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を設定するための
ガイド

Last Updated: 2019-05-07

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を設定するためのガイド

法律上の通知

概要

本書は、管理者が、さまざま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

ドキュメントの改善に関するご意見やご要望をお聞かせください。

- 特定の文章に簡単なコメントを記入する場合は、ドキュメントが Multi-page HTML 形式になっているのを確認してください。コメントを追加する部分を強調表示し、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、表示された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 OPENSSH を使用した安全な通信の使用

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間でのセキュアな通信を容易にし、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。

ssh プログラムは、**telnet** や **rsh** などのリモートホストへのログインに使用される、旧式で、セキュリティー保護が十分でないターミナルアプリケーションに代わるものとして設計されています。また、**scp** と呼ばれる関連プログラムが、ホスト間でファイルをコピーするために設計された **rcp** などの旧式プログラムの代わりとなります。このような旧式アプリケーションは、クライアントとサーバーとの間で送信するパスワードを暗号化しないため、可能な限り使用しないようにしてください。リモートシステムへのログインにセキュアな方法を使用することで、クライアントシステムとリモートホストの両方に対するリスクが低減されます。

Red Hat Enterprise Linux は、一般的な **OpenSSH** パッケージ (**openssh**) と、**OpenSSH** のサーバー (**openssh-server**) およびクライアント (**openssh-clients**) の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インストールし、暗号化通信を提供する **OpenSSH** を有効にします。

1.1. SSH プロトコル

1.1.1. SSH を使用する理由

潜在的な侵入者は、ネットワークトラフィックの中断、傍受、経路変更を可能にする様々なツールを自由に駆使して、システムに侵入します。一般的には、これらの脅威は以下のとおり分類できます。

2 システム間の通信の傍受

攻撃者は、ネットワーク上で通信を行う二者の間のどこかに潜み、両者間で渡される情報をコピーしている可能性があります。攻撃者は情報を傍受して保持する、または情報を改ざんして元の受信者に送信する場合があります。
このような攻撃は、通常 **パケットスニファー** を使用して行われます。パケットスニファーは、ネットワークを通過するパケットをキャプチャーしてその内容を分析する、ごく一般的なネットワークユーティリティーです。

特定のホストの偽装

攻撃者のシステムは、送信の対象となる受信者を装うように設定されます。これに成功すると、ユーザーのシステムは不正なホストと通信していることに気がつかないままとなります。
この攻撃は、**DNS ポイズニング** として知られる手法か **IP スプーフィング** と呼ばれる手法を用いて実行されます。前者の場合、侵入者はクラックされた DNS サーバーを使用して、クライアントシステムを不当に複製されたホストへ指定します。後者の場合、侵入者は信頼されたホストから送信されたように見せかけた偽装ネットワークパケットを送信します。

いずれの手法でも、潜在的な機密情報を傍受することが可能です。その傍受が悪意のある理由で行われる場合には、悲惨な結果をもたらしかねません。リモートシェルログインとファイルコピー用に **SSH** を使用すると、こうしたセキュリティー脅威を大幅に軽減することができます。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全通信は暗号化されます。各パケットはローカルシステムとリモートシステムのみに知られている鍵を使用して暗号化されるため、通信のいずれか一方の ID をスプーフィングする試みは成功しません。

1.1.2. 主な特長

SSH プロトコルは以下のような保護手段を提供します。

対象のサーバーになりすますことができない

初回接続後に、クライアントは以前に接続したサーバーと同じサーバーに接続していることを確認できます。

認証情報の取得はできない

クライアントは、強力な 128 ビット暗号化を使用してサーバーへ認証情報を送信します。

通信の傍受はできない

セッション中に送受信された全データは、128 ビット暗号化を使用して転送されるため、傍受された送信データの暗号解読と読み取りは非常に困難になります。

さらに、SSH プロトコルは以下のようなオプションも提供します。

ネットワーク上でグラフィカルアプリケーションを使用するセキュアな手段を提供する

クライアントは、**X11 転送** と呼ばれる手法を使用して、サーバーから **X11** (**X Window System**) アプリケーションを転送できます。

セキュアでないプロトコルをセキュアにする手段を提供する

SSH プロトコルは、送受信するものをすべて暗号化します。**SSH** サーバーは、**ポート転送** と呼ばれる技術を使用して、**POP** のようなセキュアではないプロトコルをセキュアにし、システムとデータ全体のセキュリティーを強化できます。

セキュアなチャンネルを作成する

OpenSSH サーバーとクライアントは、サーバーマシンとクライアントマシン間のトラフィックに対して、仮想プライベートネットワークに似たトンネルを作成するよう設定できます。

Kerberos 認証をサポートする

OpenSSH サーバーとクライアントは、**Kerberos** ネットワーク認証プロトコルの **GSSAPI** (Generic Security Services Application Program Interface: 汎用セキュリティーサービス API) 実装を使用して認証を行うよう設定できます。

