Red Hat Enterprise Linux 8

ネットワークのセキュリティー保護

セキュリティー保護されたネットワークおよびネットワーク通信の設定

Last Updated: 2021-06-18

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

セキュリティー保護されたネットワークおよびネットワーク通信の設定

法律上の通知

概要

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

多様性を受け入れるオープンソースの強化 6

RED HAT ドキュメントへのフィードバック (英語のみ) 7

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 8
1.1. SSH と OPENSSH 8
1.2. OPENSSH サーバーの設定および起動 9
1.3. 鍵ベースの認証用の OPENSSH サーバーの設定 10
1.4. SSH 鍵ペアの生成 11
1.5. スマートカードに保存された SSH 鍵の使用 13
1.6. OPENSSH のセキュリティーの強化 14
1.7. SSH ジャンプホストを使用してリモートサーバーに接続 16
1.8. SSH-AGENT を使用して SSH キーでリモートマシンに接続する手順 18
1.9. 関連情報 18

第2章 SSH システムロールを使用した安全な通信の設定 20
2.1. SSHD システムロール変数 20
2.2. SSHD システムロールを使用した OPENSSH サーバーの設定 22
2.3. SSH システムロール変数 24
2.4. SSH システムロールを使用した OPENSSH クライアントの設定 26

第3章 TLS の計画および実施 28
3.1. SSL プロトコルおよび TLS プロトコル 28
3.2. RHEL 8 における TLS のセキュリティー上の検討事項 28
3.2.1. プロトコル 29
3.2.2. 暗号化スイート 29
3.2.3. 公開鍵の長さ 29
3.3. アプリケーションで TLS 設定の強化 30
3.3.1. Apache HTTP サーバー の設定 30
3.3.2. Nginx HTTP およびプロキシーサーバーの設定 31
3.3.3. Dovecot メールサーバーの設定 31

第4章 IPSEC を使用した VPN の設定 33
4.1. IPSEC VPN 実装としての LIBRESWAN 33
4.2. LIBRESWAN のインストール 34
4.3. ホスト間の VPN の作成 34
4.4. サイト間 VPN の設定 35
4.5. リモートアクセスの VPN の設定 36
4.6. メッシュ VPN の設定 37
4.7. LIBRESWAN で使用される認証方法 39
4.8. FIPS 準拠の IPSEC VPN のデプロイメント 40
4.9. パスワードによる IPSEC NSS データベースの保護 43
4.10. TCP を使用するように IPSEC VPN を設定 44
4.11. IPSEC 接続を加速化するためにボンディングでの ESP ハードウェアオフロードの設定 45
4.12. 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 46
4.13. IPSEC VPN 設定のトラブルシューティング 47
4.14. 関連情報 51

第5章 MACSEC を使用した同じ物理ネットワーク内のレイヤー 2 トラフィックの暗号化 53
5.1. NMCLI を使用した MACSEC 接続の設定 53

第6章 FIREWALLD の使用および設定 56
6.1. FIREWALLD の使用 56

6.1.1. firewalld、nftables、または iptables を使用する場合 56
6.1.2. ゾーン 56
6.1.3. 事前定義サービス 58
6.2. FIREWALLD の現在の状況および設定の表示 58
6.2.1. firewalld の現在の状況の表示 58
6.2.2. GUI を使用して許可されるサービスの表示 59
6.2.3. CLI を使用した firewalld 設定の表示 59
6.3. FIREWALLD でネットワークトラフィックの制御 60
6.3.1. 緊急時に CLI を使用してすべてのトラフィックの無効化 60
6.3.2. 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 61
6.3.3. GUI を使用して事前定義サービスでトラフィックを制御 61
6.3.4. 新しいサービスの追加 62
6.3.5. GUI を使用してポートを開く 63
6.3.6. GUI を使用してプロトコルを使用したトラフィックの制御 63
6.3.7. GUI を使用してソースポートを開く 63
6.4. CLI を使用したポートの制御 64
6.4.1. ポートを開く 64
6.4.2. ポートを閉じる 64
6.5. ファイアウォールゾーンでの作業 65
6.5.1. ゾーンの一覧 65
6.5.2. 特定ゾーンに対する firewalld 設定の修正 65
6.5.3. デフォルトゾーンの変更 66
6.5.4. ゾーンへのネットワークインターフェースの割り当て 66
6.5.5. nmcliを使用して接続にゾーンを割り当て 66
6.5.6. 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 67
6.5.7. 新しいゾーンの作成 67
6.5.8. ゾーンの設定ファイル 67
6.5.9. 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 68
6.6. 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 68
6.6.1. 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 68
6.6.2. ソースの追加 69
6.6.3. ソースの削除 69
6.6.4. ソースポートの追加 70
6.6.5. ソースポートの削除 70
6.6.6. 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 70
6.7. FIREWALLD を使用した NAT の設定 71
6.7.1. 異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト 71
6.7.2. IP アドレスのマスカレードの設定 72
6.8. ポート転送 72
6.8.1. リダイレクトするポートの追加 73
6.8.2. 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 73
6.8.3. リダイレクトしているポートの削除 73
6.8.4. 同じマシンで TCP ポート 88 に転送されるポート 80 の削除 74
6.9. ICMP リクエストの管理 74
6.9.1. ICMP リクエストの一覧表示およびブロック 74
6.9.2. GUI を使用した ICMP フィルターの設定 76
6.10. FIREWALLD を使用した IP セットの設定および制御 77
6.10.1. CLI を使用した IP セットオプションの設定 77
6.11. リッチルールの優先度設定 79
6.11.1. 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 79
6.11.2. リッチルールの優先度の設定 79
6.12. ファイアウォールロックダウンの設定 80
6.12.1. CLI を使用したロックダウンの設定 80

6.12.2. CLI を使用したロックダウン許可リストオプションの設定 80
6.12.3. 設定ファイルを使用したロックダウンの許可リストオプションの設定 82
6.13. 関連情報 83
インストールされているドキュメント 83
オンラインドキュメント 84

第7章 NFTABLES の使用 85
7.1. IPTABLES から NFTABLES への移行 85
7.1.1. firewalld、nftables、または iptables を使用する場合 85
7.1.2. iptables のルールを nftables ルールに変換 86
7.1.3. 一般的な iftables コマンドおよび nftables コマンドの比較 86
7.2. NFTABLES スクリプトの作成および実行 87
7.2.1. 対応している nftables スクリプトの形式 87
7.2.2. nftables スクリプトの実行 88
7.2.3. nftables スクリプトでコメントの使用 89
7.2.4. nftables スクリプトで変数の使用 89
値を 1 つ持つ変数 89
匿名セットを含む変数 89
7.2.5. nftables スクリプトへのファイルの追加 90
7.2.6. システムの起動時に nftables ルールの自動読み込み 90
7.3. NFTABLES テーブル、チェーン、およびルールの作成および管理 91
7.3.1. 標準のチェーン優先度の値およびテキスト名 91
7.3.2. nftables ルールセットの表示 92
7.3.3. nftables テーブルの作成 93
7.3.4. nftables チェーンの作成 94
7.3.5. nftables チェーンの最後に対するルールの追加 95
7.3.6. nftables チェーンの先頭へのルールの挿入 95
7.3.7. nftables チェーンの特定の位置へのルールの挿入 96
7.4. NFTABLES を使用した NAT の設定 97
7.4.1. 異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト 98
7.4.2. nftables を使用したマスカレードの設定 98
7.4.3. nftables を使用したソース NAT の設定 99
7.4.4. nftables を使用した宛先 NAT の設定 99
7.4.5. nftables を使用したリダイレクトの設定 100
7.5. NFTABLES コマンドでのセットの使用 101
7.5.1. nftables での匿名セットの使用 101
7.5.2. nftables で名前付きセットの使用 102
7.5.3. 関連情報 103
7.6. NFTABLES コマンドにおける決定マップの使用 103
7.6.1. nftables での匿名マップの使用 103
7.6.2. nftables での名前付きマップの使用 104
7.6.3. 関連情報 106
7.7. NFTABLES を使用したポート転送の設定 106
7.7.1. 着信パケットの別のローカルポートへの転送 106
7.7.2. 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 107
7.8. NFTABLES を使用した接続の量の制限 108
7.8.1. nftables を使用した接続数の制限 108
7.8.2. 1 分以内に新しい着信 TCP 接続を 11 個以上試行する IP アドレスのブロック 109
7.9. NFTABLES ルールのデバッグ 109
7.9.1. カウンターによるルールの作成 109
7.9.2. 既存のルールへのカウンターの追加 110
7.9.3. 既存のルールに一致するパケットの監視 111
7.10. NFTABLES ルールセットのバックアップおよび復元 112

