
  right=192.1.2.45
  rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
  authby=rsasig
```

3.5. リモートアクセスの VPN の設定

ロードウォリアーとは、外出先などで、ノート PC など、動的に IP アドレスが割り当てられたモバイルクライアントを使用するユーザーを指します。モバイルクライアントは、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
  leftauthserver=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/deny client
# pam-authorize=yes
# load connection, don't initiate
auto=add
# kill vanished roadwarriors
dpddelay=1m
dpdtimeout=5m
dpdaction=clear

```

ロードウォ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

```

3.6. メッシュ VPN の設定

any-to-any VPN と呼ばれるメッシュ 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。

以下の手順では、X.509 証明書を使用します。これらの証明書は、Dogtag Certificate System などのいかなる種類の認証局 (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。

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
  type=passthrough
  authby=never
  left=%defaultroute
  right=%group

conn private
  auto=ondemand
  type=transport
  authby=rsasig
  failureshunt=drop
  negotiationshunt=drop
  # left
  left=%defaultroute
  leftcert=nodeXXXX
  leftid=%fromcert
  leftrsasigkey=%cert
  # right
  rightrsasigkey=%cert
  rightid=%fromcert
  right=%opportunisticgroup

conn private-or-clear
  auto=ondemand
```

```

type=transport
authby=rsasig
failureshunt=passthrough
negotiationshunt=passthrough
# left
left=%defaultroute
leftcert=nodeXXXX
leftid=%fromcert
    leftrsasigkey=%cert
# right
rightrsasigkey=%cert
rightid=%fromcert
right=%opportunisticgroup

```

- 適切なカテゴリーに、ネットワークの IP アドレスを追加します。たとえば、すべてのノードが 10.15.0.0/16 ネットワークにある場合は、すべてのノードに **IPsec** 暗号が必要です。

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

- 特定のノード (例: 10.15.34.0/24) を、**IPsec** を使用または使用せずに機能させるには、以下の設定を使用して、これらのノードを `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
```

- 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、例外の一覧が同じか、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。

- 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
```

- ノードを追加したら、`ping` コマンドで **IPsec** トンネルを開くだけで十分です。ノードが開くトンネルを確認するには、次のコマンドを実行します。

```
# ipsec trafficstatus
```

3.7. LIBRESWAN で使用される認証方法

終了点の認証には、以下の方法を使用できます。

- Pre-Shared Keys (PSK)** は、最も簡単な認証メソッドです。PSK はランダムな文字で構成されており、長さが 20 文字以上になります。FIPS モードでは、PSK が、使用する整合性アルゴリズムにより、最低強度の要件を満たす必要があります。PSK の値は 64 文字以上にすることが推奨されます。

- **生の 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。ホストは、相互の公開 RSA 鍵を使用して手動で設定します。この方法は、1 ダース以上のホストで、互いに **IPsec** トンネルを設定する必要がある場合には、適切に調整されません。
- **X.509 証明書** は、共通の **IPsec** ゲートウェイへの接続が必要になるホストが多数存在する、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 証明書の署名に使用されます。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。
- **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、インターネット規模の日和見 IPsec にも使用できます。この場合、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。

量子コンピューターに対する保護

これらの認証方法に加え、**Postquantum Preshared Keys (PPK)** メソッドを使用して、量子コンピューターからの考えられる攻撃から保護できます。個々のクライアントまたはクライアントグループは、帯域幅を設定した事前共有鍵に対応する (**PPKID**) を指定して、独自の **PPK** を使用できます。

事前共有鍵が設定されている **IKEv1** を使用すると、量子攻撃者に対する保護が可能になります。**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、**PPK (Postquantum Preshared Keys)** を使用して、量子攻撃に対して **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、辞書の単語は使用しません。PPK ID および PPK データ自体は **ipsec.secrets** に保存されます。以下に例を示します。

```
@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、動的 **PPK** に基づいたワンタイムパッドを使用します。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材料が残っていないと、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

動的の **PPK** の実装はテクノロジープレビューとして提供されており、この機能は注意して使用する必要があります。

3.8. 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. RHEL 8 でシステムを FIPS モードに切り替えるには、次のコマンドを実行します。

```
# 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
```

検証手順

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一覧を表示するには、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

- システム全体の暗号化ポリシーの使用

3.9. 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。以下の手順を使用して、パスワード保護を追加します。



注記

以前のリリースの RHEL 6.6 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が含まれる `/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
```

検証手順

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、**Journal** ログに、初期化が成功したことを確認するエントリーが含まれていることを確認します。

```
# journalctl -u ipsec
...
pluto[23001]: NSS DB directory: sql:/etc/ipsec.d
pluto[23001]: Initializing NSS
pluto[23001]: Opening NSS database "sql:/etc/ipsec.d" read-only
pluto[23001]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[23001]: NSS crypto library initialized
...
```

関連情報

- man ページの **certutil(1)**
- FIPS 140-2 に関する認定の詳細は、ナレッジベースの記事「[Government Standards](#)」を参照してください。

3.10. 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

接続のシステム全体の crypto-policies のオーバーライド

RHEL のシステム全体の暗号化ポリシーは、**%default** と呼ばれる特別な接続を作成します。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、次の構成では、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) を使用する接続が許可されます。

```
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。

すべての接続で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

- 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。

3.11. 関連情報

以下の資料では、**Libreswan** および **ipsec** デーモンに関する追加情報を提供します。

インストールされているドキュメント

- man ページの **ipsec (8)** - **ipsec** のコマンドオプションが説明されています。
- man ページの **ipsec.conf (5)** - **ipsec** の設定情報が記載されています。
- man ページの **ipsec.secrets (5)** - **ipsec.secrets** ファイルの形式が説明されています。
- man ページの **ipsec_auto (8)** - 鍵の自動交換を使用して確立された Libreswan IPsec 接続を操作する **auto** コマンドラインクライアントの使用方法が説明されています。
- man ページの **ipsec_rsasigkey (8)** - RSA 署名鍵の生成に使用するツールが説明されています。
- `/usr/share/doc/libreswan-version/`

オンラインドキュメント

<https://libreswan.org>

アップストリームプロジェクトの Web サイトです。

<https://libreswan.org/wiki>

Libreswan プロジェクトの Wiki です。

<https://libreswan.org/man/>

Libreswan に関する全 man ページ

[NIST Special Publication 800-77: Guide to IPsec VPNs](#)

IPsec に基づくセキュリティーサービスの実装に関する組織への実用的なガイダンス。

第4章 MACSEC の設定

次のセクションでは、イーサネットリンクのすべてのトラフィックで、安全な通信のために、802.1AE IEEE 標準セキュリティー技術である **MACsec (Media Control Access Security)** を設定する方法を説明します。

4.1. MACSEC の概要

MACsec (Media Access Control Security (IEEE 802.1AE)) は、LAN におけるすべてのトラフィックを、GCM-AES-128 アルゴリズムで認証します。**MACsec** は、**IP** だけでなく、ARP (Address Resolution Protocol)、ND (Neighbor Discovery)、または **DHCP** も保護できます。**IPsec** はネットワーク層 (レイヤー3) で機能しますが、**SSL** または **TLS** はアプリケーション層 (レイヤー7) で機能し、**MACsec** はデータリンク層 (レイヤー2) で機能します。**MACsec** を、その他のネットワーク層のセキュリティープロトコルと組み合わせて、これらの標準規格が提供するさまざまなセキュリティー機能を活用します。

4.2. NMCLI ツールで MACSEC の使用

この手順は、**nmcli** ツールを使用して **MACsec** を設定する方法を説明します。

前提条件

- **NetworkManager** が実行している。
- 16 バイトの16 進数表記 CAK (**\$MKA_CAK**) と、32 バイトの16 進数表記 CKN (**\$MKA_CKN**) がある。

手順

```
~]# nmcli connection add type macsec \
con-name test-macsec+ ifname macsec0 \
connection.autoconnect no \
macsec.parent enp1s0 macsec.mode psk \
macsec.mka-cak $MKA_CAK \
macsec.mka-ckn $MKA_CKN

~]# nmcli connection up test-macsec+
```

これにより、**macsec0** デバイスが設定され、ネットワークに使用できます。

4.3. WPA_SUPPLICANT で MACSEC の使用

この手順は、事前に共有された CAK/CKN (Connectivity Association Key/CAK Name) のペアで認証を実行するスイッチを使用して、**MACsec** を有効にする方法を説明します。

手順

1. CAK/CKN ペアを作成します。たとえば、次のコマンドにより、16 バイトの鍵が16 進数表記で生成されます。

```
~]# dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%02x"'
```

2. **wpa_supplicant.conf** 設定ファイルを作成し、次の行を追加します。

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=3
ap_scan=0
fast_reauth=1

network={
    key_mgmt=NONE
    eapol_flags=0
    macsec_policy=1

    mka_cak=0011... # 16 bytes hexadecimal
    mka_ckn=2233... # 32 bytes hexadecimal
}
```

wpa_supplicant.conf 設定ファイルの **mka_cak** 行および **mka_ckn** 行には、前の手順の値を使用します。

詳細は、man ページの **wpa_supplicant.conf(5)** を参照してください。

3. ネットワークの接続に、たとえば **wlp61s0** を使用している場合は、次のコマンドを実行して **wpa_supplicant** を起動します。

```
~]# wpa_supplicant -i wlp61s0 -Dmacsec_linux -c wpa_supplicant.conf
```