1.1.3. プロトコルのバージョン

現在、**SSH** には、バージョン 1 と、新しいバージョンのバージョン 2 があります。Red Hat Enterprise Linux 8 の **OpenSSH** スイートでは、**SSH** バージョン 2 を使用します。バージョン 2 は、バージョン 1 で既知の不正使用に対し、脆弱性のない拡張された鍵交換アルゴリズムを使用します。Red Hat Enterprise Linux 8 では、**OpenSSH** スイートはバージョン 1 の接続には対応していません。

1.1.4. SSH 接続のイベントシーケンス

以下に挙げる一連のイベントは、2 つのホスト間で行われる **SSH** 通信の整合性を保護するのに役立ちます。

1. 暗号化ハンドシェイクが行われ、クライアントが正しいサーバーと通信していることを確認できます。
2. クライアントとリモートホストとの間の接続のトランスポート層が、対称暗号方式を使用して暗号化されます。
3. クライアントが、サーバーに対して自己認証します。
4. クライアントは、暗号化された接続でリモートホストと対話します。

1.1.4.1. トランスポート層

トランスポート層の主な役割は、認証時およびその後の通信中に、2 つのホスト間の通信を簡単に安全でセキュアなものにすることです。トランスポート層は、データの暗号化と復号を処理し、データパケットの送受信時にその整合性を保護することでその役割を果たします。また、トランスポート層は、情報を圧縮して転送を高速にします。

SSH クライアントがサーバーに接触すると鍵情報が交換されるため、両システムはトランスポート層を適正に構築できます。以下は、こうした鍵情報の交換中に発生するステップです。

- 鍵を交換する
- 公開鍵暗号化アルゴリズムが決定する
- 対称暗号化アルゴリズムが決定する
- メッセージ認証アルゴリズムが決定する
- ハッシュアルゴリズムが決定する

鍵交換の間、サーバーは一意の **ホスト鍵** を用いて、クライアントに対して自己識別を行います。クライアントがこの特定のサーバーと過去に通信したことがなければ、クライアントはサーバーのホスト鍵を知らないため、接続は成立しません。**OpenSSH** はこの問題に対処するためにサーバーのホスト鍵を承認します。これは、ユーザーが通知を受けて新規のホスト鍵を受け取り、検証した後に行われます。それ以降の接続では、サーバーのホスト鍵が、クライアント上に保存されているバージョンと照合され、クライアントが実際に目的のサーバーと通信していることを確信できます。この後、ホスト鍵が一致しなくなった場合は、接続前にクライアントに保存してあるバージョンをユーザーが削除する必要があります。

警告

ローカルシステムは、対象サーバーと、攻撃者が設定した偽サーバーとの違いを認識しないため、攻撃者は初回コンタクト時に **SSH** サーバーをマスカレードすることが可能です。この問題を防ぐため、初回接続の前や、ホスト鍵の不一致が発生した場合に、サーバー管理者へ連絡して新しい **SSH** サーバーの整合性を確認してください。

SSH は、ほとんどすべての公開鍵アルゴリズムまたはエンコード形式に対応するように設計されています。初回の鍵交換で、交換に使用されるハッシュ値と共有秘密値が作成されると、2 つのシステムは新しい鍵とアルゴリズムの計算を直ちに開始して、認証と、今後の接続で送信されるデータを保護します。

所定の鍵とアルゴリズムを使用して一定量のデータ (正確な量は **SSH** 実装により異なる) が送信された後にもう 1 度鍵交換が行われ、ハッシュ値と新しい共有秘密値のセットが新たに生成されます。攻撃者がハッシュ値と共有秘密値を判別できたとしても、その情報が役に立つのは限られた時間のみです。

1.1.4.2. 認証

トランスポート層が、2 つのシステム間で情報を渡すためのセキュアなトンネルを構築すると、サーバーは、秘密鍵でエンコードされた署名の使用やパスワードの入力など、サポートされている別の認証方法をクライアントに伝えます。次に、クライアントはサポートされているいずれかの方法を使用して

サーバーに対して自己認証を試みます。

SSH サーバーとクライアントは、異なるタイプの認証を採用するように設定できるため、双方の制御が最適化されます。サーバーはそのセキュリティーモデルに基づき、サポートする暗号化方法を決定でき、クライアントは利用可能なオプションの中から試行する認証方法の順番を選択できます。

1.1.4.3. チャンネル

SSH トランスポート層での認証に成功すると、**多重化**[1] と呼ばれる手法により複数のチャンネルが開きます。これらの各チャンネルは、異なるターミナルセッションと、転送された X11 セッションの通信を処理します。

クライアントとサーバーの両方で、新しいチャンネルを作成できます。その後、各チャンネルに別々の番号が接続の両端に割り当てられます。クライアントが新しいチャンネルを開こうとする際、要求と共にチャンネル番号を送信します。この情報はサーバーにより保存され、そのチャンネルに通信を移動するのに使用されます。これは、異なるタイプのセッションが相互に影響しないように、あるセッションの終了時に、そのチャンネルが **SSH** による一次接続を停止せずに閉じることができるようにするためです。