7.10.1. ファイルへの nftables ルールセットのバックアップ 112
7.10.2. ファイルからの nftables ルールセットの復元 112
7.11. 関連情報 112

多様性を受け入れるオープンソースの強化

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

RED HAT ドキュメントへのフィードバック (英語のみ)

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 OPENSSH を使用した安全な通信の使用

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

Red Hat Enterprise Linux 8 には、基本的な **OpenSSH** パッケージ (一般的な **openssh** パッケージ、**openssh-server** パッケージ、および **openssh-clients** パッケージ) が含まれます。**OpenSSH** パッケージには、**OpenSSL** パッケージ (**openssl-libs**) が必要です。このパッケージは、重要な暗号化ライブラリーをいくつかインストールして、暗号化通信を提供する **OpenSSH** を有効にします。

1.1. SSH と OPENSSH

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

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

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

OpenSSH は、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く

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、**sshd (8)** の man ページの **FILES** セクションを参照してください。

関連情報

- **man -k ssh コマンドを使用して一覧表示される man** ページです。
- システム全体の暗号化ポリシーの使用

1.2. OPENSSH サーバーの設定および起動

お使いの環境と **OpenSSH** サーバー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.
```

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

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

ログインに成功すると表示されるメッセージを変更するには、サーバーの **/etc/motd** ファイルを編集する必要があります。詳細は、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、**sshd** を再起動して変更を適用します。

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

検証

1. **sshd** デーモンが実行していることを確認します。

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

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

2. SSH クライアントを使用して SSH サーバーに接続します。

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

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

関連情報

- ***sshd(8)*** and ***sshd_config(5)*** man pages.

1.3. 鍵ベースの認証用の OPENSSH サーバーの設定

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

前提条件

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

手順

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

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

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

```
PasswordAuthentication no
```

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

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

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

4. ***sshd*** *デーモンを再読み込みし、変更を適用します。*

```
# systemctl reload sshd
```

関連情報

- ***sshd(8)***, ***sshd_config(5)***, and ***setsebool(8)*** man pages.

1.4. SSH 鍵ペアの生成

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

重要

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

手順

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

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

```
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+. . . .      |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

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

2. *公開鍵をリモートマシンにコピーするには、次のコマンドを実行します。*

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

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

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

注記

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

検証

1. *パスワードなしで OpenSSH サーバーにログインします。*

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

関連情報

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

1.5. スマートカードに保存された SSH 鍵の使用

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

前提条件

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

手順

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

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

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

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

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

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

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

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

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

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

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

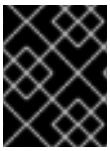
関連情報

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

1.6. OPENSSH のセキュリティーの強化

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

```
# systemctl reload sshd
```



重要

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

安全ではない接続プロトコルの無効化

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

鍵ベースの認証の有効化およびパスワードベースの認証の無効化

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

  ```
  PasswordAuthentication no
  ```

鍵のタイプ

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

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

システム全体の暗号化ポリシーの変更

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

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

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

1.7. SSH ジャンプホストを使用してリモートサーバーに接続

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

前提条件

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

手順

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

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

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

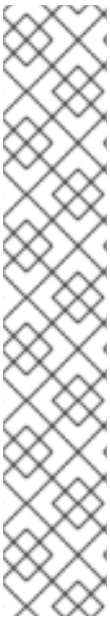
2. **ProxyJump** ディレクティブを使用してリモートサーバーのジャンプ設定を、ローカルシステムの **~/.ssh/config** ファイルに追加します。以下に例を示します。

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

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

```
$ ssh remote-server
```

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



注記

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

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

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

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

関連情報

- **ssh_config(5)** and **ssh(1)** man pages.

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

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

前提条件

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

手順

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

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

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

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

2. **ssh-agent** を起動します。

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

3. **ssh-agent** にキーを追加します。

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

検証

- オプション: SSH を使用してホストマシンにログインします。

  ```
  $ ssh example.user1@198.51.100.1

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

  パスフレーズを入力する必要がないことに注意してください。

1.9. 関連情報

- ***sshd(8)***、***ssh(1)***、***scp(1)***、***sftp(1)***、***ssh-keygen(1)***、***ssh-copy-id(1)***、***ssh_config(5)***、***sshd_config(5)***、***update-crypto-policies(8)***、および ***crypto-policies(7)*** の man ページ。
- OpenSSH ホームページ。
- 非標準設定でアプリケーションとサービスの SELinux の設定。
- firewalld でネットワークトラフィックの制御

第2章 SSH システムロールを使用した安全な通信の設定

管理者は、Red Hat Ansible Automation Platform を使うことで、任意の数の RHEL システムで一貫して、SSHD システムロールを使用して SSH サーバーを、SSH システムロールを使用して SSH クライアントを同時に設定することができます。

2.1. SSHD システムロール変数

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

これらの変数を設定しないと、システムロールは RHEL のデフォルト値に一致する **sshd_config** ファイルを作成します。

すべての場合において、ブール値は **sshd** 設定で **yes** と **no** として適切にレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。

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

以下のようにレンダリングします。

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSHD システムロールの変数

sshd_enable

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

sshd_skip_defaults

True に設定すると、システムロールはデフォルト値を適用しません。代わりに、**sshd** dict または **sshd_Key** 変数のいずれかを使用して、設定のデフォルトの完全なセットを指定します。デフォルトは **False** です。

sshd_manage_service

False に設定すると、サービスは管理されません。つまり、サービスは起動時に有効化されず、起動または再読み込みされません。Ansible サービスモジュールが現在 AIX で **enabled** になっていないため、コンテナーまたは AIX 内で実行する時以外はデフォルトで **True** に設定されます。

sshd_allow_reload

False に設定すると、設定の変更後に **sshd** は再読み込みされません。これはトラブルシューティングで役立ちます。変更した設定を適用するには、**sshd** を手動で再読み込みします。AIX を除き、**sshd_manage_service** と同じ値にデフォルト設定されます。ここで、**sshd_manage_service** はデフォルトで **False** に設定されますが、**sshd_allow_reload** はデフォルトで **True** に設定されます。

sshd_install_service

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

-

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

sshd_service

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

sshd

設定が含まれる dict。以下に例を示します。

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

sshd_OptionName

dict の代わりに、**sshd_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**sshd** dict の値を上書きします。以下に例を示します。