4.4. 関連情報

詳細は「[What's new in MACsec: setting up MACsec using wpa_supplicant and \(optionally\) NetworkManager](#)」を参照してください。また、**MACsec** ネットワークのアーキテクチャー、ユースケースシナリオ、設定例の詳細は、「[MACsec: a different solution to encrypt network traffic](#)」を参照してください。

第5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。

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。

5.1. 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** - **firewalld** ユーティリティを使用して、ワークステーションにファイアウォールを設定します。このユーティリティは使いやすく、このシナリオの一般的な使用例に対応しています。
- **nftables** - **nftables** ユーティリティを使用して、ネットワーク全体など、複雑なファイアウォールを設定します。
- **iptables** - **iptables** ユーティリティは、Red Hat Enterprise Linux 8 では非推奨になりました。代わりに **nftables** を使用します。



重要

異なるファイアウォールサービスが相互に影響することを回避するには、RHEL ホストでそのうちの1つだけを実行し、他のサービスを無効にします。

5.2. FIREWALLD の使用

5.2.1. firewalld

firewalld は、**D-Bus** インターフェースを使用して、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ースおよびソースをゾーンに割り当てるができます。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 **開く** 必要があります。**firewalld** は、明示的に開い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。

関連情報

- [man ページの firewalld\(1\)](#)

5.2.2. ゾーン

firewalld は、インターフェースに追加する信頼レベルと、そのネットワークのトラフィックに従って、複数のネットワークを複数のゾーンに分類できます。接続は、1つのゾーンにしか指定できませんが、ゾーンは多くのネットワーク接続に使用できます。

NetworkManager は、**firewalld** にインターフェースのゾーンを通知します。以下を使用して、ゾーンをインターフェースに割り当てることができます。

- **NetworkManager**
- **firewall-config** ツール
- **firewall-cmd** コマンドラインツール
- RHEL Web コンソール

後者の3つは、適切な **NetworkManager** 設定ファイルの編集のみを行います。Web コンソールを使用してインターフェースのゾーンを変更する (**firewall-cmd** または **firewall-config**) と、リクエストが **NetworkManager** に転送され、**firewalld** では処理されません。

事前定義したゾーンは `/usr/lib/firewalld/zones/` ディレクトリーに保存され、利用可能なネットワークインターフェースに即座に適用されます。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。システムで開始したネットワーク接続のみが可能です。

dmz

公開アクセスは可能ですが、内部ネットワークへのアクセスに制限がある非武装地帯にあるコンピューター向けです。選択した着信接続のみが許可されます。

drop

着信ネットワークパケットは、通知なしで遮断されます。発信ネットワーク接続だけが可能です。

external

マスカレードをルーター用に特別に有効にした外部ネットワークでの使用向けです。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターを信頼しません。選択した着信接続のみが許可されます。

home

そのネットワークでその他のコンピューターをほぼ信頼できる自宅での使用向けです。選択した着信接続のみが許可されます。

internal

そのネットワークでそ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向けです。選択した着信接続のみが許可されます。

public

そのネットワークでその他のコンピューターを信頼できないパブリックエリア向けです。選択した着信接続のみが許可されます。

trusted

すべてのネットワーク接続が許可されます。

work

そのネットワークで、その他のコンピューターをほぼ信頼できる職場での使用向けです。選択した着信接続のみが許可されます。

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。インターフェース接続を **NetworkManager** に追加すると、デフォルトゾーンに割り当てられます。**firewalld** のデフォルトゾーンは、インストール時に **public** ゾーンに設定されます。デフォルトゾーンは変更できます。



注記

ネットワークゾーン名は、分かりやすく、ユーザーが妥当な決定をすばやく下せるような名前が付けられています。セキュリティ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

- man ページの **firewalld.zone(5)**

5.2.3. 事前定義サービス

サービスが、ローカルポート、プロトコル、ソースポート、宛先、そしてサービスが有効になると自動的に読み込まれるファイアウォールのヘルパーモジュールの一覧を指す場合があります。サービスを使用すると、ポートのオープン、プロトコルの定義、パケット転送の有効化などを1つ1つ行うのではなく、1回のステップで定義できます。

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。

サービスは、グラフィカルな **firewall-config** ツールと、**firewall-cmd** および **firewall-offline-cmd** を使用して追加または削除できます。

または、**/etc/firewalld/services/** ディレクトリーのXML ファイルを変更できます。ユーザーがサービスを追加または変更しないと、**/etc/firewalld/services/** には、対応するXML ファイルが記載されません。**/usr/lib/firewalld/services/** ディレクトリーのファイルは、サービスを追加または変更する場合にテンプレートとして使用できます。

関連情報

- man ページの **firewalld.service(5)**

5.3. FIREWALL-CONFIG GUI 設定ツールのインストール

firewall-config GUI 設定ツールを使用するには、**firewall-config** パッケージをインストールします。

手順

1. **root** で次のコマンドを実行します。

```
# yum install firewall-config
```

また、**GNOME** で、**Super** キーを使用して Software と入力し、**Software Sources** アプリケーションを起動します。右上端で検索ボタンを選択すると表示される検索ボックスに **firewall** と入力します。検索結果から **Firewall** アイテムを選択し、**Install** ボタンをクリックします。

2. **firewall-config** を実行するために、**firewall-config** コマンドを実行するか、**Super** キーを押してアクティビティーの概要を開き、**firewall** と入力して **Enter** を押します。

5.4. FIREWALLD の現在の状況および設定の表示

5.4.1. firewalld の現在の状況の表示

ファイアウォールサービス **firewalld** は、システムにデフォルトでインストールされています。CLI インターフェース **firewalld** を使用して、サービスが実行していることを確認します。

手順

1. サービスの状況を表示するには、次のコマンドを実行します。

```
# firewall-cmd --state
```

2. サービスの状況の詳細は、**systemctl status** サブコマンドを実行します。

```
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
Docs: man:firewalld(1)
Main PID: 705 (firewalld)
Tasks: 2 (limit: 4915)
CGroup: /system.slice/firewalld.service
└─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

関連情報

設定を編集する前に、**firewalld** の設定方法と、強制するルールを確認することが重要です。ファイアウォール設定を表示する場合は、「[現在の firewalld 設定の表示](#)」を参照してください。

5.4.2. 現在の firewalld 設定の表示

5.4.2.1. GUI を使用して許可されるサービスの表示

グラフィカルな **firewall-config** ツールを使用してサービスの一覧を表示する場合は、**Super** キーを押してアクティビティーの概要を開き、**firewall** と入力して **Enter** を押します。**firewall-config** ツールが表示されます。**Services** タブの下にサービスの一覧が表示されます。

もしくは、コマンドラインを使用してグラフィカルなファイアウォール設定ツールを起動する場合は、次のコマンドを入力します。

```
$ firewall-config
```

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。

5.4.2.2. CLI を使用した `firewalld` 設定の表示

CLI クライアントで、現在のファイアウォール設定を、複数の方法で表示できます。 `--list-all` オプションは、`firewalld` 設定の完全概要を表示します。

`firewalld` は、ゾーンを使用してトラフィックを管理します。 `--zone` オプションでゾーンを指定しないと、コマンドは、アクティブネットワークインターフェースおよび接続に割り当てたデフォルトゾーンに対して有効になります。

デフォルトゾーンに関連する情報をすべて表示するには、次のコマンドを実行します。

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

設定を表示するゾーンを指定するには、たとえば、 `--zone=zone-name` 引数を `firewall-cmd --list-all` コマンドに指定します。

```
# firewall-cmd --list-all --zone=home
home
target: default
icmp-block-inversion: no
interfaces:
sources:
services: ssh mdns samba-client dhcpv6-client
... [trimmed for clarity]
```

サービス、ポートなど、特定情報の設定を確認するには、特定のオプションを使用します。 `man` ページの `firewalld` か、コマンドの `help` でオプションの一覧を表示します。

```
# firewall-cmd --help

Usage: firewall-cmd [OPTIONS...]