また、チャンネルは **フロー制御** もサポートしているため、規則的な方法でデータを送受信できます。この方法では、チャンネルが開いているというメッセージをクライアントが受信するまで、チャンネルでデータが送信されません。

クライアントが要求するサービスのタイプと、ユーザーがネットワークに接続される方法に応じて、クライアントとサーバーは、各チャンネルの特性を自動的にネゴシエートします。これにより、プロトコルの基本インフラストラクチャーを変更しなくても、異なるタイプのリモート接続を非常に柔軟に処理できます。

1.2. OPENSSH の設定

1.2.1. 設定ファイルを使用した OpenSSH の設定

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。

表1.1 システム全体の設定ファイル

ファイル	説明
/etc/ssh/moduli	セキュアなトランスポート層を構築するのに非常に重要となる、Diffie-Hellman 鍵交換に使用される Diffie-Hellman グループが置かれています。SSH セッションの初めに鍵が交換される時、共有秘密値が作成されますが、どちらか一方の当事者だけでは決定できません。この値は、ホスト認証を行うのに使用されます。
/etc/ssh/ssh_config	デフォルトの SSH クライアント設定ファイルです。**~/.ssh/config** が存在する場合は、これにより上書きされる点に注意して下さい。

ファイル	説明
/etc/ssh/sshd_config	**sshd** デーモンの設定ファイルです。
/etc/ssh/ssh_host_ecdsa_key	**sshd** デーモンが使用する ECDSA 秘密鍵です。
/etc/ssh/ssh_host_ecdsa_key.pub	**sshd** デーモンが使用する ECDSA 公開鍵です。
/etc/ssh/ssh_host_rsa_key	**sshd** デーモンにより使用される SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
/etc/ssh/ssh_host_rsa_key.pub	**sshd** デーモンにより使用される SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
/etc/pam.d/sshd	**sshd** デーモンの PAM 設定ファイルです。
/etc/sysconfig/sshd	**sshd** サービスの設定ファイルです。

表1.2 ユーザー固有の設定ファイル

ファイル	説明
~/.ssh/authorized_keys	サーバー用の認証済み公開鍵の一覧があります。クライアントがサーバーに接続すると、サーバーが、このファイル内に格納されている署名済み公開鍵を確認してクライアントを認証します。
~/.ssh/id_ecdsa	ユーザーの ECDSA 秘密鍵を格納します。
~/.ssh/id_ecdsa.pub	ユーザーの ECDSA 公開鍵です。
~/.ssh/id_rsa	**ssh** が使用する SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
~/.ssh/id_rsa.pub	**ssh** が使用する SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
~/.ssh/known_hosts	ユーザーがアクセスする SSH サーバーのホスト鍵を格納します。このファイルは、SSH クライアントが正しい SSH サーバーに接続していることを確認するのに使用するため、非常に重要です。

警告

SSH サーバーを設定する場合は、**/etc/ssh/sshd_config** ファイルの **UsePrivilegeSeparation no** ディレクティブで **Privilege Separation** 機能をオフにしないでください。**Privilege Separation** をオフにすると、多くのセキュリティー機能が無効になるため、サーバーがセキュリティー上の潜在的な脆弱性にさらされ、攻撃対象となります。**UsePrivilegeSeparation** の詳細は、**sshd_config**(5) の man ページまたは Red Hat ナレッジベースの記事「What is the significance of UsePrivilegeSeparation directive in /etc/ssh/sshd_config file and how to test it ?」を参照してください。

SSH 設定ファイルに使用できる各種ディレクティブの情報は、man ページの **ssh_config**(5) および **sshd_config**(5) を参照してください。

1.2.2. OpenSSH サーバーの起動

OpenSSH サーバーを実行するには、**openssh-server** パッケージをインストールします。現行セッションで **sshd** デーモンを起動するには、以下を行います。

```
# systemctl start sshd.service
```

現行セッションで **sshd** デーモンを設定するには、以下を行います。

```
# systemctl stop sshd.service
```

システムの起動時にデーモンを自動的に起動するには、以下を行います。

```
# systemctl enable sshd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/sshd.service to
/usr/lib/systemd/system/sshd.service.
```

sshd デーモンは **network.target** ターゲットユニットに依存しますが、静的設定のネットワークインターフェースやデフォルトの **ListenAddress 0.0.0.0** オプションの場合はこれで十分です。**ListenAddress** ディレクティブで別のアドレスを指定し、より遅い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、**/etc/systemd/system/sshd.service.d/local.conf** ファイルを以下のオプションで作成します。

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

この後、以下のコマンドで **systemd** マネージャー設定をリロードします。

```
# systemctl daemon-reload
```

システムを再インストールすると、新しい識別鍵のセットが作成される点に注意してください。そのため、再インストールの前にいずれかの **OpenSSH** ツールを使用してシステムに接続したことがあるクライアントには、以下のようなメッセージが表示されます。