```
sshd_Compression: no
```

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

dict のリスト、または Match セクションの dict のみ。これらの変数は、**sshd** dict で定義されている一致するブロックを上書きしないことに注意してください。すべてのソースは作成された設定ファイルに反映されます。

SSHD システムロールのセカンダリー変数

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

sshd_packages

この変数を使用して、インストール済みパッケージのデフォルト一覧を上書きできます。

sshd_config_owner、sshd_config_group、sshd_config_mode

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

sshd_config_file

このロールが作成した **openssh** サーバー設定を保存するパス。

sshd_binary

openssh の **sshd** 実行可能ファイルへのパス。

sshd_service

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

sshd_verify_hostkeys

デフォルトは **auto** です。**auto** に設定すると、生成された設定ファイルに存在するホストキーがすべて一覧表示され、存在しないパスが生成されます。また、パーミッションおよびファイルの所有者はデフォルト値に設定されます。これは、ロールがデプロイメント段階で使用され、サービスが

*最初の試行で起動できるようにする場合に便利です。このチェックを無効にするには、この変数を空のリスト **[]** に設定します。*

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

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

sshd_sysconfig

*RHEL ベースのシステムでは、この変数は **sshd** サービスに関する追加情報を設定します。**true** に設定すると、このロールは以下の設定に基づいて **/etc/sysconfig/sshd** 設定ファイルも管理します。デフォルトは **false** です。*

sshd_sysconfig_override_crypto_policy

*RHEL 8 では、**true** に設定すると、この変数はシステム全体の暗号化ポリシーを上書きします。デフォルトは **false** です。*

sshd_sysconfig_use_strong_rng

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

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

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

前提条件

- *1 つ以上の* **管理対象ノード** *(SSHD システムロールで設定するシステム) へのアクセスおよびパーミッション。*
- **コントロールノード** *(Red Hat Ansible Engine が他のシステムを設定するシステム) へのアクセスおよびパーミッション。*
  コントロールノードには、
  - *Red Hat Ansible Engine がインストールされています。*
  - ***rhel-system-roles** パッケージがインストールされています。*
  - *管理対象ノード一覧表示するインベントリーファイルがあります。*

手順

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

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

2. *以下の例のように、テキストエディターを使用してコピーした Playbook を開きます。*

```
# vim path/custom-playbook.yml

---
- hosts: all
  tasks:
```

```
- name: Configure sshd to prevent root and password login except from particular subnet
  include_role:
    name: rhel-system-roles.sshd
  vars:
    sshd:
      # root login and password login is enabled only from a particular subnet
      PermitRootLogin: no
      PasswordAuthentication: no
      Match:
      - Condition: "Address 192.0.2.0/24"
        PermitRootLogin: yes
        PasswordAuthentication: yes
```

Playbook は、対応するように、管理ノードを SSH サーバーとして設定します。

- *パスワードと **root** ユーザーのログインは無効になっています*
- *パスワードと **root** ユーザーのログインが、サブネット **192.0.2.0/24** からのみ有効です*

設定に合わせて変数を変更できます。詳細は「SSHD Server System Role variables」を参照してください。

3. *オプション: Playbook の構文を確認します。*

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. *インベントリーファイルで Playbook を実行します。*

```
# ansible-playbook -i inventory_file path/custom-playbook.yml

...

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

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

検証

1. *SSH サーバーにログインします。*

```
$ ssh user1@10.1.1.1
```

ここでは、

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

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

```
$ vim /etc/ssh/sshd_config

# Ansible managed
```

```
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
AuthorizedKeysFile .ssh/authorized_keys
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
PasswordAuthentication no
PermitRootLogin no
PrintMotd no
Subsystem sftp /usr/libexec/openssh/sftp-server
SyslogFacility AUTHPRIV
UsePAM yes
X11Forwarding yes
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

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

   a. *IP アドレスを確認します。*

   ```
   $ hostname -I
   192.0.2.1
   ```

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

   b. ***root*** *でサーバーに接続します。*

   ```
   $ ssh root@10.1.1.1
   ```

関連情報

- ***/usr/share/doc/rhel-system-roles/sshd/README.md*** *file.*
- *man ページの* ***ansible-playbook(1)***

2.3. SSH システムロール変数

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

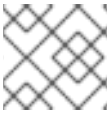
これらの変数を設定しないと、システムロールは RHEL のデフォルト値に一致するグローバル ***ssh_config*** *ファイルを作成します。*

すべてのケースでブール値が、 ***ssh*** *設定で* ***yes*** *または* ***no*** *として正しくレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。*

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

以下のようにレンダリングします。

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



注記

設定オプションには、大文字と小文字が区別されます。

SSH システムロールの変数

ssh_user

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

ssh_skip_defaults

デフォルトは **auto** です。**auto** に設定すると、システムロールはシステム全体の設定ファイル **/etc/ssh/ssh_config** を書き込み、そこで定義した RHEL のデフォルトを保持します。**ssh_drop_in_name** 変数を定義してドロップイン設定ファイルを作成すると、**ssh_skip_defaults** 変数を自動的に無効にします。

ssh_drop_in_name

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

警告

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

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

ssh

設定オプションとその値が含まれる dict。

ssh_OptionName

dict の代わりに、**ssh_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**ssh** dict の値を上書きします。

ssh_additional_packages

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

ssh_config_file

ロールが生成した設定ファイルを保存するパス。デフォルト値:

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

ssh_config_owner, ssh_config_group, ssh_config_mode

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

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

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

前提条件

- *1 つ以上の**管理対象ノード** (SSH システムロールで設定するシステム) へのアクセスおよびパーミッション。*
- **コントロールノード** *(Red Hat Ansible Engine が他のシステムを設定するシステム) へのアクセスおよびパーミッション。*
  コントロールノードには、
  - *Red Hat Ansible Engine がインストールされています。*
  - ***rhel-system-roles** パッケージがインストールされています。*
  - *管理対象ノード一覧表示するインベントリーファイルがあります。*

手順

1. *以下の内容を含む新しい **playbook.yml** ファイルを作成します。*

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    include_role:
      name: rhel-system-roles.ssh
    vars:
      ssh_user: root
      ssh:
        Compression: true
        GSSAPIAuthentication: no
```

```
    ControlMaster: auto
    ControlPath: ~/.ssh/.cm%C
    Host:
      - Condition: example
        Hostname: example.com
        User: user1
  ssh_ForwardX11: no
```

*この Playbook は、以下の設定を使用して、管理対象ノードで **root** ユーザーの SSH クライアント設定を行います。*

- *圧縮が有効になっています。*
- *ControlMaster multiplexing は **auto** に設定されます。*
- ***example.com** ホストに接続するためのエイリアスの **example** は **user1** です。*
- *ホストエイリアスの **example** が作成され、これは **user1** のユーザー名を持つ **example.com** ホストへの接続を表します。*
- *X11 転送が無効化されています。*

必要に応じて、これらの変数は設定に応じて変更できます。詳細は、「SSH Client Role variables」を参照してください。

2. *オプション: Playbook の構文を確認します。*

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. *インベントリーファイルで Playbook を実行します。*