General Options
-h, --help          Prints a short help text and exists
-V, --version       Print the version string of firewalld
-q, --quiet         Do not print status messages

Status Options
--state             Return and print firewalld state
--reload           Reload firewall and keep state information
... [trimmed for clarity]
```

たとえば、現在のゾーンで許可されているサービスを表示します。

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



注記

CLI ツールを使用して一覧表示した特定のサブパートの設定は、解釈が難しいことがしばしばあります。たとえば、**firewalld** で **SSH** サービスを許可し、そのサービスに必要なポート (22) を開くことができます。許可されたサービスを一覧表示すると、一覧には **SSH** サービスが表示されますが、開いているポートを一覧表示しても、何も表示されません。したがって、**--list-all** オプションを使用して、完全な情報を取得することが推奨されます。

5.5. FIREWALLD の起動

手順

1. **firewalld** を開始するには、**root** で次のコマンドを実行します。

```
# systemctl unmask firewalld
# systemctl start firewalld
```

2. システムの起動時に **firewalld** を自動的に起動するように設定するには、**root** で次のコマンドを実行します。

```
# systemctl enable firewalld
```

5.6. FIREWALLD の停止

手順

1. **firewalld** を停止するには、**root** で次のコマンドを実行します。

```
# systemctl stop firewalld
```

2. システムの起動時に **firewalld** を自動的に起動しないように設定するには、次のコマンドを実行します。

```
# systemctl disable firewalld
```

3. **firewalld D-Bus** インターフェースにアクセスして **firewalld** を起動していないこと、そしてその他のサービスが **firewalld** を求めているかどうかを確認するには、次のコマンドを実行します。

```
# systemctl mask firewalld
```

5.7. ランタイムおよび永続化の設定

`runtime` モードで行った変更は、`firewalld` が実行している間しか適用されません。`firewalld` を再起動すると、設定内容は永続的な値に戻ります。

変更した内容を再起動後も持続させる場合は、`--permanent` オプションを使用して適用します。`firewalld` が実行している間だけ変更を持続させる場合は、`--runtime-to-permanent firewall-cmd` オプションを使用します。

`--permanent` オプションのみを使用して `firewalld` を実行している場合にルールを設定すると、`firewalld` が再起動するまで有効にはなりません。ただし、`firewalld` を再起動すると、開いているポートがすべて閉じ、ネットワークトラフィックを停止します。

CLI を使用したランタイムおよび永続化の設定の変更

CLI では、2 つのモードのファイアウォール設定を同時に修正することができません。CLI では、ランタイムまたは永続モードを修正します。永続化モードでファイアウォール設定を修正するには、`firewall-cmd` コマンドで `--permanent` オプションを使用します。

```
# firewall-cmd --permanent <other options>
```

このオプションを使用しないと、コマンドはランタイムモードを変更します。

両方のモードで設定を変更する場合は、2 つの方法を使用できます。

1. 以下のように、ランタイム設定を変更して、永続化します。

```
# firewall-cmd <other options>
# firewall-cmd --runtime-to-permanent
```

2. 永続的な設定を行い、ランタイムモードで設定を再ロードします。

```
# firewall-cmd --permanent <other options>
# firewall-cmd --reload
```

最初の方法では、永続モードで設定を適用する前に、設定をテストできます。

注記

特にリモートシステムでは、設定を間違えると、ユーザーが自身をロックする結果となります。そのような状況を回避するには、`--timeout` オプションを使用します。指定した時間が経つと、変更は元に戻ります。このオプションを使用した場合は、`--permanent` オプションが無効になります。

たとえば、15 分間 **SSH** サービスを追加するには、次のコマンドを実行します。

```
# firewall-cmd --add-service=ssh --timeout 15m
```

5.8. 永続的な FIREWALLD 設定の確認

`firewalld` 設定ファイルを手動で編集した後など、特定の状況では、変更が正しいことを管理者が確認します。本セクションでは、`firewalld` サービスの永続的な設定を確認する方法を説明します。

前提条件

- `firewalld` サービスが実行している。

手順

1. **firewalld** サービスの永続的な設定を確認します。

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

永続的な設定が有効になると、コマンドが **success** を返します。その他の場合は、以下のような詳細で、コマンドがエラーを返します。

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

5.9. FIREWALLD でネットワークトラフィックの制御

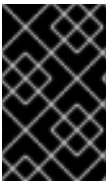
5.9.1. 緊急時に CLI を使用してすべてのトラフィックの無効化

システムへの攻撃などの緊急な状態にあるとき、すべてのネットワークトラフィックを無効にし、攻撃を遮断できます。

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
```



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。

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
```

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
```

5.9.2. CLI を使用して事前定義されたサービスで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せん。

手順

1. サービスが許可されていないことを確認します。

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

2. 事前定義したサービスの一覧を表示します。

```
# 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 ...
[trimmed for clarity]
```

3. サービスを、許可されたサービスに追加します。

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

4. 新しい設定を永続化します。

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

5.9.3. GUI を使用して事前定義サービスでトラフィックを制御

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。

1. **firewall-config** ツールを起動して、サービスを設定するネットワークゾーンを選択します。
2. **Services** タブを選択します。
3. 信頼するサービスタイプのチェックボックスを選択してください。ブロックするサービスタイプのチェックボックスの選択は解除してください。

サービスを編集するには、以下を行います。

1. **firewall-config** ツールを起動します。
2. **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。
3. 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

ランタイム モードでは、サービス設定を変更できません。

5.9.4. 新しいサービスの追加

サービスは、グラフィカルな **firewall-config** ツールと、**firewall-cmd** および **firewall-offline-cmd** を使用して追加または削除できます。または、**/etc/firewalld/services/** にある XML ファイルを編集できます。ユーザーがサービスを追加または変更しないと、対応する XML ファイルが

`/etc/firewalld/services/` に作成されません。`/usr/lib/firewalld/services/` のファイルは、サービスを追加または変更する際にテンプレートとして使用できます。



注記

サービス名は英数字にする必要があります。_(下線) 文字および-(ハイフン) 文字も使用できます。

手順

`firewalld` がアクティブでない場合に、ターミナルで新しいサービスを追加するには、`firewall-cmd` または `firewall-offline-cmd` を使用します。

1. 新しい、空のサービスを追加するには、次のコマンドを実行します。

```
$ firewall-cmd --new-service=service-name --permanent
```

2. ローカルファイルを使用して新規サービスを追加するには、次のコマンドを使用します。

```
$ firewall-cmd --new-service-from-file=service-name.xml --permanent
```

追加オプション `--name=service-name` を指定して、サービス名を変更できます。

3. サービス設定を変更すると、直ちにサービスの更新コピーが `/etc/firewalld/services/` に作成できます。

`root` で次のコマンドを実行して、サービスを手動でコピーします。

```
# cp /usr/lib/firewalld/services/service-name.xml /etc/firewalld/services/service-name.xml
```

`firewalld` は、最初に `/usr/lib/firewalld/services` のファイルを読み込みます。ファイルは `/etc/firewalld/services` に置かれ、そのファイルが有効な場合は、`/usr/lib/firewalld/services` で一致するファイルを上書きします。`/usr/lib/firewalld/services` で上書きしたファイルは、`/etc/firewalld/services` で一致するファイルが削除されるとすぐに、もしくはサービスのデフォルトを読み込むように `firewalld` が求められた場合に使用されます。これに該当するのは永続環境のみです。ランタイム環境でフォールバックさせるには、再読み込みが必要です。

5.9.5. CLI を使用したポートの制御

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、区別し、システムサービスに従って転送する論理デバイスです。これは、通常、ポートをリッスンするデーモンにより示されますが、このポートに入るトラフィックを待ちます。

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、デフォルトでは、システム管理者は、セキュリティを強化するため、またはその他の理由により、別のポートをリッスンするようにデーモンを設定します。

5.9.5.1. ポートを開く

開かれたポートを介して、システムが外部からアクセスできます。これはセキュリティリスクでもあります。一般的に、ポートを閉じたままにし、特定サービスに要求される場合に限り開きます。

手順

現在のゾーンで開かれたポートの一覧を表示するには、以下を行います。

1. 許可されているポートの一覧を表示します。

```
# firewall-cmd --list-ports
```

2. 許可されているポートにポートを追加して、着信トラフィックに対してそのポートを開きます。

```
# firewall-cmd --add-port=port-number/port-type
```

3. 新しい設定を永続化します。

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

ポートタイプは、**tcp**、**udp**、**sctp**、または **dccp** になります。このタイプは、ネットワーク接続の種類と一致させる必要があります。

5.9.5.2. ポートを閉じる

開いているポートが必要なくなった場合に、**firewalld** のポートを閉じます。ポートをそのままにするとセキュリティーリスクとなるため、使用されなくなったらすぐに不要なポートを閉じることが強く推奨されます。

手順

ポートを閉じるには、許可されているポートの一覧からそれを削除します。

1. 許可されているポートの一覧を表示します。

```
# firewall-cmd --list-ports
[WARNING]
```

```
====
```

```
This command will only give you a list of ports that have been opened as ports. You will not be able to see any open ports that have been opened as a service. Therefore, you should consider using the --list-all option instead of --list-ports.
```

```
====
```

2. 「許可されているポート」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
```

3. 新しい設定を永続化します。

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

5.9.6. GUI を使用してポートを開く

ファイアウォールを経由して特定のポートに向かうトラフィックを許可するには、以下を行います。

1. **firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。
2. 右側の **Ports** タブを選択し、**Add** ボタンをクリックします。**Port and Protocol** ウィンドウが開きます。

- 許可するポート番号またはポートの範囲を入力します。
- リストから **tcp** または **udp** を選択します。

5.9.7. 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、以下を行います。

- firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。
- 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。
- 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。

5.9.8. GUI を使用してソースポートを開く

特定ポートから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、以下を行います。

- 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。
- 右側の **Source Port** タブを選択し、**Add** ボタンをクリックします。**Source Port** ウィンドウが開きます。
- 許可するポート番号またはポートの範囲を入力します。リストから **tcp** または **udp** を選択します。

5.10. 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ー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。

5.10.1. ゾーンの一覧

手順

- システムで利用可能なゾーンを確認するには、次のコマンドを実行します。

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

firewall-cmd --get-zones コマンドは、システムで利用可能な全てのゾーンを表示し、特定ゾーンの詳細は表示しません。

- すべてのゾーンで詳細情報を表示する場合は、次のコマンドを実行します。

