Red Hat Enterprise Linux 9

Configuration des pare-feu et des filtres de paquets

Gestion du service firewalld, du cadre nftables et des fonctions de filtrage de paquets XDP

Last Updated: 2024-01-24

Red Hat Enterprise Linux 9 Configuration des pare-feu et des filtres de paquets

Gestion du service firewalld, du cadre nftables et des fonctions de filtrage de paquets XDP

Notice légale

Résumé

Les filtres de paquets, tels que les pare-feu, utilisent des règles pour contrôler le trafic réseau entrant, sortant et transféré. Dans Red Hat Enterprise Linux (RHEL), vous pouvez utiliser le service firewalld et le cadre nftables pour filtrer le trafic réseau et construire des pare-feu dont les performances sont critiques. Vous pouvez également utiliser la fonctionnalité XDP (Express Data Path) du noyau pour traiter ou abandonner des paquets réseau à l'interface réseau à un taux très élevé.

Table des matières

RENDRE L'OPEN SOURCE PLUS INCLUSIF

Red Hat s'engage à remplacer les termes problématiques dans son code, sa documentation et ses propriétés Web. Nous commençons par ces quatre termes : master, slave, blacklist et whitelist. En raison de l'ampleur de cette entreprise, ces changements seront mis en œuvre progressivement au cours de plusieurs versions à venir. Pour plus de détails, voir le message de notre directeur technique Chris Wright.

FOURNIR UN RETOUR D'INFORMATION SUR LA DOCUMENTATION DE RED HAT

Nous apprécions vos commentaires sur notre documentation. Faites-nous savoir comment nous pouvons l'améliorer.

Soumettre des commentaires sur des passages spécifiques

1. Consultez la documentation au format **Multi-page HTML** et assurez-vous que le bouton **Feedback** apparaît dans le coin supérieur droit après le chargement complet de la page.
2. Utilisez votre curseur pour mettre en évidence la partie du texte que vous souhaitez commenter.
3. Cliquez sur le bouton **Add Feedback** qui apparaît près du texte en surbrillance.
4. Ajoutez vos commentaires et cliquez sur **Submit**.

Soumettre des commentaires via Bugzilla (compte requis)

1. Connectez-vous au site Web de Bugzilla.
2. Sélectionnez la version correcte dans le menu **Version**.
3. Saisissez un titre descriptif dans le champ **Summary**.
4. Saisissez votre suggestion d'amélioration dans le champ **Description**. Incluez des liens vers les parties pertinentes de la documentation.
5. Cliquez sur **Submit Bug**.

CHAPITRE 1. UTILISATION ET CONFIGURATION DE FIREWALLD

firewall est un moyen de protéger les machines contre tout trafic indésirable provenant de l'extérieur. Il permet aux utilisateurs de contrôler le trafic réseau entrant sur les machines hôtes en définissant un ensemble de *firewall rules*. Ces règles sont utilisées pour trier le trafic entrant et le bloquer ou l'autoriser.

firewalld est un démon de service de pare-feu qui fournit un pare-feu dynamique personnalisable basé sur l'hôte avec une interface D-Bus. Comme il est dynamique, il permet de créer, de modifier et de supprimer les règles sans qu'il soit nécessaire de redémarrer le démon du pare-feu à chaque fois que les règles sont modifiées.

firewalld utilise les concepts de zones et de services, qui simplifient la gestion du trafic. Les zones sont des ensembles de règles prédéfinies. Les interfaces réseau et les sources peuvent être affectées à une zone. Le trafic autorisé dépend du réseau auquel votre ordinateur est connecté et du niveau de sécurité attribué à ce réseau. Les services de pare-feu sont des règles prédéfinies qui couvrent tous les paramètres nécessaires pour autoriser le trafic entrant pour un service spécifique et s'appliquent à l'intérieur d'une zone.

Les services utilisent un ou plusieurs ports ou adresses pour communiquer avec le réseau. Les pare-feu filtrent les communications en fonction des ports. Pour autoriser le trafic réseau d'un service, ses ports doivent être ouverts. **firewalld** bloque tout le trafic sur les ports qui ne sont pas explicitement définis comme ouverts. Certaines zones, comme la zone de confiance, autorisent tout le trafic par défaut.

Notez que **firewalld** avec le backend **nftables** ne permet pas de passer des règles **nftables** personnalisées à **firewalld**, en utilisant l'option **--direct**.

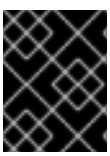
1.1. DÉMARRER AVEC FIREWALLD

Ce qui suit est une introduction aux fonctionnalités de **firewalld**, telles que les services et les zones, et à la manière de gérer le service **firewalld** systemd.

1.1.1. Quand utiliser firewalld, nftables ou iptables ?

Voici un bref aperçu des scénarios dans lesquels vous devez utiliser l'un des utilitaires suivants :

- **firewalld**: Utilisez l'utilitaire **firewalld** pour des cas d'utilisation simples de pare-feu. L'utilitaire est facile à utiliser et couvre les cas d'utilisation typiques de ces scénarios.
- **nftables**: Utilisez l'utilitaire **nftables** pour configurer des pare-feu complexes et critiques en termes de performances, par exemple pour l'ensemble d'un réseau.
- **iptables**: L'utilitaire **iptables** sur Red Hat Enterprise Linux utilise l'API du noyau **nf_tables** au lieu de l'interface **legacy**. L'API **nf_tables** offre une compatibilité ascendante de sorte que les scripts qui utilisent les commandes **iptables** fonctionnent toujours sur Red Hat Enterprise Linux. Pour les nouveaux scripts de pare-feu, Red Hat recommande d'utiliser **nftables**.



IMPORTANT

Pour éviter que les différents services de pare-feu ne s'influencent mutuellement, exécutez un seul d'entre eux sur un hôte RHEL et désactivez les autres services.

1.1.2. Zones

firewalld peut être utilisé pour séparer les réseaux en différentes zones en fonction du niveau de confiance que l'utilisateur a décidé d'accorder aux interfaces et au trafic au sein de ce réseau. Une connexion ne peut faire partie que d'une seule zone, mais une zone peut être utilisée pour plusieurs connexions réseau.

