Red Hat Enterprise Linux 9

ファイアウォールおよびパケットフィルターの設定

firewalld サービス、nftables フレームワーク、および XDP パケットフィルタリング機能の管理

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 ファイアウォールおよびパケットフィルターの設定

firewalld サービス、nftables フレームワーク、および XDP パケットフィルタリング機能の管理

法律上の通知

概要

ファイアウォールなどのパケットフィルターは、ルールを使用して、着信、発信、および転送されるネットワークトラフィックを制御します。Red Hat Enterprise Linux (RHEL) では、firewalld サービスと nftables フレームワークを使用して、ネットワークトラフィックをフィルタリングし、パフォーマンスが重要なファイアウォールを構築できます。カーネルの Express Data Path (XDP) 機能を使用して、ネットワークインターフェイスでネットワークパケットを非常に高速に処理またはドロップすることもできます。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

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

Jira からのフィードバック送信 (アカウントが必要)

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

第1章 FIREWALLD の使用および設定

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

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

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

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

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。

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 の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

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

1.2. ファイアウォールゾーン

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

firewalld はゾーンに関して厳格な原則に従います。

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

原則 4 と 5 は原則 3 の結果です。

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

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

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

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

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

block

- 適した例: **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。
- 受け入れる接続: システム内から開始したネットワーク接続のみ。

dmz

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

drop

適した例: 着信ネットワークパケットは、通知なしで遮断されます。

- 受け入れる接続: 発信ネットワーク接続のみ。

external

- 適した例: マスカレードを特にルーター用に有効にした外部ネットワーク。ネットワーク上の他のコンピューターを信頼できない状況。
- 受け入れる接続: 選択した着信接続のみ。

home

- 適した例: ネットワーク上の他のコンピューターをほぼ信頼できる自宅の環境。
- 受け入れる接続: 選択した着信接続のみ。

internal

- 適した例: ネットワーク上の他のコンピューターをほぼ信頼できる内部ネットワーク。
- 受け入れる接続: 選択した着信接続のみ。

public

- 適した例: ネットワーク上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。
- 受け入れる接続: 選択した着信接続のみ。

trusted

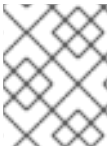
- 受け入れる接続: すべてのネットワーク接続。

work

適した例: ネットワーク上の他のコンピューターをほぼ信頼できる職場の環境。

- 受け入れる接続: 選択した着信接続のみ。

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



注記

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

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

関連情報

- **firewalld.zone(5)** の man ページ

1.3. ファイアウォールポリシー

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

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

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

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

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

1.4. ファイアウォールのルール

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

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

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

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

1.5. ゾーンの設定ファイル

firewalld ゾーン設定ファイルには、ゾーンに対する情報があります。これは、XML ファイル形式で、ゾーンの説明、サービス、ポート、プロトコル、icmp-block、マスカレード、転送ポート、およびリッチ言語ルールです。ファイル名は **zone-name.xml** となります。**zone-name** の長さは 17 文字に制限さ

れます。ゾーンの設定ファイルは、**/usr/lib/firewalld/zones/** ディレクトリーおよび **/etc/firewalld/zones/** ディレクトリーに置かれています。

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

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

関連情報

- **firewalld.zone** man ページ

1.6. 事前定義された FIREWALLD サービス

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

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

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

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

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

次の方法で **firewalld** を設定できます。

- 以下のユーティリティーを使用します。
  - **firewall-config** - グラフィカルユーティリティー

- **firewall-cmd** - コマンドラインユーティリティー
- **firewall-offline-cmd** - コマンドラインユーティリティー

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

関連情報

- **firewalld.service(5)** の man ページ

1.7. ファイアウォールゾーンでの作業

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

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

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

たとえば、次のような利点が得られます。

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

前提条件

- **firewalld** サービスが実行している。

手順

1. 利用可能なファイアウォールゾーンをリスト表示します。

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

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

2. この設定に使用するゾーンを選択します。

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

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

4. ネットワークインターフェイスをファイアウォールゾーンに割り当てます。

   a. 使用可能なネットワークインターフェイスをリスト表示します。

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

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

   b. 選択したゾーンにネットワークインターフェイスを割り当てます。

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

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

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

検証

1. 選択したゾーンの更新後の設定を表示します。

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

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

1.7.2. デフォルトゾーンの変更

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

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

前提条件

- **firewalld** サービスが実行している。

手順

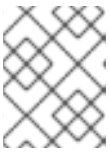
デフォルトゾーンを設定するには、以下を行います。

1. 現在のデフォルトゾーンを表示します。

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

2. 新しいデフォルトゾーンを設定します。

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



注記

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

1.7.3. ゾーンへのネットワークインターフェイスの割り当て

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

手順

特定インターフェイスにゾーンを割り当てるには、以下を行います。

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

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

2. 別のゾーンにインターフェイスを割り当てます。

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

1.7.4. nmcli を使用して接続にゾーンを割り当て

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

手順

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

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

2. 接続をアクティベートします。

```
# nmcli connection up profile
```

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

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

注記

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

手順

1. 接続プロファイルへのパスとその形式を決定します。

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

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

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

2. 形式に応じて、対応するファイルを更新します。

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

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

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

```
ZONE=internal
```

3. 接続プロファイルを再読み込みします。

```
# nmcli connection reload
```

4. 接続プロファイルを再度アクティベートします。

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

検証

- インターフェイスのゾーンを表示します。以下に例を示します。

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

1.7.6. 新しいゾーンの作成

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

前提条件

- **firewalld** サービスが実行している。

手順

1. 新しいゾーンを作成します。

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

2. 新しいゾーンを使用可能にします。

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

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

検証

- 作成したゾーンが永続設定に追加されたかどうかを確認します。

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

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

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

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

前提条件

- **firewalld** サービスが実行している。

手順

ゾーンにターゲットを設定するには、以下を行います。

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

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

2. ゾーンに新しいターゲットを設定します。

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

関連情報

- **firewall-cmd(1)** man ページ

1.8. FIREWALLD でネットワークトラフィックの制御

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

1.8.1. CLI を使用した事前定義サービスによるトラフィックの制御

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

前提条件

- **firewalld** サービスが実行している。

手順

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

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

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

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

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

このコマンドは、デフォルトゾーンで利用可能なサービスのリストを表示します。

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

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

このコマンドは、指定したサービスをデフォルトゾーンに追加します。

4. 新しい設定を永続化します。

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

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

検証

1. すべての永続的なファイアウォールのルールをリスト表示します。

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

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

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

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

永続的な設定が無効な場合、コマンドは詳細を含むエラーを返します。

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

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

1.8.2. GUI を使用した事前定義サービスによるトラフィックの制御

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

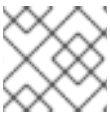
前提条件

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

手順

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

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

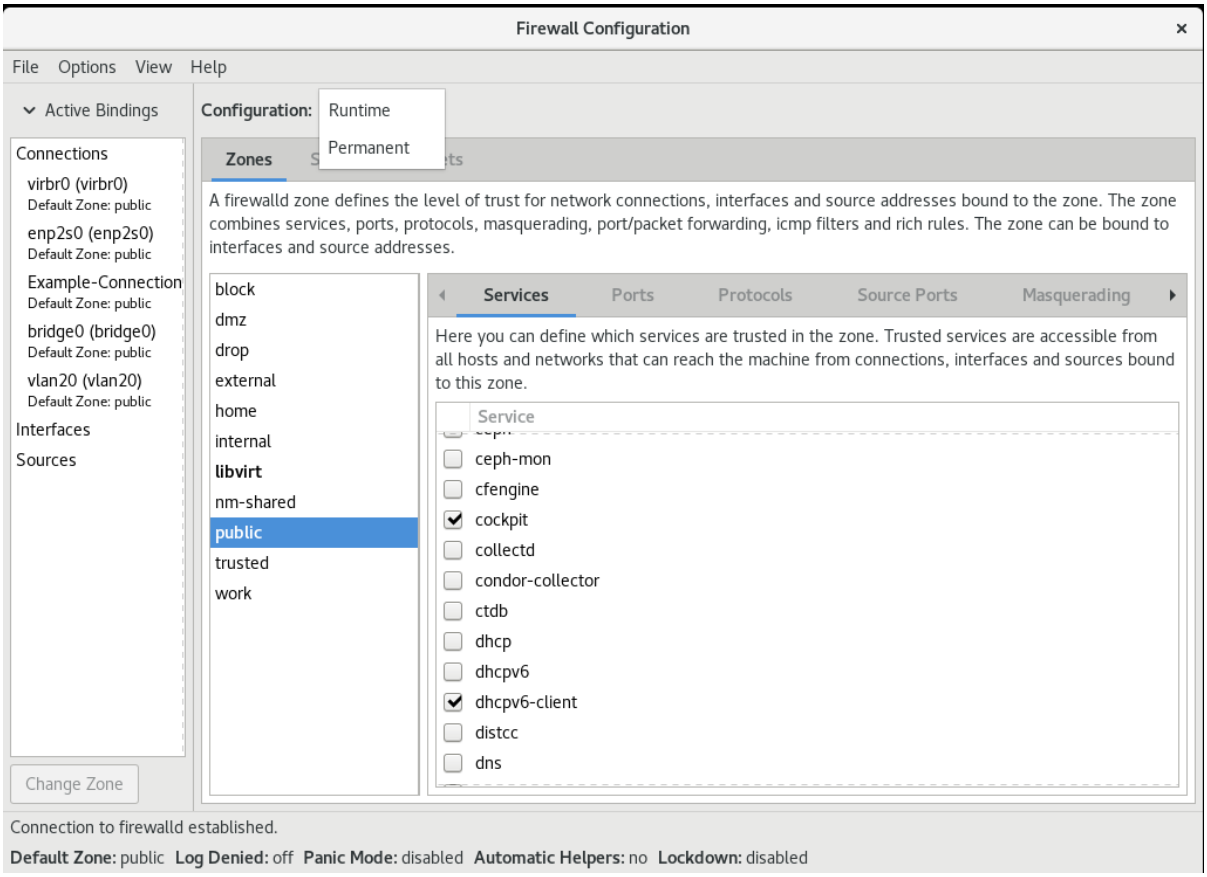


注記

Runtime モードでは、サービス設定を変更できません。

検証

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



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

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

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

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

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

前提条件

- **firewalld** サービスが実行している。

手順

1. 現在アクティブなファイアウォールゾーンを確認します。

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

2. HTTPS サービスを適切なゾーンに追加します。

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

3. ファイアウォール設定を再読み込みします。

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

検証

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

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

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

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

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

開いているポートが不要になった場合は、**firewalld** ユーティリティーを使用してポートを閉じることができます。

重要

不要なポートをすべて閉じて、潜在的な攻撃対象領域を減らし、不正アクセスや脆弱性悪用のリスクを最小限に抑えてください。

手順

1. 許可されているポートのリストを表示します。

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

デフォルトでは、このコマンドはデフォルトゾーンで有効になっているポートをリスト表示します。

注記

このコマンドでは、ポートとして開かれているポートのみが表示されます。サービスとして開かれているポートは表示されません。その場合は、**--list-ports** の代わりに **--list-all** オプションの使用を検討してください。

2. 許可されているポートのリストからポートを削除し、着信トラフィックに対して閉じます。

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

このコマンドは、ゾーンからポートを削除します。ゾーンを指定しない場合は、デフォルトゾーンからポートが削除されます。

3. 新しい設定を永続化します。

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

ゾーンを指定しない場合、このコマンドは、ランタイムの変更をデフォルトゾーンの永続的な設定に適用します。

検証

1. アクティブなゾーンをリスト表示し、検査するゾーンを選択します。

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

2. 選択したゾーンで現在開いているポートをリスト表示し、不使用または不要なポートが閉じているかどうかを確認します。

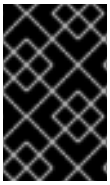
```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

1.8.5. CLI を使用したトラフィックの制御

firewall-cmd コマンドを使用すると、次のことが可能です。

- ネットワークトラフィックの無効化
- ネットワークトラフィックの有効化

その結果、たとえばシステムの防御を強化したり、データのプライバシーを確保したり、ネットワークリソースを最適化したりすることができます。



重要

パニックモードを有効にすると、ネットワークトラフィックがすべて停止します。したがって、そのマシンへの物理アクセスがある場合、またはシリアルコンソールを使用してログインする場合に限り使用してください。

手順

1. ネットワークトラフィックを直ちに無効にするには、パニックモードをオンにします。