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

検証

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

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

上記の Playbook の例の適用後、設定ファイルに以下の内容が含まれる必要があります。

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

第3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。

3.1. 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*

3.2. 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。

3.2.1. プロトコル

最新バージョンの TLS は、最高のセキュリティーメカニズムを提供します。古いバージョンの TLS に対応しないといけないような特別な事態がない限り、システムは、TLS バージョン 1.2 以上を使用して接続をネゴシエートできるようにしてください。RHEL 8 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。

3.2.2. 暗号化スイート

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

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

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

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。

3.2.3. 公開鍵の長さ

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

警告

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

関連情報

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

3.3. 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。

3.3.1. 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
```

3.3.2. 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
    ....
}
```

3.3.3. 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
```

関連情報

- *man* ページの**config (5) および ciphers(1)**
- *TTLS(Transport Layer Security)および Datagram Transport Layer Security(DTLS)のセキュアブートに関する推奨事項。*
- *Mozilla SSL 設定ジェネレーター。*
- *SSL サーバーテスト*

第4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。

4.1. 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 のバリアントは、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。

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

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

IPsec プロトコルは、以下の 2 つの操作モードを提供します。

- **トンネルモード** (デフォルト)
- **トランスポートモード**

IKE を使用せずに IPsec を使用してカーネルを設定できます。これは、**手動キーリング** と呼ばれます。また、**ip xfrm** コマンドを使用して手動キーを設定できますが、これはセキュリティー上の理由からは強く推奨されません。**Libreswan** 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ーサービス (**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。

4.2. 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** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

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

```
# 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
```

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

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

4. 鍵をインポートしたら、**ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

5. **Libreswan** を起動します。

```
# ipsec setup start
```

6. 接続を読み込みます。

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

7. トンネルを確立します。

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

8. **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
```

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

4.5. リモートアクセスの VPN の設定

ロードウォーリアーとは、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
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow/deny client
    # pam-authorize=yes
    # load connection, don't initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

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

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
    # allow narrowing to the server's suggested assigned IP and remote subnet
    narrowing=yes
    # Support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    # Initiate connection
    auto=start
```

4.6. メッシュ VPN の設定

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

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

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

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、Dongtag 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
```

3. 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
```

4. 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
```

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

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

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

すべてのノードでは、例外の一覧が同じか、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。

6. ノードを再起動して、設定したメッシュに追加します。

```
# systemctl restart ipsec
```

7. 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
```

4.7. 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マテリアルが残っていないと、接続は失敗します。詳細は、man ページの* ***ipsec.secrets(5)*** *を参照してください。*

警告

動的の 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。

4.8. FIPS 準拠の IPSEC VPN のデプロイメント

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

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

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

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

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

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

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

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

5. 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

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

4.9. 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** ディレクトリーには 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

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

- **certutil(1)** の man ページ
- Government Standards ナレッジベース記事。

4.10. TCP を使用するように IPSEC VPN を設定

Libreswan は、RFC 8229 で説明されているように、IKE パケットおよび IPsec パケットの TCP カプセル化に対応します。この機能により、UDP 経由でトラフィックが転送されないように、IPsec VPN をネットワークに確立し、セキュリティーのペイロード (ESP) を強化できます。フォールバックまたはメインの VPN トランスポートプロトコルとして TCP を使用するように VPN サーバーおよびクライアントを設定できます。TCP カプセル化にはパフォーマンスコストが大きくなるため、UDP がシナリオで永続的にブロックされている場合に限り、TCP を主な VPN プロトコルとして使用してください。

前提条件

- リモートアクセス VPN が設定されている。

手順

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

```
listen-tcp=yes
```

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

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

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

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

関連情報

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

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

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

前提条件

- ボンディングのすべてのネットワークカードは、ESP ハードウェアオフロードをサポートしています。
- ネットワークドライバーは、ボンドデバイスで ESP ハードウェアオフロードに対応しています。RHEL 8.4 では、**ixgbe** ドライバーのみがこの機能をサポートしています。
- ボンディングが設定されており、動作します。
- ボンディングは **active-backup** モードを使用します。ボンディングドライバーは、この機能の他のモードをサポートしません。
- IPsec 接続が設定され、動作します。

手順

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

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

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

2. **bond0** 接続を再度アクティブにします。

```
# nmcli connection up bond0
```

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

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

4. **ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

検証

1. ボンディングのアクティブなポートを表示します。

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

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

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

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

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

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

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

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- 『**ネットワークの設定および管理**』の「ネットワークボンディングの設定」セクション
- IPsec を使用した VPN の設定
- 『ネットワークのセキュリティー保護』の「IPsec を使用した VPN の設定」を参照してください。

4.12. 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

- システム全体の暗号化ポリシーの使用

4.13. IPSEC VPN 設定のトラブルシューティング

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

基本的な接続のトラブルシューティング

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

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

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

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

接続に問題があることを確認するには、以下を実行します。

1. **vpn.example.com** 接続をもう一度読み込みます。

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

2. 次に、VPN 接続を開始します。

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

ファイアウォール関連の問題

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

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

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

- IKEv1 の場合は、最初のコマンドの出力は以下のようになります。

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

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

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

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

アルゴリズム、プロトコル、およびポリシーが一致しない場合

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

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

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

- *IKE アルゴリズムが一致しない例:*

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

- *IPsec アルゴリズムが一致しない例:*

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

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

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

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

- *IKEv1 の IP アドレス範囲で一致しない例:*

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

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

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

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

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

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

最大伝送単位 (MTU)

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

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

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

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

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

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

ネットワークアドレス変換 (NAT)

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

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

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

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

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

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

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

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

カーネル IPsec サブシステムのバグ

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

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

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

Libreswan のログ

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

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

ipsec サービスのライブログを表示するには、次のコマンドを実行します。

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

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

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

関連情報

- ログファイルを使用した問題のトラブルシューティング
- firewalld の使用および設定
- **tcpdump(8)** および **ipsec.conf(5)** の man ページ

4.14. 関連情報

- man ページの **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)**、および **ipsec_rsasigkey(8)**
- **/usr/share/doc/libreswan-version/** directory.

- アップストリームプロジェクトの Web サイト です。
- Libreswan プロジェクトの Wiki です。
- Libreswan に関する全 man ページ
- NIST Special Publication 800-77: Guide to IPsec VPNs .

第5章 MACSEC を使用した同じ物理ネットワーク内のレイヤー2 トラフィックの暗号化

本セクションでは、イーサーネットリンクのすべてのトラフィックのセキュアな通信に MACsec を設定する方法を説明します。

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

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

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

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

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

つまり、MACsec を設定する方法は 2 つあります。

- ホスト対ホスト
- 他のホストに切り替えるホスト

重要

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

以下の例は、事前共有キーを使用して 2 つのホスト間で MACsec を設定する方法を示しています。

5.1. NMCLI を使用した MACSEC 接続の設定

nmcli ツールを使用して、MACsec を使用するようにイーサーネットインターフェースを設定できます。この手順では、イーサーネットインターフェースを使用してネットワークトラフィックを暗号化する MACsec 接続を作成する方法を説明します。

この MACsec で保護されるネットワークで通信する必要があるすべてのホストでこの手順を実行します。

手順

ホスト A の場合:

- *MACsec を設定する最初のホストで、事前共有キー用の接続関連キー (CAK: connectivity association key) および接続関連キー名 (CKN: connectivity-association key name) を作成します。*

  a. *16 バイトの 16 進数の CAK を作成します。*

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

  b. *32 バイトの 16 進数の CKN を作成します。*

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

ホスト A および B の場合：

1. *MACsec 接続を作成します。*