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
```

これを防ぐには、**/etc/ssh/** ディレクトリーから関連ファイルをバックアップしておきます。ファイルの一覧は 表1.1「システム全体の設定ファイル」 を参照してください。これで、システムの再インストール時にファイルを復元できます。

1.2.2.1. OpenSSH サーバーの鍵生成の設定

OpenSSH は、サーバー鍵の RSA、ECDSA、および ED25519 がない場合に自動的に作成します。RHEL 8 より前のバージョンでは、必要に応じて管理者が **/etc/sysconfig/sshd** ファイルに自動ホスト鍵を設定していました。RHEL 8 でホストキーの作成を設定するには、インスタンス化した **sshd-keygen@.service** サービスを使用します。

たとえば、次のコマンドは、OpenSSH サーバーにおける RSA 鍵の自動作成を無効にし、DSA 鍵の自動作成を有効にします。

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

> **警告**
>
> DSA 鍵は、対応している鍵のサイズが小さく、安全ではないため、非推奨となっています。

1.2.3. リモート接続に必要な SSH

SSH を本当の意味で有効なものにするためには、セキュリティー保護されていない接続プロトコルは使用しないことをお勧めします。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。無効にするサービスには、**telnet**、**rsh**、**rlogin**、**vsftpd** などがあります。

1.3. 鍵ベース認証の使用

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。これを行うには、**vi**、**nano** などのテキストエディターで **/etc/ssh/sshd_config** 設定ファイル開き、**PasswordAuthentication** を以下のように変更します。

```
PasswordAuthentication no
```

新規のデフォルトインストール以外のシステムで作業をしている場合は、**PubkeyAuthentication no** が設定されて **いない** ことを確認してください。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プロセス内で鍵ベースのログをテストする

ことが推奨されます。

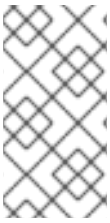
ssh、**scp**、または **sftp** を使用してクライアントマシンからサーバーに接続できるようにするには、以下のステップに従って認証鍵ペアを生成します。鍵はユーザーごとに別々に生成する必要がある点に注意してください。

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、最初に SELinux ブール値 **use_nfs_home_dirs** を有効にします。

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

重要

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



注記

システムを再インストールする際に、以前生成した鍵ペアを維持する場合は、**~/.ssh/** ディレクトリーをバックアップします。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。この手順は、システムの全ユーザー (**root** を含む) が実行できます。

1.3.1. 鍵ペアの生成

以下の手順に従って、SSH プロトコルのバージョン 2 用の RSA 鍵ペアを生成します。

1. RSA 鍵ペアを生成するには、シェルプロンプトで以下を入力します。

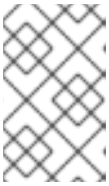
```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2. **Enter** キーを押して、新規作成される鍵のデフォルトの場所 (**~/.ssh/id_rsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントのログイン時に使用するパスワードは使用しないでください。
この後、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/USER/.ssh/id_rsa.
Your public key has been saved in /home/USER/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:UNIgIT4wfhdQH/K7yqmjsbZnnyGDKiDviv492U5z78Y
USER@penguin.example.com
The key's randomart image is:
+---[RSA 2048]----+
|o ..==o+.        |
|.+ . .=oo        |
| .o. ..o         |
|  ...  ..        |
|       .S        |
|o .     .        |
|o+ o .o+ ..      |
```

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



注記

以前のバージョンでデフォルトのフィンガープリントであった MD5 鍵フィンガープリントを取得するには、**ssh-keygen** コマンドで **-E md5** オプションを使用します。

4. デフォルトで、~/**.ssh**/ ディレクトリーのパーミッションは、**rwx------** または 8 進数表記の **700** に設定されます。これは、**USER** のみがコンテンツを表示できるようにする設定です。必要に応じて、以下のコマンドで確認できます。

```
$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. 公開鍵をリモートマシンにコピーするには、次の形式でコマンドを実行します。

```
ssh-copy-id user@hostname
```

これにより、最近変更した ~/**.ssh**/**id*.pub** 公開鍵がインストールされていない場合は、その公開鍵をコピーします。または、以下のように、公開鍵のファイルを指定します。

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

これにより、~/**.ssh**/**id_rsa.pub** の内容が、接続するマシンの ~/**.ssh**/**authorized_keys** ファイルにコピーされます。ファイルが存在する場合は、鍵がその最後に追加されます。

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、以下のステップに従います。

1. ECDSA 鍵ペアを生成するには、シェルプロンプトで以下を入力します。

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

2. **Enter** キーを押して、新規作成された鍵用のデフォルトの場所 (~/**.ssh**/**id_ecdsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントのログイン時に使用するパスワードは使用しないでください。
この後、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/USER/.ssh/id_ecdsa.
Your public key has been saved in /home/USER/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:8BhZageKrLXM99z5f/AM9aPo/KAUd8ZZFPcPFWqK6+M
USER@penguin.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|      . .      +=|
```

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

4. デフォルトで、~/**.ssh**/ ディレクトリーのパーミッションは、**rwx------** または 8 進数表記の **700** に設定されます。これは、**USER** のみがコンテンツを表示できるようにする設定です。必要に応じて、以下のコマンドで確認できます。

```
$ ls -ld ~/.ssh
            $ ls -ld ~/.ssh/
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