```
# firewall-cmd --panic-on
```

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

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

検証

- パニックモードを有効または無効にするには、次のコマンドを実行します。

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

1.8.6. GUI を使用してプロトコルを使用したトラフィックの制御

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

前提条件

- **firewall-config** パッケージがインストールされている

手順

1. firewall-config ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側で **Protocols** タブを選択し、**Add** ボタンをクリックします。**Protocol** ウィンドウが開きます。
3. リストからプロトコルを選択するか、**Other Protocol** チェックボックスを選択し、そのフィールドにプロトコルを入力します。

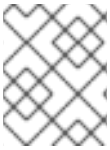
1.9. ゾーンを使用し、ソースに応じた着信トラフィックの管理

ゾーンを使用して、そのソースに基づいて着信トラフィックを管理するゾーンを使用できます。このコンテキストでの着信トラフィックとは、システム宛てのデータ、または **firewalld** を実行しているホストを通過するデータです。ソースは通常、トラフィックの発信元の IP アドレスまたはネットワーク範囲を指します。その結果、着信トラフィックをソートして異なるゾーンに割り当て、そのトラフィックが到達できるサービスを許可または禁止することができます。

ソースアドレスによる一致は、インターフェイス名による一致よりも優先されます。ソースをゾーンに追加すると、ファイアウォールは、インターフェイスベースのルールよりも着信トラフィックに対するソースベースのルールを優先します。これは、着信トラフィックが特定のゾーンに指定されたソースアドレスと一致する場合、トラフィックが通過するインターフェイスに関係なく、ソースアドレスに関連付けられたゾーンによってトラフィックの処理方法が決定されることを意味します。一方、インターフェイスベースのルールは通常、特定のソースベースのルールに一致しないトラフィックのためのフォールバックです。これらのルールは、ソースがゾーンに明示的に関連付けられていないトラフィックに適用されます。これにより、特定のソース定義ゾーンがないトラフィックのデフォルトの動作を定義できます。

1.9.1. ソースの追加

着信トラフィックを特定のゾーンに転送する場合は、そのゾーンにソースを追加します。ソースは、CIDR (Classless Inter-domain Routing) 表記法の IP アドレスまたは IP マスクになります。



注記

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

- 現在のゾーンにソースを設定するには、次のコマンドを実行します。

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

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

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

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

手順

1. 利用可能なすべてのゾーンをリストします。

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

2. 永続化モードで、信頼ゾーンにソース IP を追加します。

```
# firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. 新しい設定を永続化します。

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

1.9.2. ソースの削除

ゾーンからソースを削除すると、当該ソースに指定したルールは、そのソースから発信されたトラフィックに適用されなくなります。代わりに、トラフィックは、その発信元のインターフェイスに関連付けられたゾーンのルールと設定にフォールバックするか、デフォルトゾーンに移動します。

手順

1. 必要なゾーンに対して許可されているソースのリストを表示します。

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

2. ゾーンからソースを永続的に削除します。

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

3. 新しい設定を永続化します。

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

1.9.3. ソースポートの削除

ソースポートを削除して、送信元ポートに基づいてトラフィックの分類を無効にします。

手順

- ソースポートを削除するには、次のコマンドを実行します。

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

1.9.4. ゾーンおよびソースを使用して特定ドメインのみに対してサービスの許可

特定のネットワークからのトラフィックを許可して、マシンのサービスを使用するには、ゾーンおよびソースを使用します。以下の手順では、他のトラフィックがブロックされている間に **192.0.2.0/24** ネットワークからの HTTP トラフィックのみを許可します。

警告

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

手順

1. 利用可能なすべてのゾーンをリストします。

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

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

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

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

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

4. 新しい設定を永続化します。

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

検証

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

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

関連情報

- **firewalld.zones(5)** の man ページ

1.10. ゾーン間で転送されるトラフィックのフィルタリング

firewalld を使用すると、異なる **firewalld** ゾーン間のネットワークデータのフローを制御できます。ルールとポリシーを定義することで、これらのゾーン間を移動するトラフィックをどのように許可またはブロックするかを管理できます。

ポリシーオブジェクト機能は、**firewalld** で正引きフィルターと出力フィルターを提供します。**firewalld** を使用して、異なるゾーン間のトラフィックをフィルタリングし、ローカルでホストされている仮想マシンへのアクセスを許可して、ホストを接続できます。

1.10.1. ポリシーオブジェクトとゾーンの関係

ポリシーオブジェクトを使用すると、サービス、ポート、リッチルールなどの firewalld のプリミティブをポリシーに割り当てることができます。ポリシーオブジェクトは、ステートフルおよび一方向の方法でゾーン間を通過するトラフィックに適用することができます。

```
# firewall-cmd --permanent --new-policy myOutputPolicy

# firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST

# firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

HOST および **ANY** は、イングレスゾーンおよびエグレスゾーンのリストで使用されるシンボリックゾーンです。

- **HOST** シンボリックゾーンは、firewalld を実行しているホストから発信されるトラフィック、またはホストへの宛先を持つトラフィックのポリシーを許可します。
- **ANY** シンボリックゾーンは、現行および将来のすべてのゾーンにポリシーを適用します。**ANY** シンボリックゾーンは、すべてのゾーンのワイルドカードとして機能します。

1.10.2. 優先度を使用したポリシーのソート

同じトラフィックセットに複数のポリシーを適用できるため、優先度を使用して、適用される可能性のあるポリシーの優先順位を作成する必要があります。

ポリシーをソートする優先度を設定するには、次のコマンドを実行します。

```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

この例では、**-500** の優先度は低くなりますが、優先度は高くなります。したがって、-500 は、-100 より前に実行されます。

優先度の数値が小さいほど優先度が高く、最初に適用されます。

1.10.3. ポリシーオブジェクトを使用した、ローカルでホストされているコンテナーと、ホストに物理的に接続されているネットワークとの間でのトラフィックのフィルタリング

ポリシーオブジェクト機能を使用すると、ユーザーは Podman ゾーンと firewalld ゾーン間のトラフィックをフィルタリングできます。

注記

Red Hat は、デフォルトではすべてのトラフィックをブロックし、Podman ユーティリティーに必要なサービスを選択して開くことを推奨します。

手順

1. 新しいファイアウォールポリシーを作成します。

```
# firewall-cmd --permanent --new-policy podmanToAny
```

2. Podman から他のゾーンへのすべてのトラフィックをブロックし、Podman で必要なサービスのみを許可します。

```
# firewall-cmd --permanent --policy podmanToAny --set-target REJECT
# firewall-cmd --permanent --policy podmanToAny --add-service dhcp
# firewall-cmd --permanent --policy podmanToAny --add-service dns
# firewall-cmd --permanent --policy podmanToAny --add-service https
```

3. 新しい Podman ゾーンを作成します。

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

4. ポリシーのイングレスゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

5. 他のすべてのゾーンのエグレスゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

エグレスゾーンを ANY に設定すると、Podman と他のゾーンの間でフィルタリングすることになります。ホストに対してフィルタリングする場合は、エグレスゾーンを HOST に設定します。

6. firewalld サービスを再起動します。

```
# systemctl restart firewalld
```

検証

- 他のゾーンに対する Podman ファイアウォールポリシーを検証します。

```
# firewall-cmd --info-policy podmanToAny
podmanToAny (active)
  ...
  target: REJECT
  ingress-zones: podman
  egress-zones: ANY
  services: dhcp dns https
  ...
```

1.10.4. ポリシーオブジェクトのデフォルトターゲットの設定

ポリシーには --set-target オプションを指定できます。以下のターゲットを使用できます。

- **ACCEPT** - パケットを受け入れます

- **DROP** - 不要なパケットを破棄します
- **REJECT** - ICMP 応答で不要なパケットを拒否します
- **CONTINUE** (デフォルト) - パケットは、次のポリシーとゾーンのルールに従います。

```
# firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

検証

- ポリシーに関する情報の確認

```
# firewall-cmd --info-policy mypolicy
```

1.10.5. DNAT を使用して HTTPS トラフィックを別のホストに転送する

Web サーバーがプライベート IP アドレスを持つ DMZ で実行されている場合は、宛先ネットワークアドレス変換 (DNAT) を設定して、インターネット上のクライアントがこの Web サーバーに接続できるようにすることができます。この場合、Web サーバーのホスト名はルーターのパブリック IP アドレスに解決されます。クライアントがルーターの定義済みポートへの接続を確立すると、ルーターはパケットを内部 Web サーバーに転送します。

前提条件

- DNS サーバーが、Web サーバーのホスト名をルーターの IP アドレスに解決している。
- 次の設定を把握している。
  - 転送するプライベート IP アドレスおよびポート番号
  - 使用する IP プロトコル
  - パケットをリダイレクトする Web サーバーの宛先 IP アドレスおよびポート

手順

1. ファイアウォールポリシーを作成します。

```
# firewall-cmd --permanent --new-policy <example_policy>
```

ポリシーは、ゾーンとは対照的に、入力、出力、および転送されるトラフィックのパケットフィルタリングを許可します。ローカルで実行されている Web サーバー、コンテナー、または仮想マシン上のエンドポイントにトラフィックを転送するには、このような機能が必要になるため、これは重要です。

2. イングレストラフィックとエグレストラフィックのシンボリックゾーンを設定して、ルーター自体がローカル IP アドレスに接続し、このトラフィックを転送できるようにします。

```
# firewall-cmd --permanent --policy=<example_policy> --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=<example_policy> --add-egress-zone=ANY
```

--add-ingress-zone=HOST オプションは、ローカルで生成され、ローカルホストから送信されるパケットを参照します。**--add-egress-zone=ANY** オプションは、任意のゾーンに向かうトラフィックを参照します。

3. トラフィックを Web サーバーに転送するリッチルールを追加します。

```
# firewall-cmd --permanent --policy=<example_policy> --add-rich-rule='rule
family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp"
to-port="443" to-addr="192.51.100.20"'
```

リッチルールは、ルーターの IP アドレス (192.0.2.1) のポート 443 から Web サーバーの IP アドレス (192.51.100.20) のポート 443 に TCP トラフィックを転送します。

4. ファイアウォール設定ファイルをリロードします。

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

5. カーネルで 127.0.0.0/8 のルーティングを有効にします。

- 変更を永続化するには、次を実行します。

```
# echo "net.ipv4.conf.all.route_localnet=1" > /etc/sysctl.d/90-enable-route-
localnet.conf
```

このコマンドは、**route_localnet** カーネルパラメーターを永続的に設定し、システムの再起動後も設定が確実に保持されるようにします。

- システムを再起動することなく直ちに設定を適用するには、次のコマンドを実行します。

```
# sysctl -p /etc/sysctl.d/90-enable-route-localnet.conf
```

sysctl コマンドは、オンザフライで変更を適用するのに便利ですが、システムを再起動すると設定は元に戻ります。

検証

1. Web サーバーに転送したルーターの IP アドレスおよびポートに接続します。

```
# curl https://192.0.2.1:443
```

2. オプション: **net.ipv4.conf.all.route_localnet** カーネルパラメーターがアクティブであることを確認します。

```
# sysctl net.ipv4.conf.all.route_localnet
net.ipv4.conf.all.route_localnet = 1
```

3. **<example_policy>** がアクティブであり、必要な設定 (特にソース IP アドレスとポート、使用するプロトコル、宛先 IP アドレスとポート) が含まれていることを確認します。

```
# firewall-cmd --info-policy=<example_policy>
example_policy (active)
  priority: -1
  target: CONTINUE
  ingress-zones: HOST
  egress-zones: ANY
  services:
  ports:
```