```
# nmcli connection add type macsec con-name macsec0 ifname macsec0
connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-cak
50b71a8ef0bd5751ea76de6d6c98c03a  macsec.mka-ckn
f2b4297d39da7330910a7abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

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

2. *MACsec 接続で IP を設定します。*

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

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

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

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

3. *接続をアクティベートします。*

```
# nmcli connection up macsec0
```

検証手順

1. *トラフィックが暗号化されていることを確認するには、以下を入力します。*

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

2. *暗号化されていないトラフィックを表示するには、以下を入力します。*

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

3. MACsec の統計を表示するには、以下を入力します。

```
# ip macsec show
```

4. integrity-only (encrypt off) および encryption (encrypt on) の各タイプの保護に対して個々のカウンターを表示するには、以下を入力します。

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

関連情報

- MACsec およびそのアーキテクチャーの詳細は、MACsec: a different solution to encrypt network traffic blog を参照してください。

第6章 FIREWALLD の使用および設定

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

firewalld は、D-Bus インターフェースを使用して、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ことができます。許可されているトラフィックは、コンピューターが接続するネットワークと、このネットワークが割り当てられているセキュリティーレベルに従います。ファイアウォールサービスは、特定のサービスに着信トラフィックを許可するのに必要なすべての設定を扱う事前定義のルールで、ゾーンに適用されます。

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

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

6.1. FIREWALLD の使用

6.1.1. firewalld、nftables、または iptables を使用する場合

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

- **firewalld**: 簡単なファイアウォールのユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。
- **nftables**: **nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。
- **iptables**: Red Hat Enterprise Linux 8 の **iptables** ユーティリティーは、**レガシー** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux 8 で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

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。

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

関連情報

- **firewalld.zone(5)** *の man ページ*

6.1.3. 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、**firewalld.service(5)** の man ページ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

- ***firewalld.service(5)** の man ページ*

6.2. FIREWALLD の現在の状況および設定の表示

6.2.1. 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には、**root** ユーザーで **firewall-cmd --list-all** を使用します。

6.2.2. 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。

グラフィカルなファイアウォール設定ツールは、コマンドラインを使用して起動できます。

手順

- 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には、次のコマンドを実行します。

```
$ 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。

6.2.3. 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。

手順

- 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。**firewalld** の man ページか、コマンドの help でオプションの一覧を表示します。*

```
# firewall-cmd --help
```

- *現在のゾーンで許可されているサービスを表示するには、次のコマンドを実行します。*

```
# 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。*

6.3. FIREWALLD でネットワークトラフィックの制御

6.3.1. 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。

2. *パニックモードをオフにし、ファイアウォールを永続設定に戻します。パニックモードを無効にするには、以下のコマンドを実行します。*

```
# firewall-cmd --panic-off
```

検証

- *パニックモードが有効または無効であるかを確認するには、以下のコマンドを実行します。*

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

6.3.2. 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 ...
[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
```

6.3.3. GUI を使用して事前定義サービスでトラフィックを制御

この手順では、グラフィカルユーザーインターフェースを使用して、事前定義サービスでネットワークトラフィックを制御する方法を説明します。

手順

1. 事前定義したサービスまたはカスタマイズしたサービスを有効または無効にするには、以下を行います。

   a. **firewall-config** 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。

   b. **Services** タブを選択します。

   c. 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。

2. サービスを編集するには、以下を行います。

   a. **firewall-config** ツールを起動します。

   b. **Configuration** メニューから **Permanent** を選択します。**Services** ウィンドウの下部に、その他のアイコンおよびメニューボタンが表示されます。

   c. 設定するサービスを選択します。

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

注記

ランタイム モードでは、サービス設定を変更できません。

6.3.4. 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。

6.3.5. 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。

6.3.6. GUI を使用してプロトコルを使用したトラフィックの制御

特定のプロトコルを使用してファイアウォールを経由したトラフィックを許可するには、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。

6.3.7. 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。

6.4. 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。

6.4.1. 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
```

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. 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。

6.4.2. ポートを閉じる

開し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
```

警告

このコマンドにより、ポートとして開かれているポートのみが表示されます。サービスとして開いているポートは表示されません。したがって、**--list-ports** ではなく **--list-all** オプションの使用を検討してください。

2. 「許可されているポート」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
```

6.5. 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。

6.5.1. ゾーンの一覧

この手順では、コマンドラインでゾーンの一覧を表示する方法を説明します。

手順

1. 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。

2. 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
```

3. 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
```

6.5.2. 特定ゾーンに対する firewalld 設定の修正

cli を使用して cli を使用して事前定義されたサービスでトラフィック を 制御したり、サービスを追加するか、現在の作業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
```

6.5.3. 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

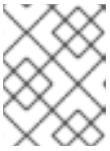
デフォルトゾーンを設定するには、以下を行います。

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。

6.5.4. 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。

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 --permanent
```

6.5.5. 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、**firewalld** 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 modify profile connection.zone zone_name
```

2. 接続の再読み込みを行います。

```
# nmcli connection up profile
```

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

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

手順

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

```
ZONE=zone_name
```

6.5.7. 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

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
```

6.5.8. 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 protocol="udp" port="1025-65535"/>
  <port protocol="tcp" port="1025-65535"/>
</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。

6.5.9. 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 --permanent --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

6.6. 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

6.6.1. 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。

6.6.2. 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。

注記

ネットワーク範囲が重複している複数のゾーンを追加する場合は、ゾーン名で順序付けされ、最初のゾーンのみが考慮されます。

- 現在のゾーンにソースを設定するには、次のコマンドを実行します。

```
# firewall-cmd --add-source=<source>
```

- 特定ゾーンのソース IP アドレスを設定するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-source=<source>
```

以下の手順は、**信頼される** ゾーンで 192.168.2.15 からのすべての着信トラフィックを許可します。

手順

1. すべての利用可能なゾーンを一覧表示します。

```
# 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
```

6.6.3. 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
```

6.6.4. 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

- 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>
```

6.6.5. 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

- 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>
```

6.6.6. 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、他のトラフィックがブロックされている間に **192.0.2.0/24** ネットワークからの HTTP トラフィックのみを許可します。

警告

このシナリオを設定する場合は、**default** のターゲットを持つゾーンを使用します。**192.0.2.0/24** からのトラフィックではネットワーク接続がすべて許可されるため、ターゲットが **ACCEPT** に設定されたゾーンを使用することは、セキュリティー上のリスクになります。

手順

1. すべての利用可能なゾーンを一覧表示します。

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

2. IP 範囲を **internal** ゾーンに追加し、ソースから発信されるトラフィックをゾーン経由でルーティングします。

```
# firewall-cmd --zone=internal --add-source=192.0.2.0/24
```

3. ***http*** *サービスを* ***internal*** *ゾーンに追加します。*

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

4. *新しい設定を永続化します。*

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

検証

- ***internal*** *ゾーンがアクティブで、サービスが許可されていることを確認します。*

```
# firewall-cmd --zone=internal --list-all
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.0.2.0/24
  services: cockpit dhcpv6-client mdns samba-client ssh http
  ...
```

関連情報

- *ゾーンの詳細は、man ページの* **firewalld.zones(5)** *を参照してください。*

6.7. FIREWALLD を使用した NAT の設定

firewalld *では、以下のネットワークアドレス変換 (NAT) タイプを設定できます。*

- *マスカレーディング*
- *ソース NAT (SNAT)*
- *宛先 NAT (DNAT)*
- *リダイレクト*

6.7.1. 異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト

以下は、さまざまなネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

*この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダーは、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。*
マスカレードおよび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、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、チェーンフックに応じてパケットをローカルマシンにリダイレクトする DNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