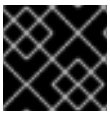
5. 公開鍵をリモートマシンにコピーするには、次の形式でコマンドを実行します。

```
ssh-copy-id USER@hostname
```

これにより、最近変更した ~/**.ssh/id*.pub** 公開鍵がインストールされていない場合は、その公開鍵をコピーします。または、以下のように、公開鍵のファイルを指定します。

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

これにより、~/**.ssh/id_ecdsa.pub** の内容が、接続するマシンの ~/**.ssh/authorized_keys** にコピーされます。ファイルが存在する場合は、鍵がその最後に追加されます。



重要

秘密鍵は、個人使用を目的としているため、他人には決して教えないでください。

1.4. 関連情報

Red Hat Enterprise Linux で **OpenSSH** サーバーを設定し、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

- **sshd**(8) - **sshd** デーモンの man ページでは、利用可能なコマンドラインオプションとサポートされる設定ファイルおよびディレクトリーの完全な一覧を提供します。
- **ssh**(1): **ssh** クライアントアプリケーション の man ページでは、利用可能なコマンドラインオプションとサポートされる設定ファイルおよびディレクトリーの完全な一覧を提供します。
- **scp**(1) - **scp** ユーティリティーの man ページでは、このユーティリティーの詳細な説明とその使用方法が説明されています。
- **sftp**(1) - **sftp** ユーティリティーの man ページ。
- **ssh-keygen**(1) - **ssh-keygen** ユーティリティーの man ページは、この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、変換する詳細な方法が説明されています。

- **ssh_config**(5) - man ページの **ssh_config** 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。
- **sshd_config**(5) - **sshd_config** のman ページは、利用可能な SSH デーモン設定オプションを詳細に説明しています。

オンラインのドキュメント

- 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を掲載した OpenSSH のホームページです。
- OpenSSL 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などの役立つリソースを掲載した OpenSSL のホームページです。

---

[1] 多重接続は、共有されている共通の媒体上で送信されるいくつかの信号で構成されます。SSH により、異なるチャンネルが共通のセキュアな接続で送信されます。

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、サポート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	使用することは推奨されません。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。

関連資料

- 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、暗号化に関する考慮事項が大幅に簡素化されています。**DEFAULT** 暗号化ポリシーでは、TLS 1.2 および 1.3 のみが許可されています。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、古いクライアントまたはサーバーとの接続を防ぎません。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をサポートしない古い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** コマンドを使用して、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、カスタム構成上好ましいプロトコル、暗号スイート、およびキー長について、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 をサポートするような特別な事情がない限り、システムが少なくとも TLS バージョン 1.2 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 に対応しているにもかかわらず、RHEL 8 コンポーネントがこのプロトコルのすべての機能をサポート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。eNULL および aNULL 暗号化スイート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-MD をベースとした暗号化スイートには深刻な欠陥があるため、可能な場合はこれも無効にしてください。いわゆるエクスポート暗号化スイートも同様です。エクスポート暗号化スイートは意図的に弱くなっているため、侵入が容易になっています。

128 ビット未満のセキュリティーしか提供しない暗号化スイートでは直ちにセキュリティーが保護されなくなるというわけではありませんが、耐用年数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 は、パディングオラクル攻撃に対して脆弱ではないため、CBC モード暗号よりも優先する必要があります。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 鍵を使用する際は常に、少なくとも SHA-256 で署名された 最低 3072 ビットの鍵の長さを優先させます。これは、真の 128 ビットのセキュリティーでは十分な大きさです。

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。

関連資料

- 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 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を提供します。

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

いずれの設定を選択しても、サーバーアプリケーションが強制的に **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、その 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。

```
# 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 およびプロキシーサーバーの設定

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 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
```

関連資料
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)** は、**/etc/ssl/openssl.conf** 設定ファイルの形式を説明します。
- 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 FIREWALLD の使用および設定

firewall 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、ホストマシンに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。

3.1. FIREWALLD の使用

3.1.1. 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、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はゾーンに割り当てることができます。許可されているトラフィックは、コンピューターが接続するネットワークと、このネットワークが割り当てられているセキュリティーレベルに従います。ファイアウォールサービスは、特定のサービスに着信トラフィックを許可するのに必要な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、デフォルトですべてのトラフィックを許可します。

関連資料
以下の資料は、**firewalld** に関する関連資料を提供します。

インストールされているドキュメント

- **firewalld(1)** の man ページ - **firewalld** のコマンドオプションが説明されています。
- **firewalld.conf(5)** の man ページ - **firewalld** を設定する情報が含まれます。
- **firewall-cmd(1)** の man ページ - **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。
- **firewall-config(1)** の man ページ - **firewall-config** ツールの設定が説明されています。
- **firewall-offline-cmd(1)** の man ページ - **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。
- **firewalld.icmptype(5)** の man ページ - **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。
- **firewalld.ipset(5)** の man ページ - **firewalld IP** セットの XML 設定ファイルが説明されています。
- **firewalld.service(5)** の man ページ - **firewalld service** 用の XML 設定ファイルが説明されています。
- **firewalld.zone(5)** の man ページ - **firewalld** ゾーン設定の XML 設定ファイルが説明されています。