```
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
 rule family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp" to-
port="443" to-addr="192.51.100.20"
```

関連情報

- **firewall-cmd(1)**、**firewalld.policies(5)**、**firewalld.richlanguage(5)**、**sysctl(8)**、および **sysctl.conf(5)** の man ページ
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

1.11. FIREWALLD を使用した NAT の設定

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

- マスカレーディング
- 宛先 NAT (DNAT)
- リダイレクト

1.11.1. ネットワークアドレス変換のタイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレーディング

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダー (ISP) は、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

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

1.11.2. IP アドレスのマスカレードの設定

システムで IP マスカレードを有効にできます。IP マスカレードは、インターネットにアクセスする際にゲートウェイの向こう側にある個々のマシンを隠します。

手順

1. **external** ゾーンなどで IP マスカレーディングが有効かどうかを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --query-masquerade
```

このコマンドでは、有効な場合は **yes** と出力され、終了ステータスは **0** になります。無効の場合は **no** と出力され、終了ステータスは **1** になります。**zone** を省略すると、デフォルトのゾーンが使用されます。

2. IP マスカレードを有効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --add-masquerade
```

3. この設定を永続化するには、**--permanent** オプションをコマンドに渡します。

4. IP マスカレードを無効にするには、**root** で次のコマンドを実行します。

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

この設定を永続化するには、**--permanent** をコマンドラインに渡します。

1.11.3. DNAT を使用した着信 HTTP トラフィックの転送

宛先ネットワークアドレス変換 (DNAT) を使用して、着信トラフィックを 1 つの宛先アドレスおよびポートから別の宛先アドレスおよびポートに転送できます。通常、外部ネットワークインターフェイスからの着信リクエストを特定の内部サーバーまたはサービスにリダイレクトする場合に役立ちます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 次の内容を含む **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルを作成します。

```
net.ipv4.ip_forward=1
```

この設定によって、カーネルでの IP 転送が有効になります。これにより、内部 RHEL サーバーがルーターとして機能し、ネットワークからネットワークへパケットを転送するようになります。

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルから設定をロードします。

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. 着信 HTTP トラフィックを転送します。

```
# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toaddr=198.51.100.10:toport=8080 --permanent
```

上記のコマンドは、次の設定で DNAT ルールを定義します。

- **--zone=public** - DNAT ルールを設定するファイアウォールゾーン。必要なゾーンに合わせて調整できます。
- **--add-forward-port** - ポート転送ルールを追加することを示すオプション。
- **port=80** - 外部宛先ポート。
- **proto=tcp** - TCP トラフィックを転送することを示すプロトコル。
- **toaddr=198.51.100.10** - 宛先 IP アドレス。
- **toport=8080** - 内部サーバーの宛先ポート。
- **--permanent** - 再起動後も DNAT ルールを永続化するオプション。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- 使用したファイアウォールゾーンの DNAT ルールを確認します。

```
# firewall-cmd --list-forward-ports --zone=public
port=80:proto=tcp:toport=8080:toaddr=198.51.100.10
```

あるいは、対応する XML 設定ファイルを表示します。

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="80" protocol="tcp" to-port="8080" to-addr="198.51.100.10"/>
  <forward/>
</zone>
```

関連情報

- ランタイム時のカーネルパラメーターの設定
- **firewall-cmd(1)** man ページ

1.11.4. 非標準ポートからのトラフィックをリダイレクトして、標準ポートで Web サービスにアクセスできるようにする

リダイレクトメカニズムを使用すると、ユーザーが URL でポートを指定しなくても、非標準ポートで内部的に実行される Web サービスにアクセスできるようになります。その結果、URL はよりシンプルになり、ブラウジングエクスペリエンスが向上します。一方で、非標準ポートは依然として内部で、または特定の要件のために使用されます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 次の内容を含む **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルを作成します。

```
net.ipv4.ip_forward=1
```

この設定によって、カーネルでの IP 転送が有効になります。

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** ファイルから設定をロードします。

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. NAT リダイレクトルールを作成します。

```
# firewall-cmd --zone=public --add-forward-
port=port=<standard_port>:proto=tcp:toport=<non_standard_port> --permanent
```

上記のコマンドは、次の設定で NAT リダイレクトルールを定義します。

- **--zone=public** - ルールを設定するファイアウォールゾーン。必要なゾーンに合わせて調整できます。
- **--add-forward-port=port=<non_standard_port>** - 着信トラフィックを最初に受信するソースポートを使用したポート転送 (リダイレクト) ルールを追加することを示すオプション。
- **proto=tcp** - TCP トラフィックをリダイレクトすることを示すプロトコル。
- **toport=<standard_port>** - 着信トラフィックがソースポートで受信された後にリダイレクトされる宛先ポート。
- **--permanent** - 再起動後もルールを永続化するオプション。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- 使用したファイアウォールゾーンのリダイレクトルールを確認します。

```
# firewall-cmd --list-forward-ports
port=8080:proto=tcp:toport=80:toaddr=
```

あるいは、対応する XML 設定ファイルを表示します。

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
```

```
  <description>For use in public areas. You do not trust the other computers on networks to
not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="8080" protocol="tcp" to-port="80"/>
  <forward/>
</zone>
```

関連情報

- ランタイム時のカーネルパラメーターの設定
- **firewall-cmd(1)** man ページ

1.12. ICMP リクエストの管理

Internet Control Message Protocol (**ICMP**) は、テスト、トラブルシューティング、診断のために、さまざまなネットワークデバイスによって使用されるサポート対象のプロトコルです。**ICMP** は、システム間でデータを交換するのに使用されていないため、TCP、UDP などの転送プロトコルとは異なります。

ICMP メッセージ (特に **echo-request** および **echo-reply**) を利用して、ネットワークに関する情報を明らかにし、その情報をさまざまな不正行為に悪用することが可能です。したがって、**firewalld** は、ネットワーク情報を保護するため、**ICMP** リクエストを制御できます。

1.12.1. ICMP フィルタリングの設定

ICMP フィルタリングを使用すると、ファイアウォールでシステムへのアクセスを許可または拒否する ICMP のタイプとコードを定義できます。ICMP のタイプとコードは、ICMP メッセージの特定のカテゴリーとサブカテゴリーです。

ICMP フィルタリングは、たとえば次の分野で役立ちます。

- セキュリティーの強化 - 潜在的に有害な ICMP のタイプとコードをブロックして、攻撃対象領域を縮小します。
- ネットワークパフォーマンス - 必要な ICMP タイプのみを許可してネットワークパフォーマンスを最適化し、過剰な ICMP トラフィックによって引き起こされる潜在的なネットワーク輻輳を防ぎます。
- トラブルシューティングの制御 - ネットワークのトラブルシューティングに不可欠な ICMP 機能を維持し、潜在的なセキュリティーリスクとなる ICMP タイプをブロックします。

前提条件

- **firewalld** サービスが実行している。

手順

1. 利用可能な ICMP のタイプとコードをリスト表示します。

```
# firewall-cmd --get-icmptypes
address-unreachable bad-header beyond-scope communication-prohibited destination-
unreachable echo-reply echo-request failed-policy fragmentation-needed host-precedence-
```

```
violation host-prohibited host-redirect host-unknown host-unreachable
...
```

この事前定義されたリストから、許可またはブロックする ICMP のタイプとコードを選択します。

2. 特定の ICMP タイプを次の方法でフィルタリングします。

   - 許可する ICMP タイプ:

     ```
     # firewall-cmd --zone=<target-zone> --remove-icmp-block=echo-request --permanent
     ```

     このコマンドは、エコーリクエスト ICMP タイプに対する既存のブロックルールを削除します。

   - ブロックする ICMP タイプ:

     ```
     # firewall-cmd --zone=<target-zone> --add-icmp-block=redirect --permanent
     ```

     このコマンドは、リダイレクトメッセージ ICMP タイプがファイアウォールによって確実にブロックされるようにします。

3. ファイアウォール設定をリロードして、変更を適用します。

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

検証

- フィルタリングルールが有効であることを確認します。

  ```
  # firewall-cmd --list-icmp-blocks
  redirect
  ```

  コマンド出力には、許可またはブロックした ICMP のタイプとコードが表示されます。

関連情報

- **firewall-cmd(1)** man ページ

1.13. FIREWALLD を使用した IP セットの設定および制御

IP セットは、より柔軟かつ効率的にファイアウォールのルールを管理するために、IP アドレスとネットワークをセットにグループ化する RHEL 機能です。

IP セットは、たとえば次のようなシナリオで役立ちます。

- 大きな IP アドレスリストを処理する場合
- これらの大きな IP アドレスリストに動的更新を実装する場合
- カスタムの IP ベースポリシーを作成して、ネットワークのセキュリティーと制御を強化する場合

警告

Red Hat では、**firewall-cmd** コマンドを使用して IP セットを作成および管理することを推奨します。

1.13.1. IP セットを使用した許可リストの動的更新の設定

ほぼリアルタイムで更新を行うことで、予測不可能な状況でも IP セット内の特定の IP アドレスまたは IP アドレス範囲を柔軟に許可できます。これらの更新は、セキュリティー脅威の検出やネットワーク動作の変化など、さまざまなイベントによってトリガーされます。通常、このようなソリューションでは自動化を活用して手動処理を減らし、素早く状況に対応することでセキュリティーを向上させます。

前提条件

- **firewalld** サービスが実行している。

手順

1. 分かりやすい名前で IP セットを作成します。

```
# firewall-cmd --permanent --new-ipset=allowlist --type=hash:ip
```

この **allowlist** という新しい IP セットには、ファイアウォールで許可する IP アドレスが含まれています。

2. IP セットに動的更新を追加します。

```
# firewall-cmd --permanent --ipset=allowlist --add-entry=198.51.100.10
```

この設定により、新しく追加した、ネットワークトラフィックを渡すことがファイアウォールにより許可される IP アドレスで、**allowlist** の IP セットが更新されます。

3. 先に作成した IP セットを参照するファイアウォールのルールを作成します。

```
# firewall-cmd --permanent --zone=public --add-source=ipset:allowlist
```

このルールがない場合、IP セットはネットワークトラフィックに影響を与えません。デフォルトのファイアウォールポリシーが優先されます。

4. ファイアウォール設定をリロードして、変更を適用します。

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

検証

1. すべての IP セットをリスト表示します。

```
# firewall-cmd --get-ipsets
allowlist
```

2. アクティブなルールをリスト表示します。

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s1
  sources: ipset:allowlist
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  ...
```

コマンドライン出力の **sources** セクションでは、どのトラフィックの発信元 (ホスト名、インターフェイス、IP セット、サブネットなど) が、特定のファイアウォールゾーンへのアクセスを許可または拒否されているかについての洞察が得られます。上記の場合、**allowlist** IP セットに含まれる IP アドレスが、**public** ゾーンのファイアウォールを通してトラフィックを渡すことが許可されています。

3. IP セットの内容を調べます。

```
# cat /etc/firewalld/ipsets/allowlist.xml
<?xml version="1.0" encoding="utf-8"?>
<ipset type="hash:ip">
  <entry>198.51.100.10</entry>
</ipset>
```

次のステップ

- スクリプトまたはセキュリティーユーティリティーを使用して脅威インテリジェンスのフィードを取得し、それに応じて **allowlist** を自動的に更新します。

関連情報

- **firewall-cmd(1)** man ページ

1.14. リッチルールの優先度設定

デフォルトでは、リッチルールはルールアクションに基づいて設定されます。たとえば、**許可** ルールよりも **拒否** ルールが優先されます。リッチルールで **priority** パラメーターを使用すると、管理者はリッチルールとその実行順序をきめ細かく制御できます。**priority** パラメーターを使用すると、ルールはまず優先度の値によって昇順にソートされます。多くのルールが同じ **priority** を持つ場合、ルールの順序はルールアクションによって決まります。アクションも同じである場合、順序は定義されない可能性があります。

1.14.1. priority パラメーターを異なるチェーンにルールを整理する方法

リッチルールの **priority** パラメーターは、**-32768** から **32767** までの任意の数値に設定でき、数値が小さいほど優先度が高くなります。

firewalld サービスは、優先度の値に基づいて、ルールを異なるチェーンに整理します。

- 優先度が 0 未満 - ルールは **_pre** 接尾辞が付いたチェーンにリダイレクトされます。

- 優先度が 0 を超える - ルールは **_post** 接尾辞が付いたチェーンにリダイレクトされます。
- 優先度が 0 - アクションに基づいて、ルールは、**_log**、**_deny**、または **_allow** のアクションを使用してチェーンにリダイレクトされます。