6.7.2. 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
```

6.8. 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。

6.8.1. 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

1. 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
```

2. 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。

   a. 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
```

   b. マスカレードを有効にします。

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

6.8.2. 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
```

6.8.3. リダイレクトしているポートの削除

この手順では、リダイレクトしたポートを削除する方法を説明します。

手順

1. 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>
```

2. 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には、以下を実行します。

   a. 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>
```

   b. 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
```

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

この手順では、ポートのリダイレクト設定を削除する方法を説明します。

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
```

6.9. 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。

6.9.1. 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い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、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>
```

情報をまったく指定せずに 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。

- ゾーンのターゲットを **DROP** に設定します。

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

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 --permanent --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 --permanent --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
```

6.9.2. 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。

6.10. 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
```

6.10.1. 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、更新したエントリー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ー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。

6.11. 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。

6.11.1. 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。

6.11.2. 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
  }
}
```

6.12. 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、すべてのアプリケーションでファイアウォール変更を要求できなくするか、ロックダウンの許可リストに追加されたアプリケーションのみがファイアウォール変更を要求できるようにすることが可能になります。ロックダウン設定はデフォルトで無効になっています。これを有効にすると、ローカルのアプリケーションやサービスによるファイアウォールへの望ましくない設定変更を確実に防ぐことができます。

6.12.1. CLI を使用したロックダウンの設定

この手順では、コマンドラインでロックダウンを有効または無効にする方法を説明します。

- 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
  ```

6.12.2. CLI を使用したロックダウン許可リストオプションの設定

ロックダウンの許可リストには、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。許可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
  ```

- 許可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
```

- 許可リストに 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'
```

- 許可リストから 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 コマンドが許可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。

- 許可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
```

- 許可リストに 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
```

- 許可リストから 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 コンテキストが許可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。

- 許可リストにあるユーザー ID すべて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
```

- 許可リストにユーザー 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
```

- *許可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
```

- *許可リストにユーザー 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。*

- *許可リストにある全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
```

- *許可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
```

- *許可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
```

- *ユーザー名 (user) が許可リストに含まれ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。*

6.12.3. 設定ファイルを使用したロックダウンの許可リストオプションの設定

*デフォルトの許可リスト設定ファイルには、**NetworkManager** コンテキストと、**libvirt** のデフォルトコンテキストが含まれます。リストには、ユーザー ID (0) もあります。*

```
<?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>
```

*以下の許可リスト設定ファイルの例では、**firewall-cmd** ユーティリティーのコマンドと、ユーザー ID が **815** である user のコマンドをすべて有効にしています。*

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <command name="/usr/libexec/platform-python -s /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

この例では、**user id** と **user name** の両方が使用されていますが、実際にはどちらか一方のオプションだけが必要です。Python はインタープリターとしてコマンドラインに追加されています。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/** ディレクトリーへのシンボリックリンクとなります。つまり、**root** で **firewall-cmd** のパスを実行すると **/bin/firewall-cmd** に対して解決しますが、**/usr/bin/firewall-cmd** が使用できるようになっています。新たなスクリプトは、すべて新しい格納場所を使用する必要があります。ただし、**root** で実行するスクリプトが **/bin/firewall-cmd** へのパスを使用するようになっているのであれば、これまでは **root** 以外のユーザーにのみ使用されていた **/usr/bin/firewall-cmd** パスに加え、このコマンドのパスも許可リストに追加する必要があります。

コマンドの名前属性の最後にある * は、その名前で始まるすべてのコマンドが一致することを意味します。* がなければ、コマンドと引数が完全に一致する必要があります。

6.13. 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。
- **firewalld.lockdown-whitelist(5)** の man ページ: **firewalld** ロックダウン許可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す。*
- *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7章 NFTABLES の使用

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、**ebtables** ツール、および **ipset** 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組み込みルックアップテーブルを使用
- **IPv4** プロトコルおよび **IPv6** プロトコルに対する 1 つのフレームワーク
- 完全ルールセットのフェッチ、更新、および保存を行わず、すべてアトミックに適用されるルール
- ルールセットにおけるデバッグおよびトレースへの対応 (**nftrace**) およびトレースイベントの監視 (**nft** ツール)
- より統一されたコンパクトな構文、プロトコル固有の拡張なし
- サードパーティーのアプリケーション用 Netlink API

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。

ルールセット変更が適用されていることを表示するには、**nft list ruleset** コマンドを使用します。これらのツールは、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。

7.1. IPTABLES から NFTABLES への移行

サーバーを RHEL 8 にアップグレードしたり、ファイアウォール設定が **iptables** ルールを使用している場合は、**iptables** ルールを **nftables** に移行できます。

7.1.1. firewalld、nftables、または iptables を使用する場合

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

- **firewalld**: 簡単なファイアウォールのユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。
- **nftables**: **nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。
- **iptables**: Red Hat Enterprise Linux 8 の **iptables** ユーティリティーは、**レガシー** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux 8 で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

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。

7.1.2. 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す。*

7.1.3. 一般的な iftables コマンドおよび nftables コマンドの比較

*以下は、一般的な **iptables** コマンドと **nftables** コマンドの比較です。*

- *すべてのルールを一覧表示します。*

iptables	nftables
iptables-save	**nft list ruleset**

- *特定のテーブルおよびチェーンを一覧表示します。*

iptables	nftables
iptables -L	**nft list table ip filter**
iptables -L INPUT	**nft list chain ip filter INPUT**
iptables -t nat -L PREROUTING	**nft list chain ip nat PREROUTING**

***nft** コマンドは、テーブルおよびチェーンを事前に作成しません。これらは、ユーザーが手動で作成した場合にのみ存在します。*

例: firewalld が生成するルールの一覧表示

```
# nft list table inet firewalld
# nft list table ip firewalld
# nft list table ip6 firewalld
```

7.2. 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/** ディレクトリーに作成します。このスクリプトには、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。

7.2.1. 対応している nftables スクリプトの形式

nftables スクリプト環境は、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す。

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
  chain example_chain {
    # Chain for incoming packets that drops all packets that
    # are not explicitly allowed by any rule in this chain
    type filter hook input priority 0; policy drop;

    # Accept connections to port 22 (ssh)
    tcp dport ssh accept
  }
}
```

- **nft** コマンドと同じ構文を使用できます。

```
#!/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
```

7.2.2. nftables スクリプトの実行

nftables スクリプトを実行するには、**nft** ユーティリティーに渡すか、スクリプトを直接実行します。

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

- **nftables** スクリプトを **nft** ユーティリティーに渡して実行するには、次のコマンドを実行します。

```
# nft -f /etc/nftables/example_firewall.nft
```

- **nftables** スクリプトを直接実行するには、次のコマンドを実行します。

  a. 以下の手順は、一度だけ必要です。

  i. スクリプトが以下のシバンシーケンスで始まることを確認します。

```
#!/usr/sbin/nft -f
```

重要

-f パラメーターを指定しないと、**nft** ユーティリティーはスクリプトを読み取らず、**Error: syntax error, unexpected newline, expecting string** を表示します。

  ii. 必要に応じて、スクリプトの所有者を **root** に設定します。