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

オンラインのドキュメント

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

3.1.2. 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、インターフェースのゾーンに **firewalld** を通知します。インターフェースにゾーンを割り当てるには、NetworkManager と、firewall-config ツールまたは **firewall-cmd** コマンドラインツールを使用して割り当てられます。後者の 2 つは、適切な NetworkManager 設定ファイルの編集のみを行います。**firewall-cmd** または firewall-config を使用してインターフェースのゾーンを変更する場合、リクエストは NetworkManager に転送され、**firewalld** には処理されません。

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。システム内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用。選択され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

マスカレードを特別にルーター用に有効にした外部ネットワーク上での使用向けです。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。選択され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。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での使用。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での使用。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。選択した着信接続のみが許可されます。

これらのゾーンのいずれかを **デフォルト** 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。セキュリティー問題を回避するために、ユーザーの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。

3.1.3. 事前定義サービス

サービスは、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ロードされるファイアウォールヘルパーモジュールの一覧になります。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。

グラフィカルな firewall-config ツール、**firewall-cmd**、および **firewall-offline-cmd** を使用してサービスを追加および削除できます。

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場合は、対応する XML ファイルが **/etc/firewalld/services/** では見つかりません。**/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。

3.1.4. ランタイムおよび永続化設定

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には、**--permanent** オプションを使用します。**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** オプションを実行します。

--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には、**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、ネットワーキングトラフィックを停止します。

3.1.4.1. CLI を使用したランタイムおよび永続設定の設定の変更

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はできません。CLI では、ランタイムまたは永続モードを修正します。永続化設定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には、2 つの方法を使用できます。

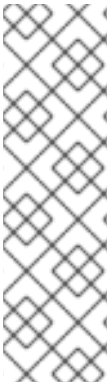
1. 以下のように、ランタイム設定を変更して、永続化します。

```
# 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、永続化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、設定を誤ると、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、以下のコマンドを実行します。

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

3.2. 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** で以下のコマンドを実行します。

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

また、GNOME では、**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** を入力します。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 **Activities Overview** を開き、**firewall** 入力して **Enter** を押します。

3.3. FIREWALLD の現在のステータスおよび設定の表示

3.3.1. firewalld の現在のステータスの表示

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

手順

1. サービスのステータスを表示するには、以下のコマンドを実行します。

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

2. サービスステータスの詳細は **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
```

関連資料
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には、「現在の firewalld 設定の表示」 を参照してください。

3.3.2. 現在の firewalld 設定の表示

3.3.2.1. GUI を使用して許可されるサービスの表示

グラフィカルの **firewall-config** ツールを使用してサービスの一覧を表示するには、**Super** キーを押して Activities Overview 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、**サービス** タブでサービスの一覧を確認できます。

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開始するには、以下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。

3.3.2.2. 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及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、以下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** コマンドに追加します。

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

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、コマンドヘルプを使用して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
... [output truncated]
```

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) を開したあと、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。

3.4. FIREWALLD の起動

手順

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

```
# systemctl enable firewalld
```

3.5. FIREWALLD の停止

手順

1. **firewalld** を停止するには、**root** で以下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、以下を行います。

```
# 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、以下を行います。

```
# systemctl mask firewalld
```

3.6. FIREWALLD を使用したネットワークトラフィックの制御

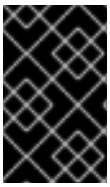
3.6.1. 緊急時に CLI を使用してすべてのトラフィックを無効

システムへの攻撃など、緊急な状態では、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、以下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、以下のコマンドを実行します。

```
# firewall-cmd --query-panic
```

3.6.2. 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。これは、必要なすべてのポートを開き、**service definition file** に従ってその他の設定を変更します。

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 ...
[output truncated]
```

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
```

3.6.3. GUI を使用した事前定義サービスでトラフィックの制御

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のチェックボックスは選択を解除してください。

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。

ポート、**プロトコル**、**ソースポート** 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、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

実行時 モードでは、サービス設定を変更できません。

3.6.4. 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 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。

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
```

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、以下のコマンドを使用します。

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

--name=service-name オプションを指定して、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** からファイルをロードします。**/etc/firewalld/services** にファイルが置かれ、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、またはサービスのデフォルトをロードするように **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。

3.6.5. 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** デーモンでは、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、セキュリティーを強化するため、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。

3.6.5.1. 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ーリスクでもあります。一般的に、ポートを閉じたままにし、特定サービスに対して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を一覧表示します。

```
# 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。

3.6.5.2. ポートを閉じる

開かれた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を一覧表示します。

```
# 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
```

3.6.6. 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。
3. 許可するポート番号またはポートの範囲を入力します。
4. リストから **tcp** または **udp** を選択します。

3.6.7. 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。

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. 右側で **Protocols** タブを選択し、**Add** ボタンをクリックします。**Protocol** ウィンドウが開きます。
3. リストからプロトコルを選択するか、**Other Protocol** チェックボックスを選択し、そのフィールドにプロトコルを入力します。