```
# firewall-cmd --list-all-zones
```

- 特定ゾーンに関する詳細情報を表示する場合は、次のコマンドを実行します。

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

5.10.2. 特定ゾーンに対する `firewalld` 設定の修正

「CLI を使用して事前定義されたサービスでトラフィックの制御」および「CLI を使用したポートの制御」は、現在作業中のゾーンの範囲にサービスを追加するか、またはゾーンの範囲にあるポートを修正する方法を説明します。別のゾーンへのルールの設定が必要になる場合もあります。

手順

- 別のゾーンに指定するには、`--zone=zone-name` オプションを使用します。たとえば、`public` ゾーンで `SSH` サービスを許可するには、次のコマンドを実行します。

```
# firewall-cmd --add-service=ssh --zone=public
```

5.10.3. 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ー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ースは、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。

手順

デフォルトゾーンを設定するには、以下を行います。

- 現在のデフォルトゾーンを表示します。

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

- 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。

5.10.4. ゾーンへのネットワークインターフェースの割り当て

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ー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

特定インターフェースにゾーンを割り当てるには、以下を行います。

- 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ースを一覧表示します。

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

2. 別のゾーンにインターフェースを割り当てます。

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

5.10.5. nmcli を使用して接続にゾーンを割り当て

この手順では、**nmcli** ユーティリティーを使用して、ファイアウォールゾーンを NetworkManager 接続に追加する方法を説明します。

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 profile modify connection.zone zone_name
```

2. 接続の再読み込みを行います。

```
# nmcli connection up profile
```

5.10.6. 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

1. 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て行を追加します。

```
ZONE=zone_name
```

5.10.7. 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。

手順

新しいゾーンを作成するには、以下を行います。

1. 新しいゾーンを作成します。

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

2. 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
```

3. 新しい設定を永続化します。

-

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

5.10.8. ゾーンの設定ファイル

また、**ゾーンの設定ファイル** を使用してゾーンを作成できます。このアプローチは、新しいゾーンを作成する必要がある場合に、別のゾーンの設定を変更して利用する場合に便利です。

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 port="1025-65535" protocol="tcp"/>
  <port port="1025-65535" protocol="udp"/>
</zone>
```

そのゾーンの設定を変更するには、セクションを追加または削除して、ポート、転送ポート、サービスなどを追加します。

関連情報

- 詳細は、man ページの **firewalld.zone** を参照してください。

5.10.9. 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。オプションは、**default**、**ACCEPT**、**REJECT**、および **DROP** の4つになります。ターゲットを **ACCEPT** に設定すると、特定ルールで無効にした着信パケット以外のパケットをすべて許可します。**REJECT** または **DROP** にターゲットを設定すると、特定のルールで許可したパケット以外の着信パケットがすべて無効になります。パケットが拒否されるとソースマシンに通知されますが、パケットが破棄される時は情報が送信されません。

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一覧表示して、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 --zone=zone-name --set-target=<default|ACCEPT|REJECT|DROP>
```

5.11. 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

5.11.1. 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。これにより、着信トラフィックを分類し、複数のゾーンに向け、トラフィックにより到達できるサービスを許可または拒否できます。

ソースをゾーンに追加する場合は、ゾーンがアクティブになり、そのソースからの着信トラフィックは、それを介して行われます。各ゾーンに異なる設定を指定できますが、それは指定したソースから順次トラフィックに適用されます。ネットワークインターフェースが1つしかない場合でも、複数のゾーンを使用できます。

5.11.2. ソースの追加

着信トラフィックを特定のソースに転送する場合は、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。

- 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. 利用可能なゾーンの一覧を表示します。

```
# 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
```

5.11.3. ソースの削除

ゾーンからソースを削除すると、そのゾーンからのトラフィックを遮断します。

手順

1. 必要なゾーンに対して許可されているソースの一覧を表示します。

```
# 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
```

5.11.4. ソースポートの追加

発信源となるポートに基づいたトラフィックの分類を有効にするには、**--add-source-port** オプションを使用してソースポートを指定します。**--add-source** オプションと組み合わせて、トラフィックを特定の IP アドレスまたは IP 範囲に制限できます。

手順

1. ソースポートを追加するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-source-port=<port-name>/<tcp|udp|sctp|dccp>
```

5.11.5. 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

1. 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>
```

5.11.6. 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、その他のトラフィックをブロックしつつ、192.168.1.0/24 からのトラフィックを許可し、HTTP サービスに到達できるようにします。

手順

1. 利用可能なゾーンの一覧を表示します。

```
# firewall-cmd --get-zones  
block dmz drop external home internal public trusted work
```

2. 信頼されるゾーンにソースを追加して、ゾーンを経由してソースから発信するトラフィックに転送します。

```
# firewall-cmd --zone=trusted --add-source=192.168.1.0/24
```

3. 信頼ゾーンに http サービスを追加するには、次のコマンドを実行します。

```
# firewall-cmd --zone=trusted --add-service=http
```


4. 新しい設定を永続化します。

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

5. 信頼されるゾーンがアクティブで、サービスが許可されているのを確認します。

```
# firewall-cmd --zone=trusted --list-all
trusted (active)
target: ACCEPT
sources: 192.168.1.0/24
services: http
```

5.11.7. プロトコルに基づいてゾーンが許可したトラフィックの設定

プロトコルに基づいて、ゾーンが着信トラフィックを許可できます。指定したプロトコルを使用したすべてのトラフィックがゾーンにより許可されていますが、そこにさらにルールおよびフィルタリングを適用できます。

5.11.7.1. ゾーンへのプロトコルの追加

特定ゾーンへプロトコルを追加すると、このゾーンが許可するこのプロトコルを使用するすべてのトラフィックを許可します。

手順

1. プロトコルをゾーンに追加するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-protocol=port-name/tcp|udp|sctp|dccp|igmp
```



注記

マルチキャストトラフィックを受けけるには、**--add-protocol** オプションで **igmp** 値を使用します。

5.11.7.2. ゾーンからプロトコルの削除

特定ゾーンからプロトコルを削除するには、ゾーンにより、このプロトコルに基づいたすべてのトラフィックの許可を停止します。

手順

1. ゾーンからプロトコルを削除するには、次のコマンドを削除します。

```
# firewall-cmd --zone=zone-name --remove-protocol=port-name/tcp|udp|sctp|dccp|igmp
```

5.12. IP アドレスのマスカレードの設定

以下の手順では、システムでIP マスカレードを有効にする方法を説明します。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. この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

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 --permanent
```

5.13. ポート転送

この方法を使用するポートのリダイレクトは、IPv4 ベースのトラフィックでのみ機能します。IPv6 リダイレクト設定には、リッチルールを使用する必要があります。

外部システムにリダイレクトするには、マスカレードを有効にする必要があります。詳細は、[「IP アドレスのマスカレードの設定」](#)を参照してください。

5.13.1. リダイレクトするポートの追加

firewalld を使用して、システムで特定のポートを到達するための着信トラフィックが、選択した別の内部ポート、または別のマシンの外部ポートに配信されるようにポートのリダイレクトを設定できます。

前提条件

- あるポートから別のポートにトラフィックをリダイレクトする前に、パケットが到達するポート、使用されるプロトコル、リダイレクト先を確認しておく必要があります。

手順

ポートを別のポートにリダイレクトする場合は、次のコマンドを実行します。

```
# firewall-cmd --add-forward-port=port=port-number:proto=tcp|udp|sctp|dccp:toport=port-number
```

別の IP アドレスで、別のポートにポートをリダイレクトする場合は、次のコマンドを実行します。

1. 転送するポートを追加します。

```
# firewall-cmd --add-forward-port=port=port-number:proto=tcp|udp:toport=port-number:toaddr=IP/mask
```

2. マスカレードを有効にします。

```
# firewall-cmd --add-masquerade
```

5.13.2. 同一マシンでTCP ポート 80 からポート 88 へのリダイレクト

TCP ポート 80 をポート 88 にリダイレクトするには、以下の手順に従います。

手順

1. TCP トラフィックに対して、ポート 80 からポート 88 へリダイレクトします。

```
# firewall-cmd --add-forward-port=port=80:proto=tcp:toport=88
```

2. 新しい設定を永続化します。

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

3. そのポートがリダイレクトされていることを確認します。

```
# firewall-cmd --list-all
```

5.13.3. リダイレクトしているポートの削除

リダイレクトしているポートを削除するには、次のコマンドを実行します。

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

別のアドレスにリダイレクトした転送ポートを削除する場合は、以下の手順を使用します。

手順

1. 転送したポートを削除するには、以下を行います。

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

2. マスカレードを無効にするには、次のコマンドを実行します。

```
~]# firewall-cmd --remove-masquerade
```

5.13.4. 同じマシンでTCP ポート 88 に転送されるポート 80 の削除

ポートのリダイレクトを削除するには、以下を行います。

手順

1. リダイレクトしたポートの一覧を表示します。