```
# chown root /etc/nftables/example_firewall.nft
```

  iii. 所有者のスクリプトを実行ファイルに変更します。

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

  b. 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、**chmod(1)** の man ページを参照してください。
- システムの起動時に **nftables** ルールを読み込む方法は、『**ネットワークの設定および管理**』の「システム起動時に nftables ルールの自動読み込み」を参照してください。

7.2.3. nftables スクリプトでコメントの使用

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

例7.1 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
...
```

7.2.4. 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。

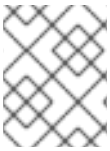
```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

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、『**ネットワークの設定および管理**』の「nftables コマンドでセットの使用」を参照してください。
- セットの詳細は、『**ネットワークの設定および管理** 』の「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。
- 決定マップの詳細は、『**ネットワークの設定および管理**』の「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。

7.2.5. nftables スクリプトへのファイルの追加

nftables スクリプト環境を使用すると、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7.2 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7.3 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。

7.2.6. システムの起動時に nftables ルールの自動読み込み

*systemd サービス **nftables** は、**/etc/sysconfig/nftables.conf** ファイルに含まれるファイアウォールスクリプトを読み込みます。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. *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
   ```

3. ***nftables** サービスを有効にします。*

   ```
   # systemctl enable nftables
   ```

関連情報

- *詳細は、『**ネットワークの設定および管理**』の「対応している nftables スクリプトの形式」を参照してください。*

7.3. NFTABLES テーブル、チェーン、およびルールの作成および管理

*本セクションでは、**nftables** ルールセットを表示する方法と、その管理方法を説明します。*

7.3.1. 標準のチェーン優先度の値およびテキスト名

*チェーンを作成する場合は、**priority** で、同じ **hook** 値を持つチェーンが通過する順番を指定する、整数値または標準名を設定できます。*

*名前と値は、デフォルトのチェーンの登録時に **xtables** が使用する優先度に基づいて定義されます。*

注記

***nft list chains** コマンドは、デフォルトで文字列の優先順位の値を表示します。**-y** オプションをコマンドに渡すことで、数値を表示できます。*

> ***例7.4 テキスト値を使用した優先順位の設定***
>
> *以下のコマンドは、標準の優先度 **50** を使用して **example_table** に **example_chain** という名前のチェーンを作成します。*
>
> ***# nft add chain inet** example_table example_chain **{ type** filter **hook** input **priority** 50 **\; policy accept \; }***
>
> *優先度は標準の値であるため、テキスト値を使用することもできます。*
>
> ***# nft add chain inet** example_table example_chain **{ type** filter **hook** input **priority** security **\; policy accept \; }***

表7.1 標準優先順位名、ファミリー、およびフックの互換性マトリックス

名前	値	ファミリー	フック
raw	-300	**ip**、**ip6**、**inet**	all
mangle	-150	**ip**、**ip6**、**inet**	all
dstnat	-100	**ip**、**ip6**、**inet**	prerouting
filter	0	**ip**、**ip6**、**inet**、**arp**、**netdev**	all
security	50	**ip**、**ip6**、**inet**	all
srcnat	100	**ip**、**ip6**、**inet**	postrouting

*すべてのファミリーは同じ値を使用しますが、**bridge** ファミリーは以下の値を使用します。*

表7.2 ブリッジファミリーの標準的な優先順位名およびフックの互換性

名前	値	フック
dstnat	-300	prerouting
filter	-200	all
out	100	出力
srcnat	300	postrouting

関連情報

- *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。*

7.3.2. 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

- ルールセットを表示するには、以下のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport 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。

7.3.3. 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
```

2. 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。

7.3.4. nftables チェーンの作成

チェーンは、ルールのコンテナーです。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。

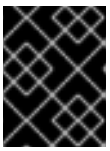
前提条件

- 新しいチェーンを追加するテーブルが存在する。

手順

1. **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るのを回避するには、セミコロンの前に \ エスケープ文字を付けます。

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ければ、すぐに切断されることに注意してください。

2. 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 filter; 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。

7.3.5. nftables チェーンの最後に対するルールの追加

本セクションでは、既存の **nftables** チェーンの最後にルールを追加する方法を説明します。

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 filter; 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。

7.3.6. nftables チェーンの先頭へのルールの挿入

本セクションでは、既存の **nftables** チェーンの先頭にルールを追加する方法を説明します。

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 filter; 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。

7.3.7. nftables チェーンの特定の位置へのルールの挿入

本セクションでは、**nftables** チェーンで、既存のルールの前後にルールを追加する方法を説明します。これにより、正しい場所に新しいルールを配置することができます。

前提条件

- ルールを追加するチェーンが存在する。

手順

1. **nft -a list ruleset** コマンドを使用して、ハンドルなど、**example_table** 内のすべてのチェーンとそのルールを表示します。

```
# nft -a list table inet example_table
table inet example_table { # handle 1
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
```

```
    tcp dport 443 accept # handle 3
    tcp dport 389 accept # handle 4
  }
}
```

-a を使用すると、ハンドルが表示されます。次の手順で新しいルールを配置するときに、この情報が必要です。

2. **example_table** の **example_chain** チェーンに新しいルールを挿入します。

   - ハンドル **3** の前に、ポート **636** で TCP トラフィックを許可するルールを挿入するには、次のコマンドを実行します。

     ```
     # nft insert rule inet example_table example_chain position 3 tcp dport 636 accept
     ```

   - ハンドル **3** の後ろに、ポート **80** で TCP トラフィックを許可するルールを追加するには、次のコマンドを実行します。

     ```
     # nft add rule inet example_table example_chain position 3 tcp dport 80 accept
     ```

3. 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 -a list table inet example_table
table inet example_table { # handle 1
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    tcp dport 80 accept # handle 6
    tcp dport 389 accept # handle 4
  }
}
```

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。

7.4. NFTABLES を使用した NAT の設定

nftables を使用すると、以下のネットワークアドレス変換 (NAT) タイプを設定できます。

- マスカレーディング
- ソース NAT (SNAT)
- 宛先 NAT (DNAT)
- リダイレクト

7.4.1. 異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト

以下は、さまざまなネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

このNAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダーは、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードおよび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、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターにDNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、チェーンフックに応じてパケットをローカルマシンにリダイレクトするDNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

7.4.2. nftables を使用したマスカレードの設定

マスカレードを使用すると、ルーターは、インターフェ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。

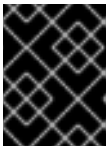
手順

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
```

7.4.3. nftables を使用したソース NAT の設定

ルーターでは、ソース NAT (SNAT) を使用して、インターフェ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。

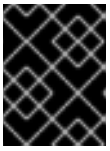
手順

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

- 詳細は、『**ネットワークの設定および管理**』の「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」を参照してください。

7.4.4. 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。*

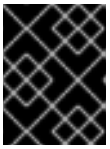
手順

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

- *詳細は、『**ネットワークの設定および管理**』の「異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト」を参照してください。*

7.4.5. nftables を使用したリダイレクトの設定

redirect 機能は、チェーンフックに応じてパケットをローカルマシンにリダイレクトする宛先ネットワークアドレス変換 (DNAT) の特殊なケースです。

以下の手順では、ローカルホストの **22** ポートに送信される着信トラフィックおよび転送されたトラフィックを **2222** ポートにリダイレクトする方法を説明します。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに **prerouting** チェーンを追加します。

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

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. **22** ポートの着信トラフィックを **2222** ポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule nat prerouting tcp dport 22 redirect to 2222
```