このサブチェーンでは、**firewalld** は優先度の値に基づいてルールを分類します。

1.14.2. リッチルールの優先度の設定

以下は、**priority** パラメーターを使用して、他のルールで許可または拒否されていないすべてのトラフィックをログに記録するリッチルールを作成する方法を示しています。このルールを使用して、予期しないトラフィックにフラグを付けることができます。

手順

- 優先度が非常に低いルールを追加して、他のルールと一致していないすべてのトラフィックをログに記録します。

```
# firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"'
```

このコマンドでは、ログエントリーの数を、毎分 **5** に制限します。

検証

- 前の手順のコマンドで作成した **nftables** ルールを表示します。

```
# nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
  chain filter_IN_public_post {
    log prefix "UNEXPECTED: " limit rate 5/minute
  }
}
```

1.15. ファイアウォールロックダウンの設定

ローカルのアプリケーションやサービスは、**root** で実行していれば、ファイアウォール設定を変更できます (たとえば **libvirt**)。管理者は、この機能を使用してファイアウォール設定をロックし、すべてのアプリケーションでファイアウォール変更を要求できなくするか、ロックダウンの許可リストに追加されたアプリケーションのみがファイアウォール変更を要求できるようにすることが可能になります。ロックダウン設定はデフォルトで無効になっています。これを有効にすると、ローカルのアプリケーションやサービスによるファイアウォールへの望ましくない設定変更を確実に防ぐことができます。

1.15.1. CLI を使用したロックダウンの設定

コマンドラインでロックダウン機能を有効または無効にすることができます。

手順

1. ロックダウンが有効かどうかをクエリーするには、以下を実行します。

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

2. 次のいずれかの方法でロックダウン設定を管理します。

   - ロックダウンを有効にする場合:

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

   - ロックダウンを無効にする場合:

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

1.15.2. ロックダウン許可リスト設定ファイルの概要

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

許可リスト設定ファイルは **/etc/firewalld/** ディレクトリーに保存されます。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
  <command name="/usr/bin/python3 -s /usr/bin/firewall-config"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

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

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

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

Red Hat Enterprise Linux では、すべてのユーティリティーが **/usr/bin/** ディレクトリーに格納されており、**/bin/** ディレクトリーは **/usr/bin/** ディレクトリーへのシンボリックリンクとなります。つまり、**root** で **firewall-cmd** のパスを実行すると **/bin/firewall-cmd** に対して解決しますが、**/usr/bin/firewall-cmd** が使用できるようになっています。新たなスクリプトは、すべて新しい格納場所を使用する必要があります。ただし、**root** で実行するスクリプトが **/bin/firewall-cmd** へのパスを使用するようになっているのであれば、これまでは **root** 以外のユーザーにのみ使用されていた **/usr/bin/firewall-cmd** パスに加え、このコマンドのパスも許可リストに追加する必要があります。

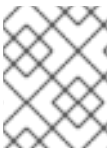
コマンドの名前属性の最後にある * は、その名前で始まるすべてのコマンドが一致することを意味します。*がなければ、コマンドと引数が完全に一致する必要があります。

1.16. FIREWALLD ゾーン内の異なるインターフェイスまたはソース間でのトラフィック転送の有効化

ゾーン内転送は、**firewalld** ゾーン内のインターフェイスまたはソース間のトラフィック転送を可能にする **firewalld** 機能です。

1.16.1. ゾーン内転送と、デフォルトのターゲットが ACCEPT に設定されているゾーンの違い

ゾーン内転送を有効にすると、1 つの **firewalld** ゾーン内のトラフィックは、あるインターフェイスまたはソースから別のインターフェイスまたはソースに流れることができます。ゾーンは、インターフェイスおよびソースの信頼レベルを指定します。信頼レベルが同じ場合、トラフィックは同じゾーン内に留まります。



注記

firewalld のデフォルトゾーンでゾーン内転送を有効にすると、現在のデフォルトゾーンに追加されたインターフェイスおよびソースにのみ適用されます。

firewalld は、異なるゾーンを使用して着信トラフィックと送信トラフィックを管理します。各ゾーンには独自のルールと動作のセットがあります。たとえば、**trusted** ゾーンでは、転送されたトラフィックがデフォルトですべて許可されます。

他のゾーンでは、異なるデフォルト動作を設定できます。標準ゾーンでは、ゾーンのターゲットが **default** に設定されている場合、転送されたトラフィックは通常デフォルトで破棄されます。

ゾーン内の異なるインターフェイスまたはソース間でトラフィックを転送する方法を制御するには、ゾーンのターゲットを理解し、それに応じてゾーンのターゲットを設定する必要があります。

1.16.2. ゾーン内転送を使用したイーサネットと Wi-Fi ネットワーク間でのトラフィックの転送

ゾーン内転送を使用して、同じ **firewalld** ゾーン内のインターフェイスとソース間のトラフィックを転送することができます。この機能には次の利点があります。

- 有線デバイスと無線デバイスの間のシームレスな接続性 (**enp1s0** に接続されたイーサネットネットワークと **wlp0s20** に接続された Wi-Fi ネットワークの間でトラフィックを転送可能)
- 柔軟な作業環境のサポート
- ネットワーク内の複数のデバイスまたはユーザーがアクセスして使用できる共有リソース (プリンター、データベース、ネットワーク接続ストレージなど)
- 効率的な内部ネットワーク (スムーズな通信、レイテンシーの短縮、リソースへのアクセス性など)

この機能は、個々の **firewalld** ゾーンに対して有効にすることができます。

手順

1. カーネルでパケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2. ゾーン内転送を有効にするインターフェイスが **internal** ゾーンにのみ割り当てられていることを確認します。

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

3. 現在、インターフェイスが **internal** 以外のゾーンに割り当てられている場合は、以下のように再割り当てします。

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

4. **enp1s0** および **wlp0s20** インターフェイスを **internal** ゾーンに追加します。

```
# firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5. ゾーン内転送を有効にします。

```
# firewall-cmd --zone=internal --add-forward
```

検証

以下の検証手順では、**nmap-ncat** パッケージが両方のホストにインストールされている必要があります。

1. ゾーン転送を有効にしたホストの **enp1s0** インターフェイスと同じネットワーク上にあるホストにログインします。
2. **ncat** で echo サービスを起動し、接続をテストします。

```
# ncat -e /usr/bin/cat -l 12345
```

3. **wlp0s20** インターフェイスと同じネットワークにあるホストにログインします。
4. **enp1s0** と同じネットワークにあるホスト上で実行している echo サーバーに接続します。

```
# ncat <other_host> 12345
```

5. 何かを入力して **Enter** を押します。テキストが返送されることを確認します。

関連情報

- **firewalld.zones(5)** の man ページ

1.17. RHEL システムロールを使用した FIREWALLD の設定

firewall RHEL システムロールを使用すると、一度に複数のクライアントに **firewalld** サービスを設定できます。この解決策は以下のとおりです。

- 入力設定が効率的なインターフェイスを提供する。
- 目的の **firewalld** パラメーターを 1 か所で保持する。

コントロールノードで **firewall** ロールを実行すると、RHEL システムロールが **firewalld** パラメーターを管理対象ノードに即座に適用し、再起動後もパラメーターを維持します。

1.17.1. RHEL システムロール firewall の概要

RHEL システムロールは、Ansible 自動化ユーティリティーのコンテンツセットです。このコンテンツは、Ansible 自動化ユーティリティーとともに、複数のシステムをリモートで管理するための一貫した設定インターフェイスを提供します。

RHEL システムロールの **rhel-system-roles.firewall** ロールが、**firewalld** サービスの自動設定のために導入されました。**rhel-system-roles** パッケージに、この RHEL システムロールとリファレンスドキュメントが含まれています。

1 つ以上のシステムに **firewalld** パラメーターを自動的に適用するには、Playbook で **firewall** RHEL システムロール変数を使用します。Playbook は、テキストベースの YAML 形式で記述された 1 つ以上のプレイのリストです。

インベントリーファイルを使用して、Ansible が設定するシステムセットを定義できます。

firewall ロールを使用すると、以下のような異なる **firewalld** パラメーターを設定できます。

- ゾーン。
- パケットが許可されるサービス。
- ポートへのトラフィックアクセスの付与、拒否、または削除。
- ゾーンのポートまたはポート範囲の転送。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリー
- Playbook の使用
- インベントリーの構築方法

1.17.2. firewall RHEL システムロールを使用した firewalld 設定のリセット

firewall RHEL システムロールを使用すると、**firewalld** 設定をデフォルトの状態にリセットできます。変数リストに **previous:replaced** パラメーターを追加すると、RHEL システムロールが既存のユーザー定義の設定をすべて削除し、**firewalld** をデフォルトにリセットします。**previous:replaced** パラメーターを他の設定と組み合わせると、**firewall** ロールは新しい設定を適用する前に既存の設定をすべて削除します。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
    - name: Reset firewalld
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - previous: replaced
```

2. Playbook の構文を検証します。

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

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

3. Playbook を実行します。

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

検証

- 管理対象ノードで **root** として次のコマンドを実行し、すべてのゾーンを確認します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリー

1.17.3. firewall RHEL システムロールを使用して、firewalld の着信トラフィックをあるローカルポートから別のローカルポートに転送する

firewall ロールを使用すると、複数の管理対象ホストで設定が永続化されるので **firewalld** パラメーターをリモートで設定できます。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Forward incoming traffic on port 8080 to 443
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. Playbook の構文を検証します。

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

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

3. Playbook を実行します。

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

検証

- 管理対象ホストで、**firewalld** 設定を表示します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリー

1.17.4. firewall RHEL システムロールを使用して、firewalld のポートを管理

firewall RHEL システムロールを使用して、着信トラフィック用のローカルファイアウォールのポートを開閉し、再起動後も新しい設定を維持できます。たとえば、HTTPS サービスの着信トラフィックを許可するようにデフォルトゾーンを設定できます。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Allow incoming HTTPS traffic to the local host
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - port: 443/tcp
            service: http
            state: enabled
            runtime: true
            permanent: true
```

   permanent: true オプションを使用すると、再起動後も新しい設定が維持されます。

2. Playbook の構文を検証します。

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

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

3. Playbook を実行します。

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

検証

- 管理対象ノードで、**HTTPS** サービスに関連付けられた **443/tcp** ポートが開いていることを確認します。

```
# firewall-cmd --list-ports
443/tcp
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリー

1.17.5. firewall RHEL システムロールを使用した firewalld DMZ ゾーンの設定

システム管理者は、**firewall** RHEL システムロールを使用して、 **enp1s0** インターフェイス上に **dmz** ゾーンを設定し、ゾーンへの **HTTPS** トラフィックを許可できます。これにより、外部ユーザーが Web サーバーにアクセスできるようにします。

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - zone: dmz
            interface: enp1s0
            service: https
            state: enabled
            runtime: true
            permanent: true
```

2. Playbook の構文を検証します。

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

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

3. Playbook を実行します。

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

検証

- 管理ノードで、**dmz** ゾーンに関する詳細情報を表示します。

```
# firewall-cmd --zone=dmz --list-all
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
```

```
forward-ports:
source-ports:
icmp-blocks:
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリー

第2章 NFTABLES の使用

nftables フレームワークはパケットを分類し、**iptables**、**ip6tables**、**arptables**、**ebtables**、および **ipset** ユーティリティーの後継です。利便性、機能、パフォーマンスにおいて、以前のパケットフィルタリングツールに多くの改良が追加されました。以下に例を示します。

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

nftables フレームワークは、テーブルを使用してチェーンを保存します。このチェーンには、アクションを実行する個々のルールが含まれます。**nft** ユーティリティーは、以前のパケットフィルタリングフレームワークのツールをすべて置き換えます。**libmnl** ライブラリーを介して、**nftables** Netlink API との低レベルの対話に **libnftnl** ライブラリーを使用できます。

ルールセット変更が適用されていることを表示するには、**nft list ruleset** コマンドを使用します。これらのユーティリティーはテーブル、チェーン、ルール、セット、およびその他のオブジェクトを **nftables** ルールセットに追加するため、**nft flush ruleset** コマンドなどの **nftables** ルールセット操作は、**iptables** コマンドを使用してインストールされたルールセットに影響を与える可能性があることに注意してください。