NetworkManager notifie à **firewalld** la zone d'une interface. Vous pouvez assigner des zones aux interfaces avec :

- **NetworkManager**
- **firewall-config** outil
- **firewall-cmd** outil en ligne de commande
- La console web RHEL

Les trois derniers ne peuvent éditer que les fichiers de configuration appropriés de **NetworkManager**. Si vous changez la zone de l'interface à l'aide de la console web, **firewall-cmd** ou **firewall-config**, la demande est transmise à **NetworkManager** et n'est pas traitée par **firewalld**.

Les zones prédéfinies sont stockées dans le répertoire **/usr/lib/firewalld/zones/** et peuvent être appliquées instantanément à toute interface réseau disponible. Ces fichiers ne sont copiés dans le répertoire **/etc/firewalld/zones/** qu'après avoir été modifiés. Les paramètres par défaut des zones prédéfinies sont les suivants :

block

Toute connexion réseau entrante est rejetée avec un message icmp-host-prohibited pour **IPv4** et icmp6-adm-prohibited pour **IPv6**. Seules les connexions réseau initiées depuis l'intérieur du système sont possibles.

dmz

Pour les ordinateurs de votre zone démilitarisée qui sont accessibles au public avec un accès limité à votre réseau interne. Seules les connexions entrantes sélectionnées sont acceptées.

drop

Tous les paquets réseau entrants sont abandonnés sans notification. Seules les connexions réseau sortantes sont possibles.

external

À utiliser sur les réseaux externes où le masquage est activé, en particulier pour les routeurs. Vous ne faites pas confiance aux autres ordinateurs du réseau pour ne pas nuire à votre ordinateur. Seules les connexions entrantes sélectionnées sont acceptées.

home

À utiliser à la maison lorsque vous faites essentiellement confiance aux autres ordinateurs du réseau. Seules les connexions entrantes sélectionnées sont acceptées.

internal

À utiliser sur les réseaux internes lorsque vous faites essentiellement confiance aux autres ordinateurs du réseau. Seules les connexions entrantes sélectionnées sont acceptées.

public

À utiliser dans les lieux publics où vous ne faites pas confiance aux autres ordinateurs du réseau. Seules les connexions entrantes sélectionnées sont acceptées.

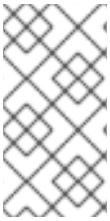
trusted

Toutes les connexions réseau sont acceptées.

work

À utiliser au travail lorsque vous faites essentiellement confiance aux autres ordinateurs du réseau. Seules les connexions entrantes sélectionnées sont acceptées.

L'une de ces zones est définie comme la zone *default*. Lorsque des connexions d'interface sont ajoutées à **NetworkManager**, elles sont affectées à la zone par défaut. Lors de l'installation, la zone par défaut dans **firewalld** est définie comme étant la zone **public**. La zone par défaut peut être modifiée.



NOTE

Les noms des zones du réseau doivent être explicites et permettre aux utilisateurs de prendre rapidement une décision raisonnable. Pour éviter tout problème de sécurité, examinez la configuration de la zone par défaut et désactivez tous les services inutiles en fonction de vos besoins et de l'évaluation des risques.

Ressources supplémentaires

- La page de manuel **firewalld.zone(5)**.

1.1.3. Services prédéfinis

Un service peut être une liste de ports locaux, de protocoles, de ports sources et de destinations, ainsi qu'une liste de modules d'aide au pare-feu automatiquement chargés si un service est activé. L'utilisation de services permet aux utilisateurs de gagner du temps car ils peuvent effectuer plusieurs tâches, telles que l'ouverture de ports, la définition de protocoles, l'activation du transfert de paquets et autres, en une seule étape, plutôt que de tout configurer l'un après l'autre.

Les options de configuration des services et les informations génériques sur les fichiers sont décrites dans la page de manuel **firewalld.service(5)**. Les services sont spécifiés au moyen de fichiers de configuration XML individuels, qui sont nommés dans le format suivant : ***service-name*.xml**. Les noms de protocoles sont préférés aux noms de services ou d'applications dans **firewalld**.

Des services peuvent être ajoutés et supprimés à l'aide de l'outil graphique **firewall-config**, **firewall-cmd** et **firewall-offline-cmd**.

Vous pouvez également modifier les fichiers XML dans le répertoire **/etc/firewalld/services/**. Si un service n'est pas ajouté ou modifié par l'utilisateur, aucun fichier XML correspondant n'est trouvé dans **/etc/firewalld/services/**. Les fichiers du répertoire **/usr/lib/firewalld/services/** peuvent être utilisés comme modèles si vous souhaitez ajouter ou modifier un service.

Ressources supplémentaires

- La page de manuel **firewalld.service(5)**

1.1.4. Démarrer firewalld

Procédure

1. Pour démarrer **firewalld**, entrez la commande suivante en tant que **root**:

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

2. Pour que **firewalld** démarre automatiquement au démarrage du système, entrez la commande suivante en tant que **root**:

```
# systemctl enable firewalld
```

1.1.5. Arrêt de firewalld

Procédure

1. Pour arrêter **firewalld**, entrez la commande suivante en tant que **root**:

```
# systemctl stop firewalld
```

2. Pour éviter que **firewalld** ne démarre automatiquement au démarrage du système :

```
# systemctl disable firewalld
```

3. Pour s'assurer que firewalld n'est pas démarré en accédant à l'interface **firewalld D-Bus** et aussi si d'autres services nécessitent **firewalld**:

```
# systemctl mask firewalld
```

1.1.6. Vérification de la configuration permanente de firewalld

Dans certaines situations, par exemple après avoir modifié manuellement les fichiers de configuration de **firewalld**, les administrateurs souhaitent vérifier que les modifications sont correctes. Vous pouvez utiliser l'utilitaire **firewall-cmd** pour vérifier la configuration.

Conditions préalables

- Le service **firewalld** est en cours d'exécution.

Procédure

1. Vérifier la configuration permanente du service **firewalld**:

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

Si la configuration permanente est valide, la commande renvoie **success**. Dans les autres cas, la commande renvoie une erreur avec des détails supplémentaires, tels que les suivants :

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

1.2. VISUALISATION DE L'ÉTAT ACTUEL ET DES PARAMÈTRES DE FIREWALLD

Pour surveiller le service **firewalld**, vous pouvez afficher l'état, les services autorisés et les paramètres.