```
~]# firewall-cmd --list-forward-ports
port=80:proto=tcp:toport=88:toaddr=
```

2. ファイアウォールからリダイレクトしたポートを削除します。

```
~]# firewall-cmd --remove-forward-port=port=80:proto=tcp:toport=88:toaddr=
```

3. 新しい設定を永続化します。

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

5.14. ICMP リクエストの管理

Internet Control Message Protocol (ICMP) は、接続問題(要求されているサービスが利用できないなど)を示すエラーメッセージと運用情報を送信するために、様々なネットワークデバイスにより使用されてい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。

ただし、**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** リクエストをブロックできます。

5.14.1. ICMP リクエストの一覧表示およびブロック

ICMP リクエストの一覧表示

ICMP リクエストは、`/usr/lib/firewalld/icmptypes/` ディレクトリーにある各XML ファイルで説明されています。リクエストの説明は、このファイルを参照してください。**firewall-cmd** コマンドは、**ICMP** リクエストの操作を制御します。

- 利用可能な **ICMP** タイプの一覧を表示するには、次のコマンドを実行します。

```
# firewall-cmd --get-icmptypes
```

- **ICMP** リクエストは、IPv4、IPv6、またはその両方のプロトコルで使用できます。**ICMP** リクエストが使用されてるプロトコルを表示するには、次のコマンドを実行します。

```
# firewall-cmd --info-icmptype=<icmptype>
```

- **ICMP** リクエストのステータスは、リクエストが現在ブロックされている場合は **yes**、ブロックされていない場合は **no** となります。**ICMP** リクエストが現在ブロックされているかどうかを確認するには、次のコマンドを実行します。

```
# firewall-cmd --query-icmp-block=<icmptype>
```

ICMP リクエストのブロックまたはブロックの解除

サーバーが **ICMP** リクエストをブロックした場合は、通常の情報提供されません。ただし、情報が全く提供されないというわけではありません。クライアントは、特定の **ICMP** リクエストがブロックされている(拒否されている)情報を受け取ります。**ICMP** リクエストは、特にIPv6 トラフィックを使用すると、接続問題が発生することがあるため、注意深く検討する必要があります。

- **ICMP** リクエストが現在ブロックされているかどうかを確認するには、次のコマンドを実行します。

```
# firewall-cmd --query-icmp-block=<icmptype>
```

- **ICMP** リクエストをブロックするには、次のコマンドを実行します。

```
# firewall-cmd --add-icmp-block=<icmptype>
```

- **ICMP** リクエストのブロックを削除するには、次のコマンドを実行します。

```
# firewall-cmd --remove-icmp-block=<icmptype>
```

情報を提供せずに **ICMP** リクエストのブロック

通常、**ICMP** リクエストをブロックすると、ブロックしていることをクライアントは認識します。したがって、ライブの IP アドレスを傍受している潜在的な攻撃者は、IP アドレスがオンラインであることを確認できます。この情報を完全に非表示にするには、**ICMP** リクエストをすべて破棄する必要があります。

- すべての **ICMP** リクエストをブロックして破棄するには、次のコマンドを実行します。
 1. ゾーンのターゲットを **DROP** に設定します。

```
# firewall-cmd --set-target=DROP
```

2. 新しい設定を永続化します。

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

これで、明示的に許可されるトラフィックを除き、**ICMP** リクエストを含むすべてのトラフィックが破棄されます。

- 特定の **ICMP** リクエストをブロックして破棄し、その他のリクエストを許可するには、以下を行います。
 1. ゾーンのターゲットを **DROP** に設定します。

```
# firewall-cmd --set-target=DROP
```

2. すべての **ICMP** リクエストを一度にブロックする、**ICMP** ブロックの反転を追加します。

```
# firewall-cmd --add-icmp-block-inversion
```

3. 許可する **ICMP** リクエストに **ICMP** ブロックを追加する場合は、次のコマンドを実行します。

```
# firewall-cmd --add-icmp-block=<icmptype>
```

4. 新しい設定を永続化します。

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

ブロックの反転 は、**ICMP** リクエストブロックの設定を反転します。そのため、ゾーンのターゲットが **DROP** に変更されたため、ブロックされていないリクエストはすべてブロックされます。ブロックされているリクエストはブロックされません。これは、リクエストのブロックを解除する場合は、ブロックコマンドを使用する必要があることを示しています。

- ブロックの反転を、完全許可の設定に戻すには、以下を行います。

1. ゾーンのターゲットを **default** または **ACCEPT** に戻すには、次のコマンドを設定します。

```
# firewall-cmd --set-target=default
```

2. **ICMP** リクエストに追加したすべてのブロックを削除します。

```
# firewall-cmd --remove-icmp-block=<icmptype>
```

3. **ICMP** ブロックの反転を削除します。

```
# firewall-cmd --remove-icmp-block-inversion
```

4. 新しい設定を永続化します。

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

5.14.2. GUI を使用した ICMP フィルターの設定

- **ICMP** フィルターを有効または無効にするには、**firewall-config** ツールを起動して、フィルターをかけるメッセージのネットワークゾーンを選択します。**ICMP** フィルター タブを選択し、フィルターをかける **ICMP** メッセージの各タイプのチェックボックスを選択します。フィルターを無効にするには、チェックボックスの選択を外します。これは方向ごとに設定され、デフォルトではすべてが許可されます。
- **ICMP** タイプを編集するには、**firewall-config** ツールを起動してから **設定** ラベルのあるメニューで **永続** モードを選択します。**サービス** ウィンドウの下部に新たなアイコンが表示されます。以下のダイアログで **はい** を選択し、マスカレーディングを有効にし、動作している別のマシンに転送します。
- **ICMP** フィルターの反転を有効にするには、右側の **フィルターの反転** チェックボックスをクリックします。マークがついた **ICMP** タイプだけが許可され、その他はすべて拒否されます。DROP ターゲットを使用するゾーンでは破棄されます。

5.15. FIREWALLD を使用した IP セットの設定および制御

firewalld で対応する IP セットタイプの一覧を表示するには、**root** で次のコマンドを実行します。

```
~]# firewall-cmd --get-ipset-types
hash:ip hash:ip,mark hash:ip,port hash:ip,port,ip hash:ip,port,net hash:mac hash:net hash:net,iface
hash:net,net hash:net,port hash:net,port,net
```

5.15.1. CLI を使用した IP セットオプションの設定

IP セットは、**firewalld** ゾーンでソースとして使用でき、リッチルールでソースとして使用できます。Red Hat Enterprise Linux で推奨される方法は、ダイレクトルールで **firewalld** を使用して作成した IP セットを使用する方法です。

- 永続的な環境で **firewalld** に認識されている IP セットの一覧を表示するには、次のコマンドを **root** で実行します。

```
# firewall-cmd --permanent --get-ipsets
```

- 新しいIP セットを追加するには、永続化環境を使用し、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --new-ipset=test --type=hash:net
success
```

上記のコマンドは、名前 **test** とタイプ **hash:net** で、**IPv4** の新しいIP セットを作成します。**IPv6** で使用するIP セットを作成する場合は、**--option=family=inet6** オプションを追加します。ランタイム環境で新しい設定を有効にするには、**firewalld** を再読み込みします。

- **root** で次のコマンドを実行して、新しいIP セットの一覧を表示します。

```
# firewall-cmd --permanent --get-ipsets
test
```

- IP セットの詳細は、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --info-ipset=test
test
type: hash:net
options:
entries:
```

この時点ではIP セットにエントリーがありません。

- IP セット **test** にエントリーを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --add-entry=192.168.0.1
success
```

上記のコマンドは、IP アドレス **192.168.0.1** をIP セットに追加します。

- IP セットの現在のエントリーを一覧表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

- IP アドレスの一覧を含むファイルを生成します。以下に例を示します。

```
# cat > iplist.txt <<EOL
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
EOL
```

IP セットのIP アドレスの一覧が含まれるファイルには、行ごとにエントリーが含まれている必要があります。ハッシュ、セミコロン、また空の行から始まる行は無視されます。

- **iplist.txt** ファイルからアドレスを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --add-entries-from-file=iplist.txt
success
```

- 拡張されたIP セットのエントリー一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

- IP セットからアドレスを削除し、更新したエントリ一覧を確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --remove-entries-from-file=iplist.txt
success
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

- IP セットをゾーンへのソースとして追加し、ゾーンを使用して、IP セットに記載されるアドレスから受信するすべてのトラフィックを処理します。たとえば、IP セットの **test** をソースとして **drop** ゾーンに追加し、IP セットの **test** の一覧に表示されるすべてのエントリから発信されるパケットをすべて破棄するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --zone=drop --add-source=ipset:test
success
```

ソースの **ipset:** プレフィックスは、ソースが IP セットで、IP アドレスまたはアドレス範囲ではない **firewalld** を示しています。

IP セットの作成および削除は、永続環境に限定されますが、その他の IP セットオプションは、**--permanent** オプションを使用しないランタイム環境で使用できます。