2.1. IPTABLES から NFTABLES への移行

ファイアウォール設定が依然として **iptables** ルールを使用している場合は、**iptables** ルールを **nftables** に移行できます。

重要

ipset パッケージおよび **iptables-nft** パッケージは、Red Hat Enterprise Linux 9 で非推奨になりました。これには、**iptables**、**ip6tables**、**arptables**、および **ebtables** ユーティリティーなどの **nft-variants** の非推奨化が含まれます。以前のバージョンの RHEL からアップグレードしたなど、このツールのいずれかを使用している場合は、Red Hat は、**nftables** パッケージが提供する **nft** コマンドラインツールへの移行を推奨します。

2.1.1. 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 の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

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

2.1.2. iptables および ip6tables ルールセットの nftables への変換

iptables-restore-translate ユーティリティーおよび **ip6tables-restore-translate** ユーティリティーを使用して、**iptables** および **ip6tables** ルールセットを **nftables** に変換します。

前提条件

- **nftables** パッケージおよび **iptables** パッケージがインストールされている。
- システムに **iptables** ルールおよび **ip6tables** ルールが設定されている。

手順

1. **iptables** ルールおよび **ip6tables** ルールをファイルに書き込みます。

```
# iptables-save >/root/iptables.dump
# ip6tables-save >/root/ip6tables.dump
```

2. ダンプファイルを **nftables** 命令に変換します。

```
# iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-from-iptables.nft
# ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-from-ip6tables.nft
```

3. 必要に応じて、生成された **nftables** ルールを手動で更新して、確認します。
4. **nftables** サービスが生成されたファイルをロードできるようにするには、以下を **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5. **iptables** サービスを停止し、無効にします。

```
# systemctl disable --now iptables
```

カスタムスクリプトを使用して **iptables** ルールを読み込んだ場合は、スクリプトが自動的に開始されなくなったことを確認し、再起動してすべてのテーブルをフラッシュします。

6. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

- **nftables** ルールセットを表示します。

```
# nft list ruleset
```

関連情報

- システムの起動時に nftables ルールの自動読み込み

2.1.3. 単一の iptables および ip6tables ルールセットの nftables への変換

Red Hat Enterprise Linux は、**iptables** ルールまたは **ip6tables** ルールを、**nftables** で同等のルールに変換する **iptables-translate** ユーティリティーおよび **ip6tables-translate** ユーティリティーを提供します。

前提条件

- **nftables** パッケージがインストールされている。

手順

- 以下のように、**iptables** または **ip6tables** の代わりに **iptables-translate** ユーティリティーまたは **ip6tables-translate** ユーティリティーを使用して、対応する **nftables** ルールを表示します。

```
# iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

拡張機能によっては変換機能がない場合もあります。このような場合には、ユーティリティーは、以下のように、前に **#** 記号が付いた未変換ルールを出力します。

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

関連情報

- **iptables-translate --help**

2.1.4. 一般的な iptables コマンドと nftables コマンドの比較

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

- すべてのルールをリスト表示します。

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

- 特定のテーブルおよびチェーンをリスト表示します。

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

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

firewalld によって生成されたルールの一覧表示:

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

2.2. NFTABLES スクリプトの作成および実行

nftables フレームワークを使用する主な利点は、スクリプトの実行がアトミックであることです。つまり、システムがスクリプト全体を適用するか、エラーが発生した場合には実行を阻止することを意味します。これにより、ファイアウォールは常に一貫した状態になります。

さらに、**nftables** スクリプト環境を使用すると、次のことができます。

- コメントの追加
- 変数の定義
- 他のルールセットファイルの組み込み

nftables パッケージをインストールすると、Red Hat Enterprise Linux が自動的に ***.nft** スクリプトを **/etc/nftables/** ディレクトリーに作成します。このスクリプトは、さまざまな目的でテーブルと空のチェーンを作成するコマンドが含まれます。

2.2.1. 対応している nftables スクリプトの形式

nftables スクリプト環境では、次の形式でスクリプトを記述できます。

- **nft list ruleset** コマンドと同じ形式でルールセットが表示されます。

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

# Flush the rule set
flush ruleset

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

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

- **nft** コマンドと同じ構文:

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

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table

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

# Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

2.2.2. nftables スクリプトの実行

nftables スクリプトは、**nft** ユーティリティーに渡すか、スクリプトを直接実行することで実行できます。

手順

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

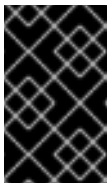
```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

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

  a. 1回だけ実行する場合:

  i. スクリプトが以下のシバンシーケンスで始まることを確認します。

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



重要

-f パラメーターを省略すると、**nft** ユーティリティーはスクリプトを読み込まず、**Error: syntax error, unexpected newline, expecting string** のように表示されます。

  ii. オプション: スクリプトの所有者を **root** に設定します。

```
# chown root /etc/nftables/<example_firewall_script>.nft
```

iii. 所有者のスクリプトを実行ファイルに変更します。

```
# chmod u+x /etc/nftables/<example_firewall_script>.nft
```

b. スクリプトを実行します。

```
# /etc/nftables/<example_firewall_script>.nft
```

出力が表示されない場合は、システムがスクリプトを正常に実行します。

重要

nft はスクリプトを正常に実行しますが、ルールの配置やパラメーター不足、またはスクリプト内のその他の問題により、ファイアウォールが期待通りの動作を起こさない可能性があります。

関連情報

- **chown(1)** の man ページ
- **chmod(1)** の man ページ
- システムの起動時に nftables ルールの自動読み込み

2.2.3. nftables スクリプトでコメントの使用

nftables スクリプト環境は、**#** 文字の右側から行末までのすべてをコメントとして解釈します。

コメントは、行の先頭またはコマンドの横から開始できます。

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

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

2.2.4. nftables スクリプトでの変数の使用

nftables スクリプトで変数を定義するには、**define** キーワードを使用します。シングル値および匿名セットを変数に保存できます。より複雑なシナリオの場合は、セットまたは決定マップを使用します。

値を 1つ持つ変数

以下の例は、値が **enp1s0** の **INET_DEV** という名前の変数を定義します。

```
define INET_DEV = enp1s0
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

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

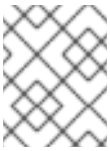
匿名セットを含む変数

以下の例では、匿名セットを含む変数を定義します。

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```



注記

中括弧は、変数がセットを表していることを示すため、ルールで使用する場合は、特別なセマンティクスを持ちます。

関連情報

- nftables コマンドでのセットの使用
- nftables コマンドにおける決定マップの使用

2.2.5. nftables スクリプトへのファイルの追加

nftables スクリプト環境では、**include** ステートメントを使用して他のスクリプトを含めることができます。

絶対パスまたは相対パスのないファイル名のみを指定すると、**nftables** には、デフォルトの検索パスのファイルが含まれます。これは、Red Hat Enterprise Linux では **/etc** に設定されています。

例2.1 デフォルト検索ディレクトリーからのファイルを含む

デフォルトの検索ディレクトリーからファイルを指定するには、次のコマンドを実行します。

```
include "example.nft"
```

例2.2 ディレクトリーの *.nft ファイルをすべて含む

***.nft** で終わるすべてのファイルを **/etc/nftables/rulesets/** ディレクトリーに保存するには、次のコマンドを実行します。

```
include "/etc/nftables/rulesets/*.nft"
```

include ステートメントは、ドットで始まるファイルに一致しないことに注意してください。

関連情報

- **nft(8)** の man ページの **Include files** セクション

2.2.6. システムの起動時に nftables ルールの自動読み込み

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

前提条件

- **nftables** スクリプトは、**/etc/nftables/** ディレクトリーに保存されます。

手順

1. **/etc/sysconfig/nftables.conf** ファイルを編集します。
   - **nftables** パッケージのインストールで **/etc/nftables/** に作成された ***.nft** スクリプトを変更した場合は、これらのスクリプトの **include** ステートメントのコメントを解除します。
   - 新しいスクリプトを作成した場合は、**include** ステートメントを追加してこれらのスクリプトを含めます。たとえば、**nftables** サービスの起動時に **/etc/nftables/example.nft** スクリプトを読み込むには、以下を追加します。

     ```
     include "/etc/nftables/_example_.nft"
     ```

2. オプション: **nftables** サービスを開始して、システムを再起動せずにファイアウォールルールを読み込みます。

   ```
   # systemctl start nftables
   ```

3. **nftables** サービスを有効にします。

   ```
   # systemctl enable nftables
   ```

関連情報

- 対応している nftables スクリプトの形式

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

nftables ルールセットを表示して管理できます。

2.3.1. nftables テーブルの基本

nftables のテーブルは、チェーン、ルール、セットなどのオブジェクトを含む名前空間です。

各テーブルにはアドレスファミリーが割り当てられている必要があります。アドレスファミリーは、このテーブルが処理するパケットタイプを定義します。テーブルを作成する際に、以下のいずれかのアドレスファミリーを設定できます。

- **ip**:IPv4 パケットだけに一致します。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 パケットだけに一致します。
- **inet**:IPv4 パケットと IPv6 パケットの両方に一致します。
- **arp**:IPv4 アドレス解決プロトコル (ARP) パケットに一致します。

- **bridge**:ブリッジデバイスを通過するパケットに一致します。
- **netdev**:ingress からのパケットに一致します。

テーブルを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
}
```

- シェルスクリプトで、以下を使用します。

```
nft add table <table_address_family> <table_name>
```

2.3.2. nftables チェーンの基本

テーブルは、ルールのコンテナーであるチェーンで構成されます。次の 2 つのルールタイプが存在します。

- **ベースチェーン**:ネットワークスタックからのパケットのエントリーポイントとしてベースチェーンを使用できます。
- **通常のチェーン**:**jump** ターゲットとして通常のチェーンを使用し、ルールをより適切に整理できます。

ベースチェーンをテーブルに追加する場合に使用する形式は、ファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority>
    policy <policy> ;
  }
}
```

- シェルスクリプトで、以下を使用します。

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook <hook> priority <priority> \; policy <policy> \; }
```

  シェルがセミコロンをコマンドの最後として解釈しないようにするには、セミコロンの前にエスケープ文字 \ を配置します。

どちらの例でも、**ベースチェーン** を作成します。**通常のチェーン** を作成する場合、中括弧内にパラメーターを設定しないでください。

チェーンタイプ
チェーンタイプとそれらを使用できるアドレスファミリーとフックの概要を以下に示します。

型	アドレスファミリー	フック	説明
filter	all	all	標準のチェーンタイプ
nat	**ip**、**ip6**、**inet**	**prerouting**、**input**、**output**、**postrouting**	このタイプのチェーンは、接続追跡エントリーに基づいてネイティブアドレス変換を実行します。最初のパケットのみがこのチェーンタイプをトラバースします。
ルート	**ip**、**ip6**	**出力 (output)**	このチェーンタイプを通過する許可済みパケットは、IP ヘッダーの関連部分が変更された場合に、新しいルートルックアップを引き起こします。

チェーンの優先度
priority パラメーターは、パケットが同じフック値を持つチェーンを通過する順序を指定します。このパラメーターは、整数値に設定することも、標準の priority 名を使用することもできます。

以下のマトリックスは、標準的な priority 名とその数値の概要、それらを使用できるファミリーおよびフックの概要です。

テキストの値	数値	アドレスファミリー	フック
raw	**-300**	**ip**、**ip6**、**inet**	all
mangle	**-150**	**ip**、**ip6**、**inet**	all
dstnat	**-100**	**ip**、**ip6**、**inet**	**prerouting**
	-300	**bridge**	**prerouting**
filter	**0**	**ip**、**ip6**、**inet**、**arp**、**netdev**	all
	-200	**bridge**	all
security	**50**	**ip**、**ip6**、**inet**	all
srcnat	**100**	**ip**、**ip6**、**inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**出力 (output)**

チェーンポリシー
チェーンポリシーは、このチェーンのルールでアクションが指定されていない場合に、**nftables** がパケットを受け入れるかドロップするかを定義します。チェーンには、以下のいずれかのポリシーを設定できます。

- **accept** (デフォルト)
- **drop**

2.3.3. nftables ルールの基本

ルールは、このルールを含むチェーンを渡すパケットに対して実行するアクションを定義します。ルールに一致する式も含まれる場合、**nftables** は、以前の式がすべて適用されている場合にのみアクションを実行します。