1.2.1. Visualisation de l'état actuel de firewalld

Le service de pare-feu, **firewalld**, est installé par défaut sur le système. Utilisez l'interface CLI de **firewalld** pour vérifier que le service est en cours d'exécution.

Procédure

1. Pour connaître l'état du service :

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

2. Pour plus d'informations sur l'état du service, utilisez la sous-commande **systemctl status**:

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

1.2.2. Visualisation des services autorisés à l'aide de l'interface graphique

Pour afficher la liste des services à l'aide de l'outil graphique **firewall-config**, appuyez sur la touche **Super** pour accéder à la vue d'ensemble des activités, tapez **firewall** et appuyez sur **Entrée**. L'outil **firewall-config** s'affiche. Vous pouvez maintenant consulter la liste des services sous l'onglet **Services**.

Vous pouvez lancer l'outil de configuration graphique du pare-feu à l'aide de la ligne de commande.

Conditions préalables

- Vous avez installé le paquetage **firewall-config**.

Procédure

- Pour lancer l'outil de configuration graphique du pare-feu à l'aide de la ligne de commande :

```
$ firewall-config
```

La fenêtre **Firewall Configuration** s'ouvre. Notez que cette commande peut être exécutée en tant qu'utilisateur normal, mais qu'un mot de passe d'administrateur vous est demandé de temps à autre.

1.2.3. Visualisation des paramètres de firewalld à l'aide de la CLI

Avec le client CLI, il est possible d'obtenir différentes vues des paramètres actuels du pare-feu. L'option **--list-all** affiche un aperçu complet des paramètres de **firewalld**.

firewalld utilise des zones pour gérer le trafic. Si aucune zone n'est spécifiée par l'option **--zone**, la commande s'applique à la zone par défaut attribuée à l'interface réseau et à la connexion actives.

Procédure

- Pour répertorier toutes les informations pertinentes pour la zone par défaut :

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

- Pour spécifier la zone pour laquelle les paramètres doivent être affichés, ajoutez l'argument **--zone=*zone-name*** à la commande **firewall-cmd --list-all**, par exemple :

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

- Pour voir les paramètres d'une information particulière, comme les services ou les ports, utilisez une option spécifique. Consultez les pages du manuel **firewalld** ou obtenez une liste des options à l'aide de la commande help :

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

- Pour connaître les services autorisés dans la zone actuelle :

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

NOTE

L'énumération des paramètres d'une sous-partie donnée à l'aide de l'outil CLI peut parfois être difficile à interpréter. Par exemple, vous autorisez le service **SSH** et **firewalld** ouvre le port nécessaire (22) pour le service. Par la suite, si vous dressez la liste des services autorisés, le service **SSH** apparaît, mais si vous dressez la liste des ports ouverts, aucun n'apparaît. Il est donc recommandé d'utiliser l'option **--list-all** pour être sûr de recevoir une information complète.

1.3. CONTRÔLE DU TRAFIC RÉSEAU À L'AIDE DE FIREWALLD

Le paquet **firewalld** installe un grand nombre de fichiers de services prédéfinis et vous pouvez en ajouter d'autres ou les personnaliser. Vous pouvez ensuite utiliser ces définitions de service pour ouvrir ou fermer des ports pour des services sans connaître le protocole et les numéros de port qu'ils utilisent.

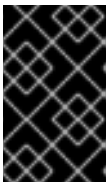
1.3.1. Désactivation de tout le trafic en cas d'urgence à l'aide de la CLI

Dans une situation d'urgence, telle qu'une attaque du système, il est possible de désactiver tout le trafic réseau et de couper l'herbe sous le pied de l'attaquant.

Procédure

1. Pour désactiver immédiatement le trafic réseau, activez le mode panique :

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



IMPORTANT

L'activation du mode panique interrompt tout le trafic réseau. C'est pourquoi il ne doit être utilisé que si vous avez un accès physique à la machine ou si vous êtes connecté à l'aide d'une console série.

2. La désactivation du mode panique ramène le pare-feu à ses paramètres permanents. Pour désactiver le mode panique, entrez :

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

Vérification

- Pour savoir si le mode panique est activé ou désactivé, utilisez :

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

1.3.2. Contrôler le trafic avec des services prédéfinis à l'aide de la CLI

La méthode la plus simple pour contrôler le trafic consiste à ajouter un service prédéfini à **firewalld**. Ce service ouvre tous les ports nécessaires et modifie d'autres paramètres en fonction de *service definition file*.

Procédure

1. Vérifiez que le service n'est pas déjà autorisé :

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

2. Liste de tous les services prédéfinis :

```
# 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 ...
```

3. Ajouter le service aux services autorisés :

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

4. Les nouveaux paramètres doivent être conservés :

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

1.3.3. Contrôle du trafic avec des services prédéfinis à l'aide de l'interface graphique

Vous pouvez contrôler le trafic réseau avec des services prédéfinis à l'aide d'une interface utilisateur graphique.

Conditions préalables

- Vous avez installé le paquetage **firewall-config**

Procédure

1. Pour activer ou désactiver un service prédéfini ou personnalisé :
   a. Lancez l'outil **firewall-config** et sélectionnez la zone du réseau dont les services doivent être configurés.
   b. Sélectionnez l'onglet **Zones** puis l'onglet **Services** ci-dessous.
   c. Cochez la case pour chaque type de service que vous souhaitez autoriser ou décochez la case pour bloquer un service dans la zone sélectionnée.
2. Pour modifier un service :
   a. Lancez l'outil **firewall-config**.
   b. Sélectionnez **Permanent** dans le menu intitulé **Configuration**. D'autres icônes et boutons de menu apparaissent au bas de la fenêtre **Services**.
   c. Sélectionnez le service que vous souhaitez configurer.

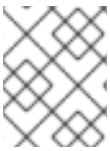
Les onglets **Ports**, **Protocols**, et **Source Port** permettent d'ajouter, de modifier et de supprimer les ports, les protocoles et le port source pour le service sélectionné. L'onglet modules permet de configurer les modules d'aide **Netfilter**. L'onglet **Destination** permet de limiter le trafic à une adresse de destination et à un protocole Internet particuliers (**IPv4** ou **IPv6**).

NOTE

Il n'est pas possible de modifier les paramètres de service en mode **Runtime**.