警告

Red Hat は、**firewalld** を介して管理していない IP セットを使用することは推奨しません。このような IP セットを使用すると、そのセットを参照する永続的なダイレクトルールが必要で、IP セットを作成するカスタムサービスを追加する必要があります。このサービスは、**firewalld** を起動する前に起動する必要があります。先に起動しておかないと、**firewalld** が、このセットを使用してダイレクトルールを追加できません。**/etc/firewalld/direct.xml** ファイルを使用して、永続的なダイレクトルールを追加できます。

5.16. 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構成されます。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。

5.16.1. 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 ~ 32767** の任意の数値に設定でき、値が小さい方が優先されます。

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。

5.16.2. リッチルールの優先度の設定

この手順では、**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説明します。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

1. 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。

2. 必要に応じて、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
  }
}
```

5.17. 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、ロックダウンのホワイト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。

5.17.1. CLI を使用したロックダウンの設定

- ロックダウンが有効になっているかどうかを確認するには、**root** で次のコマンドを使用します。

```
# firewall-cmd --query-lockdown
```

ロックダウンが有効な場合は、**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。

- ロックダウンを有効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --lockdown-on
```

- ロックダウンを無効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --lockdown-off
```

5.17.2. CLI を使用したロックダウンホワイトリストオプションの設定

ロックダウンのホワイトリストには、コマンド、セキュリティーのコンテキスト、ユーザー、およびユーザーID を追加できます。ホワイトリストのコマンドエントリーがアスタリスク「*」で終了している場合は、そのコマンドで始まるすべてのコマンドラインが一致することになります。「*」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。

- ここでのコンテキストは、実行中のアプリケーションやサービスのセキュリティー (SELinux) コンテキストです。実行中のアプリケーションのコンテキストを確認するには、次のコマンドを実行します。

```
$ ps -e --context
```

このコマンドは、実行中のアプリケーションをすべて返します。grep ツールを使用して、出力から目的のアプリケーションをパイプ処理します。以下に例を示します。

```
$ ps -e --context | grep example_program
```

- ホワイトリストにあるコマンドラインの一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-commands
```

- ホワイトリストに **command** コマンドを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

- ホワイトリストから **command** コマンドを削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

- **command** コマンドがホワイトリストに含まれる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 --query-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

このコマンドでは、含まれる場合は **yes** が出力され、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。

- ホワイトリストにあるセキュリティーコンテキストの一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-contexts
```

- ホワイトリストに **context** コンテキストを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-context=context
```

- ホワイトリストから **context** コンテキストを削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-context=context
```

- **context** コンテキストがホワイトリストに含まれる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 --query-lockdown-whitelist-context=context
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

- ホワイトリストにあるユーザーID の一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-uids
```

- ホワイトリストにユーザーID (**uid**) を追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-uid=uid
```

- ホワイトリストからユーザーID (**uid**) を削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-uid=uid
```

- ホワイトリストにユーザーID (**uid**) があるかどうかを確認するには、次のコマンドを実行します。

```
$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

- ホワイトリストにあるユーザー名の一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-users
```

- ホワイトリストにユーザー名 (**user**) を追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-user=user
```

- ホワイトリストからユーザー名 (**user**) を削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-user=user
```

- ホワイトリストにユーザー名 (**user**) があるかどうかを確認するには、次のコマンドを実行します。

```
$ firewall-cmd --query-lockdown-whitelist-user=user
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

5.17.3. 設定ファイルを使用したロックダウンのホワイトリストオプションの設定

デフォルトのホワイト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。

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist>
  <selinux context="system_u:system_r:NetworkManager_t:s0"/>
  <selinux context="system_u:system_r:virttd_t:s0-s0:c0.c1023"/>
  <user id="0"/>
</whitelist>
```

以下のホワイト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。または、以下のような明確なコマンドも使用できます。

```
/usr/bin/python3 /bin/firewall-cmd --lockdown-on
```

この例では、**--lockdown-on** コマンドだけが許可されます。

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** へのパスを使用しているのであれば、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ホワイト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、その名前が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。

5.18. 拒否されたパケットのログ

firewalld で **LogDenied** オプションを使用して、拒否したパケットに簡易ロギングメカニズムを追加できます。対象となるのは、拒否または破棄されるパケットになります。ログ設定を変更するには、**/etc/firewalld/firewalld.conf** ファイルを変更するか、コマンドラインまたは GUI 設定ツールを使用します。

LogDenied を有効にすると、デフォルトルールの INPUT チェイン、FORWARD チェイン、および OUTPUT チェインの reject ルールおよび drop ルールと、ゾーンの最後の reject ルールおよび drop ルールの直前に、ロギングルールが追加されます。ここに設定できる値は、**all**、**unicast**、**broadcast**、**multicast**、および **off** です。デフォルト設定は **off** です。**unicast**、**broadcast**、および **multicast** の設定では、リンク層のパケットタイプを一致させるのに **pkttype** 一致を使用します。**all** を使用すると、すべてのパケットがログに記録されます。

`firewall-cmd` で実際の **LogDenied** 設定を一覧表示するには、**root** で次のコマンドを使用します。

```
# firewall-cmd --get-log-denied
off
```

LogDenied 設定を変更するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --set-log-denied=all
success
```

`firewalld` の GUI 設定ツールを使用して **LogDenied** 設定を変更する場合は、`firewall-config` を起動して **Options** メニューをクリックし、**Change Log Denied** を選択します。**LogDenied** ウィンドウが表示されます。メニューから新しい **LogDenied** 設定を選択し、OK をクリックします。

5.19. 関連情報

以下の資料は、`firewalld` の関連情報を提供します。

インストールされているドキュメント

- `man` ページの **firewalld(1)** - `firewalld` のコマンドオプションが説明されています。
- `man` ページの **firewalld.conf(5)** - `firewalld` を設定する情報が含まれます。
- `man` ページの **firewall-cmd(1)** - `firewalld` コマンドラインクライアントのコマンドオプションが説明されています。
- `man` ページの **firewall-config(1)** - `firewall-config` ツールのデフォルト設定が説明されています。
- `man` ページの **firewall-offline-cmd(1)** - `firewalld` オフラインコマンドラインクライアントのコマンドオプションが説明されています。
- `man` ページの **firewalld.icmptype(5)** - **ICMP** フィルタリングの XML 設定ファイルが説明されています。
- `man` ページの **firewalld.ipset(5)** - `firewalld` IP セットの XML 設定ファイルが説明されています。
- `man` ページの **firewalld.service(5)** - `firewalld` service 用の XML 設定ファイルが説明されています。
- `man` ページの **firewalld.zone(5)** - `firewalld` ゾーン設定の XML 設定ファイルが説明されています。
- `man` ページの **firewalld.direct(5)** - `firewalld` ダイレクトインターフェースの設定ファイルが説明されています。

- man ページの **firewalld.lockdown-whitelist(5)** - **firewalld** ロックダウンホワイトリストの設定ファイルが説明されています。
- man ページの **firewalld.richlanguage(5)** - **firewalld** リッチ言語のルール構文が説明されていません。
- man ページの **firewalld.zones(5)** - ゾーンの全般的な説明と設定方法が説明されています。
- man ページの **firewalld.dbus(5)** - **firewalld** の **D-Bus** インターフェースが説明されています。

オンラインドキュメント

- <http://www.firewalld.org/> - **firewalld** ホームページ

第6章 NFTABLES の使用

管理者は、**nftables** フレームワークを使用して、Linux カーネルのファイアウォールで使用されるパケットフィルタリングルールを設定できます。

6.1. NFTABLES の概要

nftables フレームワークは、パケットの分類機能を提供し、**iptables** ツール、**ip6tables** ツール、**arptables** ツール、および **ebtables** ツールの後継となります。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 (**nfttrace**) 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

iptables と同様、**nftables** は、チェーンを保存するテーブルを使用します。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** ツールは、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。**libnftnl** ライブラリーは、**libmnl** ライブラリーの Netlink API の **nftables** で、低レベルの対話のために使用できます。

nftables ルールセットに対するモジュールの効果は、**nft list rule set** コマンドを使用して確認できます。これらのツールは、テーブル、チェーン、ルール、セットなどのオブジェクトを **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、先に別の従来のコマンドを使用してインストールしたルールセットに影響を及ぼす可能性があることに注意してください。

関連情報

- man ページの **nft(8)** は、**nft** コマンドラインツールで **nftables** を使用して、パケットのフィルタリングを設定および検査するための包括的な参考資料を提供します。

6.2. 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** - **firewalld** ユーティリティーを使用して、ワークステーションにファイアウォールを設定します。このユーティリティーは使いやすく、この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、複雑なファイアウォールを設定します。
- **iptables** - **iptables** ユーティリティーは、Red Hat Enterprise Linux 8 では非推奨になりました。代わりに **nftables** を使用します。



重要

異なるファイアウォールサービスが相互に影響することを回避するには、RHEL ホストでそのうちの1つだけを実行し、他のサービスを無効にします。

6.3. IPTABLES のルールを NFTABLES ルールに変換

Red Hat Enterprise Linux 8 は、既存の **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** ツールおよび **ip6tables-translate** ツールが追加されました。

拡張機能によっては変換機能がない場合もあります。対応する機能がない拡張機能が存在する場合は、ツールにより、その前に # 記号が付いた未変換ルールが出力されます。以下に例を示します。