チェーンにルールを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority> ; policy <policy> ;
      <rule>
  }
}
```

- シェルスクリプトで、以下を使用します。

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

このシェルコマンドは、チェーンの最後に新しいルールを追加します。チェーンの先頭にルールを追加する場合は、**nft add** の代わりに **nft insert** コマンドを使用します。

2.3.4. nft コマンドを使用したテーブル、チェーン、ルールの管理

コマンドラインまたはシェルスクリプトで **nftables** ファイアウォールを管理するには、**nft** ユーティリティーを使用します。

重要

この手順のコマンドは通常のワークフローを表しておらず、最適化されていません。この手順では、**nft** コマンドを使用して、一般的なテーブル、チェーン、およびルールを管理する方法を説明します。

手順

1. テーブルが IPv4 パケットと IPv6 パケットの両方を処理できるように、**inet** アドレスファミリーを使用して **nftables_svc** という名前のテーブルを作成します。

```
# nft add table inet nftables_svc
```

2. 受信ネットワークトラフィックを処理する **INPUT** という名前のベースチェーンを **inet nftables_svc** テーブルに追加します。

```
# nft add chain inet nftables_svc INPUT { type filter hook input priority filter \; policy accept \; }
```

シェルがセミコロンをコマンドの最後として解釈しないようにするには、\ 文字を使用してセミコロンをエスケープします。

3. **INPUT** チェーンにルールを追加します。たとえば、**INPUT** チェーンの最後のルールとして、ポート 22 および 443 で着信 TCP トラフィックを許可し、Internet Control Message Protocol (ICMP)ポートに到達できないメッセージで他の着信トラフィックを拒否します。

```
# nft add rule inet nftables_svc INPUT tcp dport 22 accept
# nft add rule inet nftables_svc INPUT tcp dport 443 accept
# nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

ここで示されたように **nft add rule** コマンドを実行すると、**nft** はコマンド実行と同じ順序でルールをチェーンに追加します。

4. ハンドルを含む現在のルールセットを表示します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

5. ハンドル 3 で既存ルールの前にルールを挿入します。たとえば、ポート 636 で TCP トラフィックを許可するルールを挿入するには、以下を入力します。

```
# nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6. ハンドル 3 で、既存ルールの後ろにルールを追加します。たとえば、ポート 80 で TCP トラフィックを許可するルールを追加するには、以下を入力します。

```
# nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7. ハンドルでルールセットを再表示します。後で追加したルールが指定の位置に追加されていることを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    tcp dport 80 accept # handle 6
    reject # handle 4
  }
}
```

8. ハンドル 6 でルールを削除します。

```
# nft delete rule inet nftables_svc INPUT handle 6
```

ルールを削除するには、ハンドルを指定する必要があります。

9. ルールセットを表示し、削除されたルールがもう存在しないことを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

10. **INPUT** チェーンから残りのルールをすべて削除します。

```
# nft flush chain inet nftables_svc INPUT
```

11. ルールセットを表示し、**INPUT** チェーンが空であることを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
    type filter hook input priority filter; policy accept
  }
}
```

12. **INPUT** チェーンを削除します。

```
# nft delete chain inet nftables_svc INPUT
```

このコマンドを使用して、まだルールが含まれているチェーンを削除することもできます。

13. ルールセットを表示し、**INPUT** チェーンが削除されたことを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
}
```

14. **nftables_svc** テーブルを削除します。

```
# nft delete table inet nftables_svc
```

このコマンドを使用して、まだルールが含まれているテーブルを削除することもできます。

注記

ルールセット全体を削除するには、個別のコマンドですべてのルール、チェーン、およびテーブルを手動で削除するのではなく、**nft flush ruleset** コマンドを使用します。

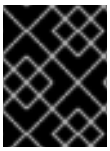
関連情報

nft(8) の man ページ

2.4. NFTABLES を使用した NAT の設定

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

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



重要

iifname パラメーターおよび **oifname** パラメーターでは実インターフェイス名のみを使用でき、代替名 (**altname**) には対応していません。

2.4.1. NAT タイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダー (ISP) は、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードと SNAT は互いに非常に似ています。相違点は次のとおりです。

- マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。
- SNAT は、パケットのソース IP アドレスを指定された IP に設定し、出力インターフェイスの IP アドレスを動的に検索しません。そのため、SNAT の方がマスカレードよりも高速です。出力インターフェイスが固定 IP アドレスを使用する場合は、SNAT を使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

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

2.4.2. nftables を使用したマスカレードの設定

マスカレードを使用すると、ルーターは、インターフェイスを介して送信されるパケットのソース IP を、インターフェイスの IP アドレスに動的に変更できます。これは、インターフェイスに新しい IP が割り当てられている場合に、**nftables** はソース IP の置き換え時に新しい IP を自動的に使用することを意味します。

ens3 インターフェイスを介してホストから出るパケットの送信元 IP を、**ens3** で設定された IP に置き換えます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

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

重要

prerouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、着信パケット返信に一致するようにこのチェーンが必要になります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **postrouting** チェーンに、**ens3** インターフェイスの出力パケットに一致するルールを追加します。

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

2.4.3. nftables を使用したソース NAT の設定

ルーターでは、ソース NAT (SNAT) を使用して、インターフェイスを介して特定の IP アドレスに送信するパケットの IP を変更できます。次に、ルーターは送信パケットのソース IP を置き換えます。

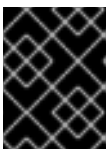
手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

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



重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **ens3** を介した発信パケットのソース IP を **192.0.2.1** に置き換えるルールを **postrouting** チェーンに追加します。

```
# nft add rule nat postrouting oifname "ens3" snat to 192.0.2.1
```

関連情報

- 特定のローカルポートで着信パケットを別のホストに転送

2.4.4. nftables を使用した宛先 NAT の設定

宛先 NAT (DNAT)を使用すると、ルーター上のトラフィックをインターネットから直接アクセスできないホストにリダイレクトできます。

たとえば、DNAT を使用すると、ルーターはポート **80** および **443** に送信された受信トラフィックを、IP アドレス **192.0.2.1** の Web サーバーにリダイレクトします。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

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

重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **prerouting** チェーンに、ルーターの **ens3** インターフェイスのポート **80** および **443** への受信トラフィックを、IP アドレス **192.0.2.1** の Web サーバーにリダイレクトするルールを追加します。

```
# nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4. 環境に応じて、SNAT ルールまたはマスカレードルールを追加して、Web サーバーから返されるパケットのソースアドレスを送信者に変更します。

   a. **ens3** インターフェイスが動的 IP アドレスを使用している場合は、マスカレードルールを追加します。

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

   b. **ens3** インターフェイスが静的 IP アドレスを使用する場合は、SNAT ルールを追加します。たとえば、**ens3** が IP アドレス **198.51.100.1** を使用している場合は、以下のようになります。

```
# nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

関連情報

- NAT タイプ

2.4.5. nftables を使用したリダイレクトの設定

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

たとえば、ローカルホストのポート **22** に送信された着信および転送されたトラフィックを **2222** ポートにリダイレクトすることができます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに **prerouting** チェーンを追加します。

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

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **22** ポートの着信トラフィックを **2222** ポートにリダイレクトするルールを **prerouting** チェーンに追加します。

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

関連情報

- NAT タイプ

2.4.6. nftables を使用したフローテーブルの設定

nftables ユーティリティーは、**netfilter** フレームワークを使用してネットワークトラフィックにネットワークアドレス変換 (NAT) を提供し、高速パス機能ベースの **flowtable** メカニズムを提供してパケット転送を高速化します。

フローテーブルメカニズムには次の機能があります。

- 接続追跡を使用して、従来のパケット転送パスをバイパスします。
- 従来のパケット処理をバイパスすることで、ルーティングテーブルの再参照を回避します。
- TCP および UDP プロトコルでのみ動作します。
- ハードウェアに依存しないソフトウェア高速パスです。

手順

1. **inet** ファミリーの **example-table** テーブルを追加します。

```
# nft add table inet <example-table>
```

2. 優先度タイプとして **ingress** フックと **filter** を含む **example-flowtable** フローテーブルを追加します。

```
# nft add flowtable inet <example-table> <example-flowtable> { hook ingress priority filter \; devices = { enp1s0, enp7s0 } \; }
```

3. **example-forwardchain** フローをパケット処理テーブルからフローテーブルに追加します。

```
# nft add chain inet <example-table> <example-forwardchain> { type filter hook forward priority filter \; }
```

このコマンドは、**forward** フックと **filter** 優先度を備えた **filter** タイプのフローテーブルを追加します。

4. **established** 接続追跡状態を含むルールを追加して、**example-flowtable** フローをオフロードします。

```
# nft add rule inet <example-table> <example-forwardchain> ct state established flow add @<example-flowtable>
```

検証

- **example-table** のプロパティーを確認します。

```
# nft list table inet <example-table>
table inet example-table {
    flowtable example-flowtable {
      hook ingress priority filter
          devices = { enp1s0, enp7s0 }
    }

    chain example-forwardchain {
type filter hook forward priority filter; policy accept;
ct state established flow add @example-flowtable
   }
}
```

関連情報

- **nft(8)** の man ページ

2.5. NFTABLES コマンドでのセットの使用

nftables フレームワークは、セットをネイティブに対応します。たとえば、ルールが複数の IP アドレス、ポート番号、インターフェイス、またはその他の一致基準に一致する必要がある場合など、セットを使用できます。

2.5.1. nftables での匿名セットの使用

匿名セットには、ルールで直接使用する **{ 22, 80, 443 }** などの中括弧で囲まれたコンマ区切りの値が含まれます。IP アドレスやその他の一致基準にも匿名セットを使用できます。

匿名セットの欠点は、セットを変更する場合はルールを置き換える必要があることです。動的なソリューションの場合は、nftables で名前付きセットの使用 で説明されているように名前付きセットを使用します。

前提条件

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

手順

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

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

2. オプション: **example_table** 内の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
  }
}
```

2.5.2. nftables で名前付きセットの使用

nftables フレームワークは、変更可能な名前付きセットに対応します。名前付きセットは、テーブル内の複数のルールで使用できる要素のリストまたは範囲です。匿名セットに対する別の利点として、セットを使用するルールを置き換えることなく、名前付きセットを更新できます。

名前付きセットを作成する場合は、セットに含まれる要素のタイプを指定する必要があります。以下のタイプを設定できます。

- **192.0.2.1** や **192.0.2.0/24** など、IPv4 アドレスまたは範囲を含むセットの場合は **ipv4_addr**。
- **2001:db8:1::1** や **2001:db8:1::1/64** など、IPv6 アドレスまたは範囲を含むセットの場合は **ipv6_addr**。
- **52:54:00:6b:66:42** など、メディアアクセス制御 (MAC) アドレスのリストを含むセットの場合は **ether_addr**。
- **tcp** など、インターネットプロトコルタイプの一覧が含まれるセットの場合は **inet_proto**。
- **ssh** など、インターネットサービスの一覧を含むセットの場合は **inet_service**。
- パケットマークの一覧を含むセットの場合は **mark**。パケットマークは、任意の 32 ビットの正の整数値 (**0** から **2147483647**) にすることができます。

前提条件

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

手順

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

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

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

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

     重要

     シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

2. オプション: セットを使用するルールを作成します。たとえば、次のコマンドは、**example_set** の IPv4 アドレスからのパケットをすべて破棄するルールを、**example_table** の **example_chain** に追加します。

   ```
   # nft add rule inet example_table example_chain ip saddr @example_set drop
   ```

   example_set が空のままなので、ルールには現在影響がありません。

3. IPv4 アドレスを **example_set** に追加します。
   - 個々の IPv4 アドレスを保存するセットを作成する場合は、次のコマンドを実行します。

     ```
     # nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
     ```

   - IPv4 範囲を保存するセットを作成する場合は、次のコマンドを実行します。

     ```
     # nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
     ```

     IP アドレス範囲を指定する場合は、上記の例の **192.0.2.0/24** のように、CIDR (Classless Inter-Domain Routing) 表記を使用することもできます。