1.3.4. Ajout de nouveaux services

Les services peuvent être ajoutés et supprimés à l'aide de l'outil graphique **firewall-config**, **firewall-cmd** et **firewall-offline-cmd**. Vous pouvez également modifier les fichiers XML à l'adresse **/etc/firewalld/services/**. Si un service n'est pas ajouté ou modifié par l'utilisateur, aucun fichier XML correspondant n'est trouvé à l'adresse **/etc/firewalld/services/**. Les fichiers **/usr/lib/firewalld/services/** peuvent être utilisés comme modèles si vous souhaitez ajouter ou modifier un service.



NOTE

Les noms des services doivent être alphanumériques et ne peuvent comporter que les caractères _ (trait de soulignement) et **-** (tiret).

Procédure

Pour ajouter un nouveau service dans un terminal, utilisez **firewall-cmd**, ou **firewall-offline-cmd** si **firewalld** n'est pas actif.

1. Entrez la commande suivante pour ajouter un nouveau service vide :

```
$ firewall-cmd --new-service=<service_name> --permanent
```

2. Pour ajouter un nouveau service à l'aide d'un fichier local, utilisez la commande suivante :

```
$ firewall-cmd --new-service-from-file=<service_xml_file> --permanent
```

   Vous pouvez modifier le nom du service avec l'option supplémentaire **--name=*<service_name>*** supplémentaire.

3. Dès que les paramètres du service sont modifiés, une copie mise à jour du service est placée sur **/etc/firewalld/services/**.
   Comme **root**, vous pouvez entrer la commande suivante pour copier un service manuellement :

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

firewalld charge les fichiers de **/usr/lib/firewalld/services** en premier lieu. Si des fichiers sont placés dans **/etc/firewalld/services** et qu'ils sont valides, ils remplaceront les fichiers correspondants de **/usr/lib/firewalld/services**. Les fichiers remplacés dans **/usr/lib/firewalld/services** sont utilisés dès que les fichiers correspondants de **/etc/firewalld/services** ont été supprimés ou si l'on a demandé à **firewalld** de charger les valeurs par défaut des services. Ceci ne s'applique qu'à l'environnement permanent. Un rechargement est nécessaire pour obtenir ces fallbacks également dans l'environnement d'exécution.

1.3.5. Ouverture de ports à l'aide de l'interface graphique

Pour autoriser le trafic à travers le pare-feu vers un certain port, vous pouvez ouvrir le port dans l'interface graphique.

Conditions préalables

- Vous avez installé le paquetage **firewall-config**

Procédure

1. Lancez l'outil **firewall-config** et sélectionnez la zone du réseau dont vous souhaitez modifier les paramètres.
2. Sélectionnez l'onglet **Ports** et cliquez sur le bouton **Ajouter** sur le côté droit. La fenêtre **Port and Protocol** s'ouvre.
3. Saisissez le numéro de port ou la plage de ports à autoriser.
4. Sélectionnez **tcp** ou **udp** dans la liste.

1.3.6. Contrôle du trafic avec des protocoles à l'aide de l'interface graphique

Pour autoriser le trafic à travers le pare-feu en utilisant un certain protocole, vous pouvez utiliser l'interface graphique.

Conditions préalables

- Vous avez installé le paquetage **firewall-config**

Procédure

1. Lancez l'outil **firewall-config** et sélectionnez la zone du réseau dont vous souhaitez modifier les paramètres.
2. Sélectionnez l'onglet **Protocols** et cliquez sur le bouton **Add** à droite. La fenêtre **Protocol** s'ouvre.
3. Sélectionnez un protocole dans la liste ou cochez la case **Other Protocol** et saisissez le protocole dans le champ.

1.3.7. Ouverture des ports source à l'aide de l'interface graphique

Pour autoriser le trafic à travers le pare-feu à partir d'un certain port, vous pouvez utiliser l'interface graphique.

Conditions préalables

- Vous avez installé le paquetage **firewall-config**

Procédure

1. Lancez l'outil firewall-config et sélectionnez la zone réseau dont vous souhaitez modifier les paramètres.
2. Sélectionnez l'onglet **Source Port** et cliquez sur le bouton **Add** à droite. La fenêtre **Source Port** s'ouvre.
3. Saisissez le numéro de port ou la plage de ports à autoriser. Sélectionnez **tcp** ou **udp** dans la liste.

1.4. CONTRÔLE DES PORTS À L'AIDE DE L'INTERFACE DE PROGRAMMATION

Les ports sont des dispositifs logiques qui permettent à un système d'exploitation de recevoir et de distinguer le trafic réseau et de le transmettre en conséquence aux services du système. Ils sont généralement représentés par un démon qui écoute sur le port, c'est-à-dire qu'il attend tout trafic arrivant sur ce port.

Normalement, les services système écoutent sur les ports standard qui leur sont réservés. Le démon **httpd**, par exemple, écoute sur le port 80. Toutefois, les administrateurs système configurent par défaut les démons pour qu'ils écoutent sur des ports différents afin d'améliorer la sécurité ou pour d'autres raisons.

1.4.1. Ouverture d'un port

Les ports ouverts permettent au système d'être accessible de l'extérieur, ce qui représente un risque pour la sécurité. En règle générale, il convient de garder les ports fermés et de ne les ouvrir que s'ils sont nécessaires pour certains services.

Procédure

Pour obtenir une liste des ports ouverts dans la zone actuelle :

1. Liste de tous les ports autorisés :

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

2. Ajouter un port aux ports autorisés pour l'ouvrir au trafic entrant :

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

   Les types de port sont soit **tcp**, **udp**, **sctp**, ou **dccp**. Le type doit correspondre au type de communication réseau.

3. Les nouveaux paramètres doivent être conservés :

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

   Les types de port sont soit **tcp**, **udp**, **sctp**, ou **dccp**. Le type doit correspondre au type de communication réseau.

1.4.2. Fermeture d'un port

Lorsqu'un port ouvert n'est plus nécessaire, fermez-le dans **firewalld**. Il est fortement recommandé de fermer tous les ports inutiles dès qu'ils ne sont plus utilisés, car laisser un port ouvert représente un risque pour la sécurité.

Procédure

Pour fermer un port, il faut le supprimer de la liste des ports autorisés :

1. Liste de tous les ports autorisés :

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

AVERTISSEMENT

Cette commande ne vous donnera qu'une liste des ports qui ont été ouverts en tant que ports. Vous ne pourrez pas voir les ports ouverts en tant que service. Par conséquent, vous devriez envisager d'utiliser l'option **--list-all** au lieu de **--list-ports**.