関連情報

- 詳細は、『**ネットワークの設定および管理**』の「異なる NAT タイプ: マスカレード、ソース NAT、宛先 NAT、リダイレクト」を参照してください。

7.5. NFTABLES コマンドでのセットの使用

nftables フレームワークは、セットをネイティブに対応します。たとえば、ルールが複数の IP アドレス、ポート番号、インターフェース、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。

7.5.1. nftables での匿名セットの使用

匿名セットには、ルールで直接使用する **{ 22, 80, 443 }** などの中括弧で囲まれたコンマ区切りの値が含まれます。IP アドレスやその他の一致基準に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」で説明されているように名前付きセットを使用 します。

前提条件

- **inet** ファミリーに **example_chain** チェーンおよび **example_table** テーブルがある。

手順

1. たとえば、ポート **22**、**80**、および **443** に着信トラフィックを許可するルールを、**example_table** の **example_chain** に追加するには、次のコマンドを実行します。

```
# nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2. 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 filter; policy accept;
    tcp dport { ssh, http, https } accept
  }
}
```

7.5.2. nftables で名前付きセットの使用

nftables フレームワークは、変更可能な名前付きセットに対応します。名前付きセットは、テーブル内の複数のルールで使用できる要素の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 32 ビットの正の整数値 (**0** から **2147483647**) にすることができます。

前提条件

- **example_chain** チェーンと **example_table** テーブルが存在する。

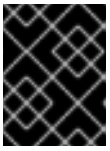
手順

1. 空のファイルを作成します。以下の例では、IPv4 アドレスのセットを作成します。
   - 複数の IPv4 アドレスを格納することができるセットを作成するには、次のコマンドを実行します。

```
# nft add set inet example_table example_set { type ipv4_addr \; }
```

   - IPv4 アドレス範囲を保存できるセットを作成するには、次のコマンドを実行します。

```
# nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
```



重要

シェルで、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。

7.5.3. 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。

7.6. NFTABLES コマンドにおける決定マップの使用

ディクショナリーとしても知られている決定マップにより、**nft** は一致基準をアクションにマッピングすることで、パケット情報に基づいてアクションを実行できます。

7.6.1. nftables での匿名マップの使用

匿名マップは、ルールで直接使用する **{ match_criteria : action }** ステートメントです。ステートメントには、複数のコンマ区切りマッピングを含めることができます。

匿名マップの欠点は、マップを変更する場合には、ルールを置き換える必要があることです。動的なソリューションの場合は、「nftables での名前付きマップの使用」 の説明に従って名前付きマップを使用します。

この例では、匿名マップを使用して、IPv4 プロトコルおよび IPv6 プロトコルの TCP パケットと UDP パケットの両方を別のチェーンにルーティングし、着信 TCP パケットと UDP パケットを個別にカウント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. *匿名マップを持つルールを* ***incoming_traffic*** *に追加します。*

```
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump tcp_packets,
udp : jump udp_packets }
```

匿名マップはパケットを区別し、プロトコルに基づいて別のカウンターチェーンに送信します。

8. *トラフィックカウンターの一覧を表示する場合は、* ***example_table*** *を表示します。*

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
    counter packets 36379 bytes 2103816
  }

  chain udp_packets {
    counter packets 10 bytes 1559
  }

  chain incoming_traffic {
    type filter hook input priority filter; policy accept;
    ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
  }
}
```

tcp_packets *チェーンおよび* ***udp_packets*** *チェーンのカウンターは、受信パケットとバイトの両方を表示します。*

7.6.2. nftables での名前付きマップの使用

nftables フレームワークは、名前付きマップに対応します。テーブルの複数のルールでこれらのマップを使用できます。匿名マップに対する別の利点は、名前付きマップを使用するルールを置き換えることなく、名前付きマップを更新できることです。

名前付きマップを作成する場合は、要素のタイプを指定する必要があります。

- *一致する部分に **192.0.2.1** などの IPv4 アドレスが含まれるマップの場合は **ipv4_addr**。*
- *一致する部分に **2001:db8:1::1** などの IPv6 アドレスが含まれるマップの場合は **ipv6_addr**。*
- ***52:54:00:6b:66:42** などのメディアアクセス制御 (MAC) アドレスを含むマップの場合は **ether_addr**。*
- *一致する部分に **tcp** などのインターネットプロトコルタイプが含まれるマップの場合は **inet_proto**。*
- *一致する部分に **ssh** や **22** などのインターネットサービス名のポート番号が含まれるマップの場合は **inet_service**。*
- *一致する部分にパケットマークが含まれるマップの場合は **mark**。パケットマークは、任意の正の 32 ビットの整数値 (**0** ~ **2147483647**) にできます。*
- *一致する部分にカウンターの値が含まれるマップの場合は **counter**。カウンター値は、正の値の 64 ビットであれば任意の値にすることができます。*
- *一致する部分にクォータ値が含まれるマップの場合は **quota**。クォータの値は、64 ビットの整数値にできます。*

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。名前付き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についても説明します。

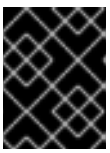
手順

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、マップからエントリーを削除します。

```
# 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 filter; policy accept;
    ip saddr vmap @example_map
  }
}
```

7.6.3. 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。

7.7. 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。

7.7.1. 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
```

7.7.2. 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。

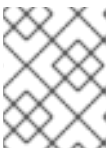
手順

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
```

7.8. NFTABLES を使用した接続の量の制限

nftables *を使用して、接続の数を制限したり、一定の数の接続の確立を試みる IP アドレスをブロックして、システムリソースを過剰に使用されないようにします。*

7.8.1. nftables を使用した接続数の制限

nft *ユーティリティーの* ***ct count*** *パラメーターを使用すると、管理者は接続数を制限できます。この手順では、着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。*

7.8.2.1 分以内に新しい着信 TCP 接続を 11 個以上試行する IP アドレスのブロック

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 アドレスを拒否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. **denylist** という名前のセットを **filter** テーブルに追加します。

```
# nft add set ip filter denylist { 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 **denylist** 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 @denylist { ip saddr }
```

5. **denylist** 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 @denylist drop
```

関連情報

- 接続のテストの詳細は、『**ネットワークの設定および管理**』の「nftables で名前付きセットの使用」を参照してください。

7.9. 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。

7.9.1. 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の詳細は、『**ネットワークの設定および管理**』の「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 filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

7.9.2. 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の詳細は、『**ネットワークの設定および管理**』の「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 filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールの代わりに、**counter** パラメーターを使用してカウンターを追加します。以下の例は、前の手順で表示したルールの代わりに、カウンターを追加します。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter accept
```

3. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
```

```
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

7.9.3. 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 filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールの代わりに **meta nftrace 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 nftrace set 1 accept
```

3. **nft monitor** コマンドを使用して、トレースを表示します。以下の例は、コマンドの出力をフィルタリングして、**inet example_table example_chain** が含まれるエントリーのみを表示します。

```
# nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr 52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728 tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1 accept (verdict accept)
...
```

警告

nft monitor コマンドは、トレースが有効になっているルールの数と、一致するトラフィックの量に応じて、大量の出力を表示できます。**grep** などのユーティリティーを使用して出力をフィルタリングします。

7.10. 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する方法のほか、ファイルからルールを復元する方法についても説明します。

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。

7.10.1. 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

- **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、以下のようになります。

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

  - JSON 形式の場合は、以下のようになります。

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

7.10.2. 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

- **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
```

7.11. 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。