3.6.8. 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。

1. firewall-config ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側の **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。
3. 許可するポート番号またはポート範囲を入力します。リストから **tcp** または **udp** を選択します。

3.7. ファイアウォールゾーンでの作業

ゾーンは、着信トラフィックをより透過的な管理を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。

3.7.1. ゾーンの一覧

手順

1. システムで利用可能なゾーンを確認するには、以下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。

2. すべてのゾーンで詳細情報を表示する場合は、以下のコマンドを実行します。

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

3. 特定ゾーンに関する詳細情報を表示する場合は、以下のコマンドを実行します。

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

3.7.2. 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。別のゾーンにルールを設定しないといけない場合もあります。

手順

1. 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、以下のコマンドを実行します。

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

3.7.3. 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。

1. 現在のデフォルトゾーンを表示します。

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

2. 新しいデフォルトゾーンを設定します。

```
# firewall-cmd --set-default-zone zone-name
```

注記

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

3.7.4. 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。複数のインターフェースを使用して、その各インターフェースに、トラフィックを通過する特定のゾーンを設定できます。

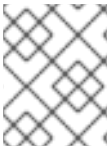
手順
特定インターフェースにゾーンを割り当てるには、以下を行います。

1. アクティブゾーン、およびそのゾーンに割り当てられているインターフェ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>
```



注記

再起動後も設定を持続させる **--permanent** オプションを使用する必要はありません。新しいデフォルトゾーンを設定すると、設定は永続化されます。

3.7.5. ネットワーク接続にデフォルトゾーンの割り当て

接続が NetworkManager により管理されると、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、さまざまなファイアウォール設定の柔軟性を提供します。したがって、ゾーンおよび設定には、会社または自宅など、さまざまな場所を指定できます。

手順

1. インターネット接続にデフォルトゾーンを設定するには、NetworkManager GUI を使用するか、**/etc/sysconfig/network-scripts/ifcfg-connection-name** ファイルを変更して、この接続にゾーンを割り当てる行を追加します。

```
ZONE=zone-name
```

3.7.6. 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、このコマンドがなければコマンドが動作しません。

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化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
```

3.7.7. 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 ファイルフォーマットで、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/** ディレクトリーです。

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。

関連資料

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

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

すべてのゾーンに対して、特に指定されていない着信トラフィックを処理するデフォルト動作を設定できます。そのような動作は、ゾーンのターゲットを設定することで定義されます。オプションは、**default**、**ACCEPT**、**REJECT**、および **DROP** の 3 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、拒否についてソースマシンに通知しますが、パケットが破棄される時に送られる情報はありません。

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>
```

3.8. 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

3.8.1. 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仕分けし、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。

3.8.2. ソースの追加

着信トラフィックを特定のソースにルートするには、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、以下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、以下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
```

3.8.3. 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を一覧表示します。

```
# 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
```

3.8.4. ソースポートの追加

発信源となるポートに基づいてトラフィックの仕分けを有効にするには、**--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、以下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>
```

3.8.5. 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仕分けを無効にします。

手順

1. ソースポートを削除するには、以下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>
```

3.8.6. 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. ソースを信頼されるゾーンに追加して、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、以下を行います。

```
# 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
```

3.8.7. 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。

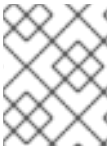
3.8.7.1. ゾーンへのプロトコルの追加

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、以下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るには、 **--add-protocol** オプションで **igmp** 値を使用します。

3.8.7.2. 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、以下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
```

3.9. 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** で以下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、有効な場合は終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。**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** で以下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** で以下のコマンドを実行します。

```
# firewall-cmd --zone=external --remove-masquerade --permanent
```

3.10. ICMP 要求の管理

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、さまざま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、システム間のデータ交換するには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。

3.10.1. ICMP 要求の一覧表示およびブロック

ICMP リクエストの一覧表示

ICMP リクエストについては、**/usr/lib/firewalld/icmptypes/** ディレクトリー内の各 XML ファイルで説明されています。このファイルを読み、リクエストの説明を確認します。**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** タイプを一覧表示するには、以下を行います。

```
# 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、以下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、以下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、通常の情報が提供されません。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、以下を行います。

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

- **ICMP** リクエストをブロックするには、以下を行います。

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

- **ICMP** リクエストのブロックを削除するには、以下を行います。

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

情報を提供せずに ICMP 要求のブロック

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見ることができます。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、以下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** 要求をブロックして破棄し、その他の要求は許可するには、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には、以下を行います。

```
# 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するため、すでにブロックしていないリクエストをすべてブロックします。ブロックされているものはブロックされません。したがって、リクエストのブロックを解除する必要がある場合は、ブロックコマンドを使用してください。

- 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、以下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
```