2. Retirer le port des ports autorisés pour le fermer au trafic entrant :

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

3. Les nouveaux paramètres doivent être conservés :

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

1.5. TRAVAILLER AVEC LES ZONES FIREWALLD

Les zones représentent un concept permettant de gérer le trafic entrant de manière plus transparente. Les zones sont connectées à des interfaces de réseau ou se voient attribuer une série d'adresses sources. Vous gérez les règles de pare-feu pour chaque zone indépendamment, ce qui vous permet de définir des paramètres de pare-feu complexes et de les appliquer au trafic.

1.5.1. Zones d'inscription

Vous pouvez dresser la liste des zones à l'aide de la ligne de commande.

Procédure

1. Pour connaître les zones disponibles sur votre système :

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

   La commande **firewall-cmd --get-zones** affiche toutes les zones disponibles sur le système, mais n'indique aucun détail sur des zones particulières.

2. Pour obtenir des informations détaillées sur toutes les zones :

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

3. Pour obtenir des informations détaillées sur une zone spécifique :

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

1.5.2. Modifier les paramètres de firewalld pour une certaine zone

Les sections Contrôle du trafic avec des services prédéfinis à l'aide de cli et Contrôle des ports à l'aide de cli expliquent comment ajouter des services ou modifier des ports dans le cadre de la zone de travail actuelle. Il est parfois nécessaire de définir des règles dans une autre zone.

Procédure

- Pour travailler dans une autre zone, utilisez l'option **--zone=*<zone_name>*** pour fonctionner dans une autre zone. Par exemple, pour autoriser le service **SSH** dans la zone **public**:

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

1.5.3. Modifier la zone par défaut

Les administrateurs système attribuent une zone à une interface réseau dans ses fichiers de configuration. Si une interface n'est pas assignée à une zone spécifique, elle est assignée à la zone par défaut. Après chaque redémarrage du service **firewalld**, **firewalld** charge les paramètres de la zone par défaut et la rend active.

Procédure

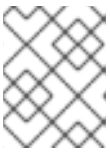
Pour configurer la zone par défaut :

1. Affiche la zone par défaut actuelle :

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

2. Définir la nouvelle zone par défaut :

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



NOTE

En suivant cette procédure, le réglage est permanent, même sans l'option **--permanent**.

1.5.4. Affectation d'une interface réseau à une zone

Il est possible de définir différents ensembles de règles pour différentes zones, puis de modifier rapidement les paramètres en changeant la zone de l'interface utilisée. En cas d'interfaces multiples, une zone spécifique peut être définie pour chacune d'entre elles afin de distinguer le trafic qui les traverse.

Procédure

Pour affecter la zone à une interface spécifique :

1. Liste des zones actives et des interfaces qui leur sont affectées :

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

2. Assigner l'interface à une zone différente :

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

1.5.5. Attribution d'une zone à une connexion à l'aide de nmcli

Vous pouvez ajouter une zone **firewalld** à une connexion **NetworkManager** à l'aide de l'utilitaire **nmcli**.

Procédure

1. Attribuer la zone au profil de connexion **NetworkManager**:

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

2. Activer la connexion :

```
# nmcli connection up profile
```

1.5.6. Attribution manuelle d'une zone à une connexion réseau dans un fichier ifcfg

Lorsque la connexion est gérée par **NetworkManager**, elle doit connaître la zone qu'elle utilise. Pour

chaque connexion réseau, une zone peut être spécifiée, ce qui permet d'adapter les paramètres du pare-feu en fonction de l'emplacement de l'ordinateur et des appareils portables. Ainsi, les zones et les paramètres peuvent être spécifiés pour différents lieux, tels que l'entreprise ou le domicile.

Procédure

- Pour définir une zone pour une connexion, modifiez le fichier **/etc/sysconfig/network-scripts/ifcfg-*connection_name*** et ajoutez une ligne qui attribue une zone à cette connexion :

```
ZONE=zone_name
```

1.5.7. Création d'une nouvelle zone

Pour utiliser des zones personnalisées, créez une nouvelle zone et utilisez-la comme une zone prédéfinie. Les nouvelles zones nécessitent l'option **--permanent**, sinon la commande ne fonctionne pas.

Procédure

1. Créer une nouvelle zone :

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

2. Vérifiez si la nouvelle zone est ajoutée à vos paramètres permanents :

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

3. Les nouveaux paramètres doivent être conservés :

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

1.5.8. Fichiers de configuration de la zone

Les zones peuvent également être créées à l'aide de *zone configuration file*. Cette approche peut être utile lorsque vous devez créer une nouvelle zone, mais que vous souhaitez réutiliser les paramètres d'une autre zone et ne les modifier que légèrement.

Un fichier de configuration de zone **firewalld** contient les informations relatives à une zone. Il s'agit de la description de la zone, des services, des ports, des protocoles, des blocs icmp, de la mascarade, des ports de renvoi et des règles de langage riche dans un format de fichier XML. Le nom du fichier doit être ***zone-name*.xml** où la longueur de *zone-name* est actuellement limitée à 17 caractères. Les fichiers de configuration de la zone sont situés dans les répertoires **/usr/lib/firewalld/zones/** et **/etc/firewalld/zones/**.

L'exemple suivant montre une configuration qui autorise un service (**SSH**) et une plage de ports, pour les protocoles **TCP** et **UDP**:

```
<?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>
```

Pour modifier les paramètres de cette zone, ajoutez ou supprimez des sections pour ajouter des ports, transférer des ports, des services, etc.

Ressources supplémentaires

- **firewalld.zone** page du manuel

1.5.9. Utilisation des cibles de zone pour définir le comportement par défaut du trafic entrant

Pour chaque zone, vous pouvez définir un comportement par défaut qui gère le trafic entrant non spécifié. Ce comportement est défini en définissant la cible de la zone. Il existe quatre options :

- **ACCEPT**: Accepte tous les paquets entrants, à l'exception de ceux qui sont interdits par des règles spécifiques.
- **REJECT**: Rejette tous les paquets entrants sauf ceux autorisés par des règles spécifiques. Lorsque **firewalld** rejette des paquets, la machine source est informée du rejet.
- **DROP**: Abandonne tous les paquets entrants à l'exception de ceux autorisés par des règles spécifiques. Lorsque **firewalld** abandonne des paquets, la machine source n'est pas informée de l'abandon du paquet.
- **default**: Comportement similaire à celui de **REJECT**, mais avec des significations spéciales dans certains scénarios. Pour plus de détails, voir la section **Options to Adapt and Query Zones and Policies** dans la page de manuel **firewall-cmd(1)**.