2.5.3. 関連情報

- **nft(8)** の man ページの **Sets** セクション

2.6. 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。

2.6.1. nftables での匿名マップの使用

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

匿名マップの欠点は、マップを変更する場合には、ルールを置き換える必要があることです。動的なソリューションの場合は、nftables での名前付きマップの使用 で説明されているように名前付きマップを使用します。

たとえば、匿名マップを使用して、IPv4 プロトコルおよび IPv6 プロトコルの TCP パケットと UDP パケットの両方を異なるチェーンにルーティングし、着信 TCP パケットと UDP パケットを個別にカウントできます。

手順

1. 新しいテーブルを作成します。

```
# nft add table inet example_table
```

2. **example_table** に **tcp_packets** チェーンを作成します。

```
# nft add chain inet example_table tcp_packets
```

3. このチェーンのトラフィックをカウントする **tcp_packets** にルールを追加します。

```
# nft add rule inet example_table tcp_packets counter
```

4. **example_table** で **udp_packets** チェーンを作成します。

```
# nft add chain inet example_table udp_packets
```

5. このチェーンのトラフィックをカウントする **udp_packets** にルールを追加します。

```
# nft add rule inet example_table udp_packets counter
```

6. 着信トラフィックのチェーンを作成します。たとえば、**example_table** に、着信トラフィックをフィルタリングする **incoming_traffic** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7. 匿名マップを持つルールを **incoming_traffic** に追加します。

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

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

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

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
```

```
    counter packets 36379 bytes 2103816
  }

  chain udp_packets {
    counter packets 10 bytes 1559
  }

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

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

2.6.2. nftables での名前付きマップの使用

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

名前付きマップを作成する場合は、要素のタイプを指定する必要があります。

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

たとえば、送信元 IP アドレスに基づいて着信パケットを許可または拒否できます。名前付きマップを使用すると、このシナリオを設定するのに必要なルールは 1 つだけで、IP アドレスとアクションがマップに動的に保存されます。

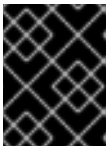
手順

1. テーブルを作成します。たとえば、IPv4 パケットを処理する **example_table** という名前のテーブルを作成するには、次のコマンドを実行します。

```
# nft add table ip example_table
```

2. チェーンを作成します。たとえば、**example_table** に、**example_chain** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```



重要

シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

3. 空のマップを作成します。たとえば、IPv4 アドレスのマッピングを作成するには、次のコマンドを実行します。

```
# nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4. マップを使用するルールを作成します。たとえば、次のコマンドは、両方とも **example_map** で定義されている IPv4 アドレスにアクションを適用するルールを、 **example_table** の **example_chain** に追加します。

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. IPv4 アドレスと対応するアクションを **example_map** に追加します。

```
# nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

以下の例では、IPv4 アドレスのアクションへのマッピングを定義します。上記で作成したルールと組み合わせて、ファイアウォールは **192.0.2.1** からのパケットを許可し、**192.0.2.2** からのパケットを破棄します。

6. オプション: 別の IP アドレスおよび action ステートメントを追加してマップを拡張します。

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. オプション: マップからエントリーを削除します。

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

8. オプション: ルールセットを表示します。

```
# nft list ruleset
table ip example_table {
  map example_map {
    type ipv4_addr : verdict
    elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
  }

  chain example_chain {
    type filter hook input priority filter; policy accept;
    ip saddr vmap @example_map
  }
}
```

2.6.3. 関連情報

- **nft(8)** の man ページの **Maps** セクション

2.7. 以下に例を示します。NFTABLES スクリプトを使用した LAN および DMZ の保護

RHEL ルーターで **nftables** フレームワークを使用して、内部 LAN 内のネットワーククライアントと DMZ の Web サーバーを、インターネットやその他のネットワークからの不正アクセスから保護するファイアウォールスクリプトを作成およびインストールします。

重要

この例はデモ目的専用で、特定の要件があるシナリオを説明しています。

ファイアウォールスクリプトは、ネットワークインフラストラクチャーとセキュリティー要件に大きく依存します。この例を使用して、独自の環境用のスクリプトを作成する際に **nftables** ファイアウォールの概念を理解してください。

2.7.1. ネットワークの状態

この例のネットワークは、以下の条件下にあります。

- ルーターは以下のネットワークに接続されています。
  - インターフェイス **enp1s0**を介したインターネット
  - インターフェイス **enp7s0**を介した内部 LAN
  - **enp8s0**までの DMZ
- ルーターのインターネットインターフェイスには、静的 IPv4 アドレス (**203.0.113.1**) と IPv6 アドレス (**2001:db8:a::1**) の両方が割り当てられています。
- 内部 LAN のクライアントは **10.0.0.0/24** の範囲のプライベート IPv4 アドレスのみを使用します。その結果、LAN からインターネットへのトラフィックには、送信元ネットワークアドレス変換 (SNAT) が必要です。
- 内部 LAN の管理者用 PC は、IP アドレス **10.0.0.100** および **10.0.0.200** を使用します。
- DMZ は、**198.51.100.0/24** および **2001:db8:b::/56** の範囲のパブリック IP アドレスを使用します。
- DMZ の Web サーバーは、IP アドレス **198.51.100.5** および **2001:db8:b::5** を使用します。
- ルーターは、LAN および DMZ 内のホストのキャッシング DNS サーバーとして機能します。

2.7.2. ファイアウォールスクリプトのセキュリティー要件

以下は、サンプルネットワークにおける **nftables** ファイアウォールの要件です。

- ルーターは以下を実行できる必要があります。
  - DNS クエリーを再帰的に解決します。

    - ループバックインターフェイスですべての接続を実行します。
- 内部 LAN のクライアントは以下を実行できる必要があります。
    - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
    - DMZ の HTTPS サーバーにアクセスします。
    - インターネット上の任意の HTTPS サーバーにアクセスします。
- 管理者用の PC は、SSH を使用してルーターと DMZ 内のすべてのサーバーにアクセスできる必要があります。
- DMZ の Web サーバーは以下を実行できる必要があります。
    - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
    - インターネット上の HTTPS サーバーにアクセスして更新をダウンロードします。
- インターネット上のホストは以下を実行できる必要があります。
    - DMZ の HTTPS サーバーにアクセスします。
- さらに、以下のセキュリティー要件が存在します。
    - 明示的に許可されていない接続の試行はドロップする必要があります。
    - ドロップされたパケットはログに記録する必要があります。

2.7.3. ドロップされたパケットをファイルにロギングするための設定

デフォルトでは、**systemd** は、ドロップされたパケットなどのカーネルメッセージをジャーナルに記録します。さらに、このようなエントリーを別のファイルに記録するように **rsyslog** サービスを設定することもできます。ログファイルが無限に大きくならないようにするために、ローテーションポリシーを設定します。

前提条件

- **rsyslog** パッケージがインストールされている。
- **rsyslog** サービスが実行されている。

手順

1. 以下の内容で **/etc/rsyslog.d/nftables.conf** ファイルを作成します。

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

   この設定を使用すると、**rsyslog** サービスはドロップされたパケットを **/var/log/messages** ではなく **/var/log/nftables.log** ファイルに記録します。

2. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

3. サイズが 10 MB を超える場合は、以下の内容で **/etc/logrotate.d/nftables** ファイルを作成し、**/var/log/nftables.log** をローテーションします。

```
/var/log/nftables.log {
  size +10M
  maxage 30
  sharedscripts
  postrotate
    /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
  endscript
}
```

maxage 30 設定は、次のローテーション操作中に **logrotate** が 30 日経過したローテーション済みログを削除することを定義します。

関連情報

- **rsyslog.conf(5)** の man ページ
- **logrotate(8)** の man ページ

2.7.4. nftables スクリプトの作成とアクティブ化

この例は、RHEL ルーターで実行され、DMZ の内部 LAN および Web サーバーのクライアントを保護する **nftables** ファイアウォールスクリプトです。この例で使用されているネットワークとファイアウォールの要件について、詳しくはファイアウォールスクリプトの ネットワークの状態 および ファイアウォールスクリプトのセキュリティー要件 を参照してください。

警告

この **nftables** ファイアウォールスクリプトは、デモ専用です。お使いの環境やセキュリティー要件に適応させて使用してください。

前提条件

- ネットワークは、ネットワークの状態 で説明されているとおりに設定されます。

手順

1. 以下の内容で **/etc/nftables/firewall.nft** スクリプトを作成します。

```
# Remove all rules
flush ruleset


# Table for both IPv4 and IPv6 rules
table inet nftables_svc {

  # Define variables for the interface name
  define INET_DEV = enp1s0
```

```
define LAN_DEV  = enp7s0
define DMZ_DEV  = enp8s0


# Set with the IPv4 addresses of admin PCs
set admin_pc_ipv4 {
  type ipv4_addr
  elements = { 10.0.0.100, 10.0.0.200 }
}


# Chain for incoming trafic. Default policy: drop
chain INPUT {
  type filter hook input priority filter
  policy drop

  # Accept packets in established and related state, drop invalid packets
  ct state vmap { established:accept, related:accept, invalid:drop }

  # Accept incoming traffic on loopback interface
  iifname lo accept

  # Allow request from LAN and DMZ to local DNS server
  iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

  # Allow admins PCs to access the router using SSH
  iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

  # Last action: Log blocked packets
  # (packets that were not accepted in previous rules in this chain)
  log prefix "nft drop IN : "
}


# Chain for outgoing traffic. Default policy: drop
chain OUTPUT {
  type filter hook output priority filter
  policy drop

  # Accept packets in established and related state, drop invalid packets
  ct state vmap { established:accept, related:accept, invalid:drop }

  # Accept outgoing traffic on loopback interface
  oifname lo accept

  # Allow local DNS server to recursively resolve queries
  oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

  # Last action: Log blocked packets
  log prefix "nft drop OUT: "
}


# Chain for forwarding traffic. Default policy: drop
chain FORWARD {
  type filter hook forward priority filter
```

```
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # IPv4 access from LAN and internet to the HTTPS server in the DMZ
    iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp dport
443 accept

    # IPv6 access from internet to the HTTPS server in the DMZ
    iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443 accept

    # Access from LAN and DMZ to HTTPS servers on the internet
    iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

    # Last action: Log blocked packets
    log prefix "nft drop FWD: "
  }


  # Postrouting chain to handle SNAT
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;

    # SNAT for IPv4 traffic from LAN to internet
    iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
  }
}
```

2. **/etc/nftables/firewall.nft** スクリプトを **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/firewall.nft"
```

3. IPv4 転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

1. オプション: **nftables** ルールセットを確認します。

```
# nft list ruleset
...
```

2. ファイアウォールが阻止するアクセスの実行を試みます。たとえば、DMZ から SSH を使用してルーターにアクセスします。

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. ロギング設定に応じて、以下を検索します。

   - ブロックされたパケットの **systemd** ジャーナル:

     ```
     # journalctl -k -g "nft drop"
     Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
     SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
     ```

   - ブロックされたパケットの **/var/log/nftables.log** ファイル:

     ```
     Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
     SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
     ```

2.8. NFTABLES を使用したポート転送の設定

ポート転送を使用すると、管理者は特定の宛先ポートに送信されたパケットを、別のローカルまたはリモートポートに転送できます。

たとえば、Web サーバーにパブリック IP アドレスがない場合は、ファイアウォールの **80** ポートおよび **443** ポートの着信パケットを Web サーバーに転送するファイアウォールのポート転送ルールを設定できます。このファイアウォールルールを使用すると、インターネットのユーザーは、ファイアウォールの IP またはホスト名を使用して Web サーバーにアクセスできます。

2.8.1. 着信パケットの別のローカルポートへの転送

nftables を使用してパケットを転送できます。たとえば、ポート **8022** の着信 IPv4 パケットを、ローカルシステムのポート **22** に転送できます。

手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

   ```
   # nft add table ip nat
   ```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

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

   注記

   -- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **8022** ポートの着信パケットを、ローカルポート **22** にリダイレクトするルールを **prerouting** チェーンに追加します。

   ```
   # nft add rule ip nat prerouting tcp dport 8022 redirect to :22
   ```

2.8.2. 特定のローカルポートで着信パケットを別のホストに転送

宛先ネットワークアドレス変換 (DNAT) ルールを使用して、ローカルポートの着信パケットをリモートホストに転送できます。これにより、インターネット上のユーザーは、プライベート IP アドレスを持つホストで実行しているサービスにアクセスできるようになります。

たとえば、ローカルポート **443** の着信 IPv4 パケットを、IP アドレス **192.0.2.1** を持つリモートシステムの同じポート番号に転送できます。

前提条件

- パケットを転送するシステムに **root** ユーザーとしてログインしている。

手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