```
# iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

また、ユーザーは、**iptables-restore-translate** ツールおよび **ip6tables-restore-translate** ツールを使用して、ルールのダンプを変換できます。その前に、**iptables-save** コマンドまたは **ip6tables-save** コマンドを使用して、現在のルールのダンプを出力できます。以下に例を示します。

```
# iptables-save >/tmp/iptables.dump
# iptables-restore-translate -f /tmp/iptables.dump

# Translated by iptables-restore-translate v1.8.0 on Wed Oct 17 17:00:13 2018
add table ip nat
...
```

詳細と、設定可能なオプションおよび値の一覧は、**iptables-translate --help** コマンドで確認できません。

6.4. NFTABLES スクリプトの作成および実行

nftables フレームワークは、シェルスクリプトを使用してファイアウォールルールを維持するための主な利点を提供するネイティブのスクリプト環境を提供します。スクリプトの実行はアトミックです。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、管理者は、**nftables** スクリプト環境で以下を行うことができます。

- 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

本セクションでは、この機能を使用する方法と、**nftables** スクリプトの作成方法と実行方法を説明します。

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。これらのファイルを拡張するか、またはスクリプトを作成できます。

6.4.1. nftables スクリプトで必要なスクリプトヘッダー

他のスクリプトと同様に、**nftables** スクリプトでは、`interpreter` ディレクティブを設定するスクリプトの最初の行に `shebang` のシーケンスが必要になります。

nftables スクリプトは、常に次の行で起動する必要があります。

```
#!/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。

6.4.2. 対応している nftables スクリプトの形式

nftables スクリプト環境は、次の形式でスクリプトに対応します。

- **nft list ruleset** コマンドが、ルールセットを表示するのと同じ形式でスクリプトを作成できません。

```
#!/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を使用できます。

```
#!/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
```

6.4.3. nftables スクリプトの実行

nftables スクリプトを実行するには、スクリプトを実行ファイルにする必要があります。スクリプトが別のスクリプトに含まれている場合に限り、スクリプトが実行ファイルである必要がありません。この手順では、スクリプトを実行ファイルにし、スクリプトを実行する方法を説明します。

前提条件

- このセクションの手順では、**nftables** スクリプトを `/etc/nftables/example_firewall.nft` ファイルに保存していることを前提としています。

手順

- 以下の手順は、一度だけ必要です。
 - 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.nft
```

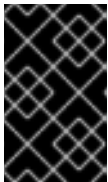
- 所有者のスクリプトを実行ファイルに変更します。

```
# chmod u+x /etc/nftables/example_firewall.nft
```

- スクリプトを実行します。

```
# /etc/nftables/example_firewall.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

- ファイルの所有者の設定は、man ページの **chown(1)** を参照してください。
- ファイルのパーミッション設定の詳細は、man ページの **chmod(1)** を参照してください。
- [「システムの起動時に nftables ルールの自動読み込み」](#)

6.4.4. nftables スクリプトでコメントの使用

nftables スクリプト環境は、**#** 文字の右側にあるすべてをコメントとして解釈します。

例6.1 nftables スクリプトのコメント

コメントは、コマンドの横だけでなく、行の先頭からも開始できます。

```
...
# Flush the rule set
flush ruleset
```

```
add table inet example_table # Create a table
...
```

6.4.5. nftables スクリプトで変数の使用

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
...
```

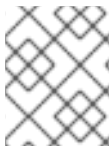
匿名セットを含む変数

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つことに注意してください。

関連情報

- セットの詳細は、[「nftables コマンドを使用したセットの使用」](#) を参照してください。
- 決定マップの詳細は、[「nftables コマンドにおける決定マップの使用」](#) を参照してください。

6.4.6. nftables スクリプトへのファイルの追加

nftables スクリプト環境を使用すると、管理者は **include** ステートメントを使用して他のスクリプトを追加できます。

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6.2 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6.3 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

- 詳細は、`man` ページの `nft(8)` の **Include files** セクションを参照してください。

6.4.7. 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。本セクションでは、システムの起動時にファイアウォールルールを読み込む方法を説明します。

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` パッケージをインストールしたときに `/etc/nftables/` に作成された *.nft スクリプトを拡張する場合は、そのスクリプトの **include** ステートメントのコメントを解除します。
 - スクリプトを新規に作成する場合は、そのスクリプトを含む **include** ステートメントを追加します。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. `nftables` サービスを有効にします。

```
# systemctl enable nftables
```

3. 必要に応じて、`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
```

関連情報

- 「対応している nftables スクリプトの形式」

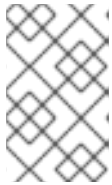
6.5. NFTABLES ルールセットの表示

nftables のルールセットには、テーブル、チェーン、およびルールが含まれます。本セクションでは、ルールセットを表示する方法を説明します。

手順

1. すべてのルールセットを表示するには、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 0; policy accept;
    tcp dport http accept
    tcp dport ssh accept
  }
}
```



注記

デフォルトで、**nftables** は事前作成のテーブルではありません。したがって、テーブルがないホストに設定したルールを表示すると、**nft list ruleset** コマンドが出力を表示しません。

6.6. NFTABLES テーブルの作成

nftables の表は、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。本セクションでは、テーブルの作成方法を説明します。

各テーブルには、アドレスファミリーが定義されている必要があります。テーブルのアドレスファミリーは、テーブルプロセスのアドレス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と一致します。
- **netdev** ingress からのパケットに一致します。

手順

1. **nft add table** コマンドを使用してテーブルを新規作成します。たとえば、IPv4 パケットおよび IPv6 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 inet example_table
```

- 必要に応じて、ルールセットのテーブルを一覧表示します。

```
# nft list tables
table inet example_table
```

関連情報

- アドレスファミリーの詳細は、man ページの **nft(8)** の **Address families** セクションを参照してください。
- テーブルで実行可能なその他のアクションの詳細は、man ページの **nft(8)** の **Tables** セクションを参照してください。

6.7. NFTABLES チェーンの作成

チェーンは、ルールのコンテナです。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- ジャンプ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。

この手順では、既存のテーブルにベースチェーンを追加する方法を説明します。

前提条件

- 新しいチェーンを追加するテーブルが存在する。

手順

- nft add chain** コマンドを使用してチェーンを新規作成します。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 inet example_table example_chain { type filter hook input priority 0 \; policy accept \; }
```



重要

シェルで、セミコロンがコマンドの最後として解釈されないようにするには、セミコロンをバックスラッシュでエスケープする必要があります。

このチェーンは、着信パケットをフィルタリングします。**priority** パラメーターは、**nftables** プロセスが同じフック値を持つチェーンを処理する順序を指定します。優先度の値が低いほど優先されます。**policy** パラメーターは、このチェーン内のルールのデフォルトアクションを設定します。サーバーにリモートでログインし、デフォルトのポリシーを **drop** に設定すると、他のルールでリモートアクセスが許可されていないと、すぐに切断されることに注意してください。

- 必要に応じて、すべてのチェーンを表示します。

```
# nft list chains
table inet example_table {
  chain example_chain {
    type filter hook input priority 0; policy accept;
  }
}
```

関連情報

- アドレスファミリーの詳細は、man ページの **nft(8)** の **Address families** セクションを参照してください。
- チェーンで実行できるその他のアクションの詳細は、man ページの **nft(8)** の **Chains** セクションを参照してください。

6.8. NFTABLES チェーンへのルールの追加

本セクションでは、既存の **nftables** チェーンにルールを追加する方法を説明します。デフォルトでは、**nftables add rule** 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。

代わりにチェーンの先頭にルールを追加する場合は、[「nftables チェーンへのルールの追加」](#)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. 新規ルールを追加するには、**nft add rule** コマンドを使用します。たとえば、**example_table** の **example_chain** に、ポート 22 の TCP トラフィックを許可するルールを追加するには、次のコマンドを実行します。

```
# nft add rule inet example_table example_chain tcp dport 22 accept
```

ポート番号の代わりに、サービス名を指定することもできます。この例では、ポート番号 **22** の代わりに **ssh** を使用できます。サービス名は、**/etc/services** ファイルのエントリーに基づいてポート番号に解決されることに注意してください。

2. 必要に応じて、**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 0; policy accept;
    ...
    tcp dport ssh accept
  }
}
```

関連情報

- アドレスファミリーの詳細は、man ページの **nft(8)** の **Address families** セクションを参照してください。
- ルールを実行できるその他のアクションの詳細は、man ページの **nft(8)** の **Rules** セクションを参照してください。

6.9. NFTABLES チェーンへのルールの追加

本セクションでは、**nftables insert rule** コマンドを使用して、既存の **nftables** チェーンの前頭にルールを追加する方法を説明します。チェーンの最後にルールを追加する場合は、[「nftables チェーンへのルールの追加」](#)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. 新しいルールを追加するには、**nft insert rule** コマンドを使用します。たとえば、ポート 22 で TCP 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 insert rule inet example_table example_chain tcp dport 22 accept
```

代わりに、ポート番号の代わりにサービスの名前を指定することもできます。この例では、ポート番号 **22** の代わりに **ssh** を使用できます。サービス名は、**/etc/services** ファイルのエントリーに基づいてポート番号に解決されることに注意してください。

2. 必要に応じて、**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 0; policy accept;
    tcp dport ssh accept
  }
  ...
}
```