Procédure

Pour définir une cible pour une zone :

1. Lister les informations pour la zone spécifique afin de voir la cible par défaut :

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

2. Fixer un nouvel objectif dans la zone :

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

Ressources supplémentaires

- **firewall-cmd(1)** page de manuel

1.6. UTILISATION DE ZONES POUR GÉRER LE TRAFIC ENTRANT EN FONCTION D'UNE SOURCE

Vous pouvez utiliser des zones pour gérer le trafic entrant en fonction de sa source. Cela vous permet de trier le trafic entrant et de le faire passer par différentes zones afin d'autoriser ou d'interdire les services qui peuvent être atteints par ce trafic.

Si vous ajoutez une source à une zone, celle-ci devient active et tout le trafic entrant provenant de cette source sera dirigé vers elle. Vous pouvez spécifier des paramètres différents pour chaque zone, qui seront appliqués au trafic provenant des sources données. Vous pouvez utiliser plusieurs zones même si vous n'avez qu'une seule interface réseau.

1.6.1. Ajout d'une source

Pour acheminer le trafic entrant vers une zone spécifique, ajoutez la source à cette zone. La source peut être une adresse IP ou un masque IP dans la notation CIDR (classless inter-domain routing).

NOTE

Si vous ajoutez plusieurs zones dont la portée du réseau se chevauche, elles sont classées par ordre alphanumérique selon le nom de la zone et seule la première est prise en compte.

- Pour définir la source dans la zone actuelle :

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

- Pour définir l'adresse IP source d'une zone spécifique :

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

La procédure suivante autorise tout le trafic entrant de *192.168.2.15* dans la zone **trusted**:

Procédure

1. Liste de toutes les zones disponibles :

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

2. Ajouter l'IP source à la zone de confiance en mode permanent :

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

3. Les nouveaux paramètres doivent être conservés :

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

1.6.2. Suppression d'une source

La suppression d'une source de la zone coupe le trafic provenant de cette source.

Procédure

1. Liste des sources autorisées pour la zone requise :

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

2. Supprimer définitivement la source de la zone :

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

3. Les nouveaux paramètres doivent être conservés :

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

1.6.3. Ajout d'un port source

Pour trier le trafic en fonction du port d'origine, spécifiez un port source à l'aide de l'option **--add-source-port**. Vous pouvez également combiner cette option avec l'option **--add-source** pour limiter le trafic à une certaine adresse IP ou à une certaine plage d'adresses IP.

Procédure

- Pour ajouter un port source :

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

1.6.4. Suppression d'un port source

En supprimant un port source, vous désactivez le tri du trafic en fonction du port d'origine.

Procédure

- Pour supprimer un port source :

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

1.6.5. Utiliser les zones et les sources pour autoriser un service uniquement pour un domaine spécifique

Pour autoriser le trafic provenant d'un réseau spécifique à utiliser un service sur une machine, utilisez les zones et la source. La procédure suivante autorise uniquement le trafic HTTP en provenance du réseau **192.0.2.0/24**, tandis que tout autre trafic est bloqué.

AVERTISSEMENT

Lorsque vous configurez ce scénario, utilisez une zone dont la cible est **default**. L'utilisation d'une zone dont la cible est **ACCEPT** présente un risque pour la sécurité, car pour le trafic provenant de **192.0.2.0/24**, toutes les connexions réseau seraient acceptées.

Procédure

1. Liste de toutes les zones disponibles :

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

2. Ajoutez la plage d'adresses IP à la zone **internal** pour acheminer le trafic provenant de la source à travers la zone :

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

3. Ajouter le service **http** à la zone **internal**:

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

4. Les nouveaux paramètres doivent être conservés :

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

Vérification

- Vérifiez que la zone **internal** est active et que le service y est autorisé :

```
# 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
  ...
```

Ressources supplémentaires

- **firewalld.zones(5)** page de manuel

1.7. FILTRAGE DU TRAFIC TRANSFÉRÉ ENTRE LES ZONES

Avec un objet de politique, les utilisateurs peuvent regrouper différentes identités qui requièrent des autorisations similaires dans la politique. Vous pouvez appliquer des règles en fonction de la direction du trafic.

La fonction d'objets de stratégie permet le filtrage en amont et en aval dans firewalld. Vous pouvez utiliser firewalld pour filtrer le trafic entre différentes zones afin d'autoriser l'accès aux machines virtuelles hébergées localement à se connecter à l'hôte.

1.7.1. La relation entre les objets de politique et les zones

Les objets de politique permettent à l'utilisateur d'attacher les primitives de firewalld telles que les services, les ports et les règles riches à la politique. Vous pouvez appliquer les objets de politique au trafic qui passe entre les zones de manière unidirectionnelle et avec état.

```
# 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 et **ANY** sont les zones symboliques utilisées dans les listes de zones d'entrée et de sortie.

- La zone symbolique **HOST** autorise les stratégies pour le trafic provenant de ou ayant une destination vers l'hôte exécutant firewalld.
- La zone symbolique **ANY** applique la politique à toutes les zones actuelles et futures. La zone symbolique **ANY** agit comme un joker pour toutes les zones.

1.7.2. Utiliser les priorités pour trier les politiques

Plusieurs politiques peuvent s'appliquer au même ensemble de trafic, c'est pourquoi les priorités doivent être utilisées pour créer un ordre de préséance pour les politiques qui peuvent être appliquées.

Pour définir une priorité afin de trier les politiques :

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

Dans l'exemple ci-dessus, *-500* est une valeur moins prioritaire, mais a une priorité plus élevée. Ainsi, -500 sera exécuté avant -100. Les valeurs de priorité supérieure ont la priorité sur les valeurs inférieures.

Les règles suivantes s'appliquent aux priorités politiques :

- Les politiques avec des priorités négatives s'appliquent avant les règles dans les zones.
- Les politiques ayant des priorités positives s'appliquent après les règles dans les zones.
- La priorité 0 est réservée et donc inutilisable.

1.7.3. Utilisation d'objets de stratégie pour filtrer le trafic entre les conteneurs hébergés localement et un réseau physiquement connecté à l'hôte

La fonction d'objets de stratégie permet aux utilisateurs de filtrer le trafic des conteneurs et des machines virtuelles.

Procédure

1. Créer une nouvelle politique.

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

2. Bloquer tout le trafic.