3.10.2. 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。

3.11. FIREWALLD を使用した IP セットの設定および制御

firewalld でサポートする IP セットタイプの一覧を表示するには、root で以下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
```

3.11.1. 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 セットを一覧表示するには、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た以下のコマンドを **root** で実行します。

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

上記のコマンドは、**IPv4** の名前 **test** とタイプ **hash:net** で新しい IP セットを作成します。**IPv6** と使用する IP セットを作成するには、**--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** を再ロードします。

- **root** で以下のコマンドを実行し、新しい IP セットを一覧表示します。

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

- IP セットの詳細は、**root** で以下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。

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

```
# 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** で以下のコマンドを実行します。

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

- IP セットの拡張エントリーの一覧を表示するには、**root** で以下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、更新したエントリー一覧を確認するには、以下のコマンドを **root** で実行します。

```
# 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、**test** IP セットをソースとして **drop** ゾーンに追加し、**test** の IP セット一覧に表示されるすべてエントリーから発信されるパケットをすべて破棄するには、**root** で以下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:** 接頭辞は、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ており、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。

3.12. 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、どのアプリケーションも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の望ましくない設定変更を確実に防ぐことができます。

3.12.1. 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を確認するには、**root** で以下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終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。

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

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

3.12.2. 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。「*」がない場合は、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、以下のコマンドを実行します。

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

  このコマンドで、実行中のアプリケーションがすべて返されます。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以下のようにパイプ処理します。

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

- ホワイトリストにあるコマンドラインを一覧表示するには、**root** で以下のコマンドを実行します。

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

- ホワイトリストにコマンド **command** を追加するには、**root** で以下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コマンド **command** を削除するには、**root** で以下のコマンドを実行します。

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

- ホワイトリストにコマンド **command** があるかどうかを確認するには、**root** で以下のコマンドを実行します。

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

True の場合は終了ステータスが **0** で **yes** が出力され、False の場合は終了ステータスが **1** で **no** が出力されます。

- ホワイトリストにあるセキュリティーコンテキストを一覧表示するには、**root** で以下のコマンドを実行します。

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

- ホワイトリストにコンテキスト **context** を追加するには、**root** で以下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コンテキスト **context** を削除するには、**root** で以下のコマンドを実行します。

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

- ホワイトリストにコンテキスト **context** があるかどうかを確認するには、**root** で以下のコマンドを実行します。

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

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

- ホワイトリストにあるユーザー ID を一覧表示するには、**root** で以下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** で以下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** で以下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、以下のコマンドを実行します。

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

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

- ホワイトリストにあるユーザー名を一覧表示するには、**root** で以下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** で以下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** で以下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、以下のコマンドを実行します。

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

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

3.12.3. 設定ファイル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

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:virtd_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/bin/python3 -Es /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

この例では、**user id** と **user name** の両方が使用されていますが、実際にはどちらか一方のオプションだけが必要です。Python はインタープリターとしてコマンドラインに追加されています。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。

注記

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が、**root** で実行するスクリプトが **/bin/firewall-cmd** のパスを使用するようになっているのであれば、これまでは **root** 以外のユーザーにのみ使用されていた **/usr/bin/firewall-cmd** パスに加え、このコマンドのパスも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、その名前で始まるすべてのコマンドが一致することを意味します。「*」がなければ、コマンドと引数が完全に一致する必要があります。

3.13. 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、FORWARD、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**、**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** で以下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** で以下のコマンドを実行します。

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

firewalld GUI 設定ツールを使用して **LogDenied** 設定を変更するには、**firewall-config** を起動し、**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。

第4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。

第5章 NFTABLES の概要

nftables フレームワークは、パケットの分離機能を提供し、**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
- **IPv4** プロトコルおよび **IPv6** プロトコルの両方に対する単一のフレームワーク
- 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のサポート (**nftrace**) およびトレースイベントの監視 (**nft** ツール)
- より一貫性のあるコンパクトな構文、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 **nftables** Netlink API で、低レベルの対話のために使用できます。

RHEL 8 では、**nftables** は、デフォルトの **firewalld** バックエンドとして動作します。**nftables** バックエンドは、ファイアウォール設定で以前の **iptables** バックエンドと後方互換性がありますが、**/etc/firewalld/firewalld.conf** ファイルの **FirewallBackend** オプションに **iptables** を設定して、バックエンドを **iptables** に戻すことができます。

nftables ルールセットに対するモジュールの効果は、**nft** list ruleset コマンドを使用して確認できます。このツールは、テーブル、チェーン、およびルールを **nftables** ルールセットに追加するため、**nft flush ruleset** などの **nftables** ルールセット操作は、以前は別々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。

どの種類のツールが存在するかをすばやく識別するために、バックエンド名を追加するようにバージョン情報が更新されました。RHEL 8 では、nftables ベースの **iptables** ツールで、次のバージョン文字列が出力されます。

```
$ iptables --version
iptables v1.8.0 (nf_tables)
```

一方、従来の iptables ツールが存在する場合は、次のバージョン情報が出力されます。

```
$ iptables --version
iptables v1.8.0 (legacy)
```

関連資料

- 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。

5.1. 関連情報

- 「What comes after iptables? Its successor, of course: nftables」 のブログ投稿では、**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。