関連情報

- アドレスファミリーの詳細は、man ページの **nft(8)** の **Address families** セクションを参照してください。
- ルールを実行できるその他のアクションの詳細は、man ページの **nft(8)** の **Rules** セクションを参照してください。

6.10. 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)

6.10.1. 異なる NAT タイプ- マスカレード、ソース NAT、および宛先 NAT

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、インターネットプロバイダーは予約 IP 範囲 (10.0.0.0/8 など) をルーティングしません。ネットワークで予約済みの 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。

マスカレードおよび SNAT の両方は非常に似ています。相違点は次のとおりです。

- マスカレードは、出力インターフェースの IP アドレスを自動的に使用します。したがって、出力インターフェースが動的 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、出力インターフェースの 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。出力インターフェースが固定 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、着信トラフィックを別のホストにルーティングします。たとえば、Web サーバーが予約済み 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。

6.10.2. 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、インターフェースを介して送信されるパケットのソース IP を、インターフェースの IP アドレスに動的に変更できます。これは、インターフェースに新しい 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。

次の手順では、**ens3** インターフェイスを介して、ホストから **ens3** の IP セットに送信されるパケットのソース IP を置き換える方法を説明します。

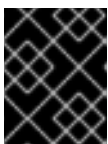
手順

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

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。

nft コマンドに `-- オプション` を渡すと、シェルが優先度の負の値を **nft** コマンドのオプションとして解釈する必要がなくなります。

3. **postrouting** チェーンに、**ens3** インターフェー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
```

6.10.3. 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、インターフェースを介して特定の IP アドレスに送信するパケットの IP を変更できます。

次の手順では、**ens3** インターフェイスを介して、ルーターから **192.0.2.1** に送信されるパケットのソース IP を置き換える方法を説明します。

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。

nft コマンドに `-- オプション` を渡すと、シェルが優先度の負の値を **nft** コマンドのオプションとして解釈する必要がなくなります。

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」](#)

6.10.4. nftables を使用した宛先 NAT の設定

宛先 NAT により、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。

以下の手順では、ルーターの **80** ポートおよび **443** ポートに送信される着信トラフィックを、IP アドレス **192.0.2.1** のホストにリダイレクトする方法を説明します。

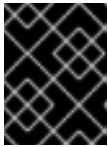
手順

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。

nft コマンドに `--` オプションを渡すと、シェルが優先度の負の値を **nft** コマンドのオプションとして解釈する必要がなくなります。

3. **prerouting** チェーンに、**80** ポートおよび **443** ポートに送信された **ens3** インターフェースの着信トラフィックを、IP アドレスが **192.0.2.1** であるホスト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、ソースアドレスを変更します。
 - a. **ens3** インターフェースが動的 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** インターフェースが静的 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
```

関連情報

- [「異なる NAT タイプ- マスカレード、ソース NAT、および宛先 NAT」](#)

6.11. NFTABLES コマンドを使用したセットの使用

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、インターフェース、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。

6.11.1. 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匿名セットを使用することもできます。

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](#)」に従って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. 必要に応じて、`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 0; policy accept;
    tcp dport { ssh, http, https } accept
  }
}
```

6.11.2. 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一覧または範囲です。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、任意の正の 32 ビットの整数値 (0 ~ 2147483647) にできます。

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

シェルで、セミコロンがコマンドの最後として解釈されないようにするには、セミコロンをバックスラッシュでエスケープする必要があります。

2. 必要に応じて、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。

6.11.3. 関連情報

- セットの詳細は、man ページの **nft(8)** の **Sets** セクションを参照してください。

6.12. 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。

6.12.1. nftables でリテラルマップの使用

リテラルマップは、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、複数のカンマ区切りマッピングを含めることができます。

リテラルマップの欠点は、マップを変更する際に、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 で変更可能な決定マップの使用](#)」にあるように、名前付き決定マップを使用します。

この例では、literal 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 パケットの両方を別のチェーンに転送し、着信のTCP パケットとUDP パケットを個別にカウントする方法を説明します。

手順

1. **example_table** 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. リテラル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 }
```

リテラル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 0; 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。

6.12.2. nftables で変更可能な決定マップの使用

nftables フレームワークは、変更可能な決定マップをサポートします。テーブルの複数のルールでこのマップを使用できます。リテラルマップに対するもう1つの利点は、変更可能なマップを使用するルールを置き換えることなく、そのマップを更新できることです。

変更可能な決定マップを作成するときは、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。

この例では、送信元の IP アドレスに基づいて着信パケットを許可または破棄する方法を説明します。変更可能な決定マップを使用すると、このシナリオを構成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。この手順では、マップからエントリーを追加および削除する方法を説明します。

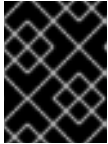
手順

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

シェルで、セミコロンがコマンドの最後として解釈されないようにするには、セミコロンをバックスラッシュで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. 必要に応じて、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. 必要に応じて、マップからエンタリーを削除します。

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. 必要に応じて、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 0; policy accept;
    ip saddr vmap @example_map
  }
}
```

6.12.3. 関連情報

- 決定マップの詳細は、man ページの **nft(8)** の **Maps** セクションを参照してください。

6.13. 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。

6.13.1. 着信パケットの別のローカルポートへの転送

本セクションでは、**8022** ポートの着信 IPv4 パケットを、ローカルシステムの **22** ポートに転送する例を説明します。

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

nft コマンドに **--** オプションを渡して、シェルが優先度の負の値を **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
```

6.13.2. 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。

この手順では、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する方法を説明します。

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

nft コマンドに `--` オプションを渡して、シェルが優先度の負の値を **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 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
```

6.14. 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できます。この手順では、受信接続を制限する方法の基本的な例を説明します。

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
```

2. 必要に応じて、前の手順で作成した meter を表示します。

```
# nft list meter ip example_table example_meter
table ip 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、現在の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 アドレスの一覧を表示します。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。

6.15. 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

nftables フレームワークにより、管理者はセットを動的に更新できます。このセクションでは、この機能を使用して、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する方法を説明します。**nftables** は、5 分経つと自動的にその IP アドレスをブラックリストから削除します。

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. **filter** テーブルに **blacklist** という名前のセットを追加します。

```
# nft add set ip filter blacklist { type ipv4_addr \; flags dynamic, timeout \; timeout 5m \; }
```

このコマンドは、IPv4 アドレスの動的セットを作成します。**timeout 5m** パラメーターは、**nftables** がセットから 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。

4. 1 分以内に 11 個以上の新しい TCP 接続を確立しようとするホストのソース IP アドレスを **blacklist** セットに自動的に追加するルールを追加します。

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked limit rate over 10/minute
add @blacklist { ip saddr }
```

5. **blacklist** セットの IP アドレスからの接続をすべて破棄するルールを追加します。

```
# nft add rule ip filter input ip saddr @blacklist drop
```

関連情報

- [「nftables で名前付きセットの使用」](#)

6.16. 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。本セクションでは、3 つのオプションを説明します。

6.16.1. 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。本セクションでは、カウンターを使用して新しいルールを作成する方法を説明します。

既存のルールにカウンターを追加する手順は、[「既存のルールへのカウンターの追加」](#)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 0; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

6.16.2. 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。本セクションでは、既存のルールにカウンターを追加する方法を説明します。

カウンターで新しいルールを追加する手順は、[「カウンターによるルールの作成」](#)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 0; 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 0; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

6.16.3. 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。この手順では、ルールのトレースと、このルールに一致するパケットの監視を有効にする方法を説明します。

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 0; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールの代わりに **meta nfttrace set 1** パラメーターを使用して、トレース機能を追加します。以下の例は、前の手順で表示したルールの代わりに、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 nfttrace 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 nfttrace 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。

6.17. NFTABLES ルールセットのバックアップおよび復元

本セクションでは、**nftables** ルールのバックアップをファイルに作成し、ファイルからのルールを復元する方法を説明します。

管理者は、ルールを持つ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。

6.17.1. ファイルへの nftables ルールセットのバックアップ

本セクションでは、**nftables** ルールセットをファイルにバックアップする方法を説明します。

手順

1. **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** 形式の場合は、以下ようになります。

```
# nft list ruleset > file.nft
```

- JSON 形式の場合は、以下ようになります。

```
# nft -j list ruleset > file.json
```

6.17.2. ファイルからの nftables ルールセットの復元

本セクションでは、**nftables** ルールセットを復元する方法を説明します。

手順

1. **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** 形式であるか、**nft** コマンドが含まれている場合は、次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
```

6.18. 関連情報

- ブログ投稿の「[Using nftables in Red Hat Enterprise Linux 8](#)」には、**nftables** 機能の使用に関する概要が記載されています。
- [What comes after iptables? Its successor, of course: nftables](#) では、**nftables** が **iptables** の代替になった理由が説明されています。
- 「[Firewalld: The Future is nftables](#)」の記事では、**firewalld** でデフォルトのバックエンドとなる **nftables** に関する追加情報が提供されます。