```
# firewall-cmd --permanent --policy podmanToHost --set-target REJECT

# firewall-cmd --permanent --policy podmanToHost --add-service dhcp

# firewall-cmd --permanent --policy podmanToHost --add-service dns
```

NOTE

Red Hat recommande de bloquer tout le trafic vers l'hôte par défaut, puis d'ouvrir sélectivement les services dont vous avez besoin pour l'hôte.

3. Définir la zone d'entrée à utiliser avec la politique.

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

4. Définir la zone de sortie à utiliser avec la politique.

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

Vérification

- Vérifier les informations relatives à la police.

```
# firewall-cmd --info-policy podmanToHost
```

1.7.4. Définition de la cible par défaut des objets de politique

Vous pouvez spécifier des options --set-target pour les politiques. Les cibles suivantes sont disponibles :

- **ACCEPT** - accepte le paquet
- **DROP** - laisse tomber les paquets indésirables
- **REJECT** - rejette les paquets non désirés avec une réponse ICMP
- **CONTINUE** (par défaut) - les paquets seront soumis aux règles des politiques et zones suivantes.

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

Vérification

- Vérifier les informations relatives à la police

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

1.8. CONFIGURATION DU NAT À L'AIDE DE FIREWALLD

Avec **firewalld**, vous pouvez configurer les types de traduction d'adresses réseau (NAT) suivants :

- Mascarade
- Source NAT (SNAT)
- NAT de destination (DNAT)
- Redirection

1.8.1. Types de NAT

Il s'agit des différents types de traduction d'adresses réseau (NAT) :

Masquerading et NAT à la source (SNAT)

Utilisez l'un de ces types de NAT pour modifier l'adresse IP source des paquets. Par exemple, les fournisseurs d'accès à Internet n'acheminent pas les plages d'adresses IP privées, telles que **10.0.0.0/8**. Si vous utilisez des plages d'adresses IP privées dans votre réseau et que les utilisateurs doivent pouvoir accéder à des serveurs sur Internet, mappez l'adresse IP source des paquets provenant de ces plages vers une adresse IP publique.
La mascarade et le SNAT sont très semblables l'un à l'autre. Les différences sont les suivantes :

- Le masquage utilise automatiquement l'adresse IP de l'interface sortante. Par conséquent, il convient d'utiliser le masquage si l'interface sortante utilise une adresse IP dynamique.
- SNAT fixe l'adresse IP source des paquets à une IP spécifiée et ne recherche pas dynamiquement l'IP de l'interface sortante. Le SNAT est donc plus rapide que le masquage. Utilisez SNAT si l'interface sortante utilise une adresse IP fixe.

NAT de destination (DNAT)

Ce type de NAT permet de réécrire l'adresse et le port de destination des paquets entrants. Par exemple, si votre serveur web utilise une adresse IP d'une plage IP privée et n'est donc pas directement accessible depuis Internet, vous pouvez définir une règle DNAT sur le routeur pour rediriger le trafic entrant vers ce serveur.

Redirection

Ce type est un cas particulier de DNAT qui redirige les paquets vers la machine locale en fonction du crochet de la chaîne. Par exemple, si un service fonctionne sur un port différent de son port standard, vous pouvez rediriger le trafic entrant du port standard vers ce port spécifique.

1.8.2. Configuration du masquage d'adresses IP

Vous pouvez activer le masquage d'IP sur votre système. Le masquage d'IP dissimule les machines individuelles derrière une passerelle lorsqu'elles accèdent à l'internet.

Procédure

1. Pour vérifier si le masquage IP est activé (par exemple, pour la zone **external**), entrez la commande suivante en tant que **root**:

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

   La commande imprime **yes** avec l'état de sortie **0** si elle est activée. Dans le cas contraire, elle affiche **no** avec l'état de sortie **1**. Si **zone** est omis, la zone par défaut sera utilisée.

2. Pour activer le masquage IP, entrez la commande suivante à l'adresse **root**:

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

3. Pour rendre ce paramètre persistant, ajoutez l'option **--permanent** à la commande.

4. Pour désactiver le masquage IP, entrez la commande suivante à l'adresse **root**:

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

Pour rendre ce paramètre permanent, ajoutez l'option **--permanent** à la commande.

1.9. UTILISATION DE DNAT POUR TRANSFÉRER LE TRAFIC HTTPS VERS UN AUTRE HÔTE

Si votre serveur web fonctionne dans une zone démilitarisée avec des adresses IP privées, vous pouvez configurer la traduction d'adresse de réseau de destination (DNAT) pour permettre aux clients sur l'internet de se connecter à ce serveur web. Dans ce cas, le nom d'hôte du serveur web est résolu en fonction de l'adresse IP publique du routeur. Lorsqu'un client établit une connexion à un port défini du routeur, ce dernier transmet les paquets au serveur web interne.

Conditions préalables

- Le serveur DNS résout le nom d'hôte du serveur web en adresse IP du routeur.
- Vous connaissez les paramètres suivants :
  - L'adresse IP privée et le numéro de port que vous souhaitez transférer
  - Le protocole IP à utiliser
  - L'adresse IP et le port de destination du serveur web où vous souhaitez rediriger les paquets

Procédure

1. Créer une politique de pare-feu :

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

   Les stratégies, contrairement aux zones, permettent le filtrage des paquets pour le trafic d'entrée, de sortie et transféré. C'est important, car le transfert du trafic vers des points d'extrémité sur des serveurs web, des conteneurs ou des machines virtuelles exécutés localement nécessite une telle capacité.

2. Configurez des zones symboliques pour le trafic entrant et sortant afin de permettre au routeur lui-même de se connecter à son adresse IP locale et de transmettre ce trafic :