```
# nft add table ip nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

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

注記

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **443** ポートの着信パケットを **192.0.2.1** 上の同じポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 出力トラフィックをマスカレードするルールを **postrouting** チェーンに追加します。

```
# nft add rule ip nat postrouting daddr 192.0.2.1 masquerade
```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.9. NFTABLES を使用した接続の量の制限

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

2.9.1. nftables を使用した接続数の制限

nft ユーティリティーの **ct count** パラメーターを使用すると、管理者は接続数を制限することができます。

前提条件

- **example_table** にベースの **example_chain** が存在する。

手順

1. IPv4 アドレスの動的セットを作成します。

```
# nft add set inet example_table example_meter { type ipv4_addr\; flags dynamic \;}
```

2. IPv4 アドレスから SSH ポート (22) への 2 つの同時接続のみを許可し、同じ IP からのすべての接続を拒否するルールを追加します。

```
# nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

3. オプション: 前の手順で作成したセットを表示します。

```
# nft list set inet example_table example_meter
table inet example_table {
  meter example_meter {
    type ipv4_addr
    size 65535
    elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2  }
  }
}
```

elements エントリーは、現時点でルールに一致するアドレスを表示します。この例では、**elements** は、SSH ポートへのアクティブな接続がある IP アドレスを一覧表示します。出力には、アクティブな接続の数を表示しないため、接続が拒否された場合は表示されないことに注意してください。

2.9.2. 1 分以内に新しい着信 TCP 接続を 11 個以上試行する IP アドレスのブロック

1 分以内に 11 個以上の IPv4 TCP 接続を確立しているホストを一時的にブロックできます。

手順

1. **ip** アドレスファミリーを使用して **filter** テーブルを作成します。

```
# nft add table ip filter
```

2. **input** チェーンを **filter** テーブルに追加します。

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. **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、セットが古いエントリーで一杯にならないように、5 分後にエントリーを自動的に削除することを定義します。

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 add @denylist { ip saddr limit rate over 10/minute } drop
```

関連情報

- nftables で名前付きセットの使用

2.10. NFTABLES ルールのデバッグ

nftables フレームワークは、管理者がルールをデバッグし、パケットがそれに一致するかどうかを確認するためのさまざまなオプションを提供します。

2.10.1. カウンターによるルールの作成

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- 既存のルールにカウンターを追加する手順の詳細は、Adding a counter to an existing rule を参照してください。

前提条件

- ルールを追加するチェーンが存在する。

手順

1. **counter** パラメーターで新しいルールをチェーンに追加します。以下の例では、ポート 22 で TCP トラフィックを許可し、このルールに一致するパケットとトラフィックをカウントするカウンターを使用するルールを追加します。

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

2. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

2.10.2. 既存のルールへのカウンターの追加

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- カウンターで新しいルールを追加する手順の詳細は、Creating a rule with the counter を参照してください。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

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

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

3. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

2.10.3. 既存のルールに一致するパケットの監視

nftables のトレース機能と、**nft monitor** コマンドを組み合わせることにより、管理者はルールに一致するパケットを表示できます。このルールに一致するパケットを監視するために、ルールのトレースを有効にできます。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールを置き換えてトレース機能を追加しますが、**meta nftrace set 1** パラメーターを使用します。以下の例は、前の手順で表示したルールの代わりに、トレースを有効にします。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
```

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

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

警告

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

2.11. NFTABLES ルールセットのバックアップおよび復元

nftables ルールをファイルにバックアップし、後で復元できます。また、管理者はルールが含まれるファイルを使用して、たとえばルールを別のサーバーに転送できます。

2.11.1. ファイルへの nftables ルールセットのバックアップ

nft ユーティリティーを使用して、**nftables** ルールセットをファイルにバックアップできます。

手順

- **nftables** ルールのバックアップを作成するには、次のコマンドを実行します。
  - **nft list ruleset** 形式で生成された形式の場合:

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

  - JSON 形式の場合は、以下のようになります。

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

2.11.2. ファイルからの nftables ルールセットの復元

ファイルから **nftables** ルールセットを復元できます。

手順

- **nftables** ルールを復元するには、以下を行います。
  - 復元するファイルが、**nft list ruleset** が生成した形式であるか、**nft** コマンドを直接含んでいる場合は、以下のコマンドを実行します。

    ```
    # nft -f file.nft
    ```

  - 復元するファイルが JSON 形式の場合は、次のコマンドを実行します。

    ```
    # nft -j -f file.json
    ```

2.12. 関連情報

- Using nftables in Red Hat Enterprise Linux 8
- What comes after iptables?Its successor, of course: nftables
- Firewalld:The Future is nftables

第3章 DDOS 攻撃を防ぐために、高パフォーマンストラフィックのフィルタリングで XDP-FILTER を使用

nftables と比べて、Express Data Path (XDP) は、パネットワークインターフェイスでネットワークパケットを処理して破棄します。したがって、XDP は、ファイアウォールやその他のアプリケーションに到達する前に、パッケージの次のステップを決定します。その結果、XDP フィルターは必要なリソースが少なく、DDoS (Distributed Denial of Service) 攻撃に備えるために、従来のパケットフィルターよりもはるかに高いレートでネットワークパケットを処理できます。たとえば、テスト時に、Red Hat は、1 つのコア上で 1 秒あたり 26 のネットワークパケットを破棄します。これは、同じハードウェアの **nftables** ドロップレートよりもはるかに高くなります。

xdp-filter ユーティリティーは、XDP を使用して着信ネットワークパケットを許可または破棄します。特定のトラフィックに対するトラフィックのフィルターを行うルールを作成できます。

- IP アドレス
- MAC アドレス
- ポート

xdp-filter にパケット処理速度が大幅に高くなりますが、nftables など、**nftables** は同じ機能がないことに注意してください。XDP を使用したパケットのフィルタリングを例示します。**xdp-filter** は、XDP を使用したパケットのフィルタリングを実証します。また、独自の XDP アプリケーションを作成する方法を理解するために、ユーティリティーのコードを使用できます。

重要

AMD および Intel 64 ビット以外のアーキテクチャーでは、**xdp-filter** ユーティリティーはテクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビュー機能では、最新の製品機能をいち早く提供します。これにより、お客様は開発段階で機能をテストし、フィードバックを提供できます。

テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

3.1. XDP-FILTER ルールに一致するネットワークパケットの削除

xdp-filter を使用して、ネットワークパケットをドロップできます。

- 特定の宛先ポートへの特定の宛先ポート
- 特定の IP アドレスの使用
- 特定の MAC アドレスの使用

xdp-filter の **allow** ポリシーは、すべてのトラフィックが許可され、フィルターが特定のルールに一致するネットワークパケットのみをドロップするように定義します。たとえば、ドロップするパケットのソース IP アドレスを知っている場合は、この方法を使用します。

前提条件

- **xdp-tools** パッケージがインストールされている。

- XDP プログラムをサポートするネットワークドライバー。

手順

1. **xdp-filter** を読み込み、**enp1s0** などの特定のインターフェイスの着信パケットを処理します。

```
# xdp-filter load enp1s0
```

デフォルトでは、**xdp-filter** は **allow** ポリシーを使用し、ユーティリティーはすべてのルールに一致するトラフィックのみを破棄します。

オプションで、**-f feature** オプションを使用して、**tcp**、**ipv4**、**ethernet** などの特定の機能のみを有効にします。すべての機能をロードするのではなく、必要な機能のみをロードすることで、パケット処理の速度が向上します。複数の機能を有効にするには、コンマで区切ります。

コマンドがエラーで失敗した場合、ネットワークドライバーは XDP プログラムをサポートしません。

2. ルールを追加して、それに一致するパケットをドロップします。以下に例を示します。

- 受信パケットをポート **22** に破棄するには、次のコマンドを実行します。

```
# xdp-filter port 22
```

このコマンドは、TCP および UDP トラフィックに一致するルールを追加します。特定のプロトコルのみと一致する場合は、**-p protocol** オプションを使用します。

- **192.0.2.1** から着信パケットを破棄するには、次のコマンドを実行します。

```
# xdp-filter ip 192.0.2.1 -m src
```

xdp-filter は IP 範囲に対応していないことに注意してください。

- MAC アドレス **00:53:00:AA:07:BE** から着信パケットを破棄するには、次のコマンドを実行します。

```
# xdp-filter ether 00:53:00:AA:07:BE -m src
```

検証

- 以下のコマンドを使用して、破棄されたパケットおよび許可されるパケットに関する統計を表示します。

```
# xdp-filter status
```

関連情報

- **xdp-filter(8)** の man ページ
- 開発者であり、**xdp-filter** のコードに関心がある場合は、Red Hat カスタマーポータルから対応するソース RPM (SRPM) をダウンロードしてインストールします。

3.2. XDP-FILTER ルールに一致するネットワークパケット以外のネットワークパケットをすべて削除

xdp-filter を使用して、ネットワークパケットのみを許可できます。

- 特定の宛先ポートから、あるいは指定された宛先ポートへ
- 特定の IP アドレスから、あるいは特定の IP アドレスへ
- 特定の MAC アドレスから、あるいは特定の MAC アドレスまで

これを行うには、特定のルールに一致するネットワークパケット以外のネットワークパケットをすべて破棄する **xdp-filter** の **deny** ポリシーを使用します。たとえば、ドロップするパケットのソース IP アドレスがわからない場合は、この方法を使用します。

警告

インターフェイスで **xdp-filter** を読み込む際にデフォルトのポリシーを **deny** に設定すると、特定のトラフィックを許可するルールを作成するまで、カーネルはこのインターフェイスからのパケットをすべて直ちに破棄します。システムからロックアウトしないようにするには、ローカルにコマンドを入力するか、別のネットワークインターフェイスからホストに接続します。

前提条件

- **xdp-tools** パッケージがインストールされている。
- ホストにローカルにログインするか、トラフィックのフィルタリングを予定しないネットワークインターフェイスを使用してホストにログインします。
- XDP プログラムをサポートするネットワークドライバー。

手順

1. **xdp-filter** を読み込み、**enp1s0** などの特定のインターフェイスのパケットを処理します。

```
# xdp-filter load enp1s0 -p deny
```

   オプションで、**-f feature** オプションを使用して、**tcp**、**ipv4**、**ethernet** などの特定の機能のみを有効にします。すべての機能をロードするのではなく、必要な機能のみをロードすることで、パケット処理の速度が向上します。複数の機能を有効にするには、コンマで区切ります。

   コマンドがエラーで失敗した場合、ネットワークドライバーは XDP プログラムをサポートしません。

2. ルールを追加して、一致するパケットを許可します。以下に例を示します。
   - パケットのポート **22** を許可するには、以下のコマンドを実行します。

```
# xdp-filter port 22
```

このコマンドは、TCP および UDP トラフィックに一致するルールを追加します。特定のプロトコルのみと一致するように、**-p protocol** オプションをコマンドに渡します。

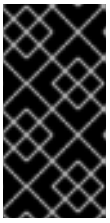
- パケットの **192.0.2.1** を許可するには、次のコマンドを実行します。

```
# xdp-filter ip 192.0.2.1
```

xdp-filter は IP 範囲に対応していないことに注意してください。

- MAC アドレス **00:53:00:AA:07:BE** へのパケットを許可するには、次のコマンドを実行します。

```
# xdp-filter ether 00:53:00:AA:07:BE
```



重要

xdp-filter ユーティリティーは、ステートフルパケットの検査に対応していません。これには、**-m mode** オプションでモードを設定せず、マシンが送信トラフィックに応答して受信トラフィックを許可する明示的なルールを追加する必要があります。

検証

- 以下のコマンドを使用して、破棄されたパケットおよび許可されるパケットに関する統計を表示します。

```
# xdp-filter status
```

関連情報

- **xdp-filter(8)** の man ページ。
- 開発者であり、**xdp-filter** のコードに関心がある場合は、Red Hat カスタマーポータルから対応するソース RPM (SRPM) をダウンロードしてインストールします。