```
# firewall-cmd --permanent --policy=ExamplePolicy --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=ExamplePolicy --add-egress-zone=ANY
```

   L'option **--add-ingress-zone=HOST** se réfère aux paquets générés localement, qui sont transmis à partir de l'hôte local. L'option **--add-egress-zone=ANY** fait référence au trafic destiné à n'importe quelle zone.

3. Ajoutez une règle riche qui redirige le trafic vers le serveur web :

```
# firewall-cmd --permanent --policy=ExamplePolicy --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"'
```

   La règle riche transfère le trafic TCP du port 443 de l'adresse IP 192.0.2.1 du routeur vers le port 443 de l'adresse IP 192.51.100.20 du serveur web. La règle utilise le site **ExamplePolicy** pour s'assurer que le routeur peut également se connecter à son adresse IP locale.

4. Recharger les fichiers de configuration du pare-feu :

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

5. Activer le routage de 127.0.0.0/8 dans le noyau :

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

Vérification

1. Connectez-vous à l'adresse IP du routeur et au port que vous avez transféré au serveur web :

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

2. Facultatif : Vérifiez que **net.ipv4.conf.all.route_localnet** est actif :

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

3. Vérifiez que **ExamplePolicy** est actif et contient les paramètres dont vous avez besoin. En particulier l'adresse IP et le port de la source, le protocole à utiliser, ainsi que l'adresse IP et le port de la destination :

```
# firewall-cmd --info-policy=ExamplePolicy
ExamplePolicy (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"
```

Ressources supplémentaires

- **firewall-cmd(1)**, **firewalld.policies(5)**, **firewalld.richlanguage(5)**, **sysctl(8)**, et **sysctl.conf(5)** pages de manuel
- Utilisation des fichiers de configuration dans /etc/sysctl.d/ pour ajuster les paramètres du noyau

1.10. GESTION DES REQUÊTES ICMP

Le protocole **Internet Control Message Protocol** (**ICMP**) est un protocole de support utilisé par divers dispositifs de réseau pour envoyer des messages d'erreur et des informations opérationnelles indiquant un problème de connexion, par exemple, qu'un service demandé n'est pas disponible. **ICMP** diffère des

protocoles de transport tels que TCP et UDP parce qu'il n'est pas utilisé pour échanger des données entre systèmes.

Malheureusement, il est possible d'utiliser les messages **ICMP**, en particulier **echo-request** et **echo-reply**, pour révéler des informations sur votre réseau et les utiliser à des fins frauduleuses. C'est pourquoi **firewalld** permet de bloquer les requêtes **ICMP** afin de protéger les informations de votre réseau.

1.10.1. Liste et blocage des requêtes ICMP

Listing ICMP requêtes

Les demandes **ICMP** sont décrites dans des fichiers XML individuels qui se trouvent dans le répertoire **/usr/lib/firewalld/icmptypes/**. Vous pouvez lire ces fichiers pour obtenir une description de la demande. La commande **firewall-cmd** contrôle la manipulation des demandes **ICMP**.

- Pour dresser la liste de tous les types de **ICMP** disponibles :

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

- La requête **ICMP** peut être utilisée par IPv4, IPv6 ou par les deux protocoles. Pour savoir pour quel protocole la requête **ICMP** a été utilisée :

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

- L'état d'une demande **ICMP** indique **yes** si la demande est actuellement bloquée ou **no** si elle ne l'est pas. Pour savoir si une demande **ICMP** est actuellement bloquée :

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

Blocage ou déblocage des demandes ICMP

Lorsque votre serveur bloque les requêtes **ICMP**, il ne fournit pas les informations qu'il devrait normalement fournir. Toutefois, cela ne signifie pas qu'aucune information n'est fournie. Les clients sont informés que la requête **ICMP** est bloquée (rejetée). Le blocage des requêtes **ICMP** doit être envisagé avec prudence, car il peut entraîner des problèmes de communication, en particulier avec le trafic IPv6.

- Pour savoir si une demande **ICMP** est actuellement bloquée :

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

- Pour bloquer une demande **ICMP**:

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

- Pour supprimer le blocage d'une demande **ICMP**:

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

Blocage des demandes ICMP sans fournir la moindre information

Normalement, si vous bloquez les requêtes **ICMP**, les clients savent que vous les bloquez. Ainsi, un attaquant potentiel qui recherche des adresses IP en direct peut toujours voir que votre adresse IP est en ligne. Pour masquer complètement cette information, vous devez supprimer toutes les requêtes

ICMP.

- Bloquer et abandonner toutes les demandes de **ICMP**:
- Réglez l'objectif de votre zone sur **DROP**:

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

Désormais, tout le trafic, y compris les requêtes **ICMP**, est supprimé, à l'exception du trafic que vous avez explicitement autorisé.

Pour bloquer et abandonner certaines demandes **ICMP** et en autoriser d'autres :

1. Réglez l'objectif de votre zone sur **DROP**:

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

2. Ajoutez l'inversion de bloc ICMP pour bloquer toutes les demandes **ICMP** en même temps :

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

3. Ajoutez le bloc ICMP pour les requêtes **ICMP** que vous souhaitez autoriser :

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

4. Les nouveaux paramètres doivent être conservés :

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

La commande *block inversion* inverse les paramètres de blocage des demandes de **ICMP**, de sorte que toutes les demandes qui n'étaient pas bloquées auparavant sont bloquées en raison du changement de la cible de votre zone à **DROP**. Les demandes qui étaient bloquées ne sont pas bloquées. Cela signifie que si vous souhaitez débloquer une requête, vous devez utiliser la commande de blocage.

Pour rétablir l'inversion de bloc à un niveau totalement permissif :

1. Réglez l'objectif de votre zone sur **default** ou **ACCEPT**:

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

2. Supprimer tous les blocs ajoutés pour les demandes **ICMP**:

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

3. Retirer le bloc d'inversion **ICMP**:

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

4. Les nouveaux paramètres doivent être conservés :

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

1.10.2. Configuration du filtre ICMP à l'aide de l'interface graphique

- Pour activer ou désactiver un filtre **ICMP**, démarrez l'outil **firewall-config** et sélectionnez la zone du réseau dont les messages doivent être filtrés. Sélectionnez l'onglet **ICMP Filter** et cochez la case correspondant à chaque type de message **ICMP** que vous souhaitez filtrer. Décochez la case pour désactiver un filtre. Ce réglage s'effectue par direction et la valeur par défaut autorise tout.

- Pour activer l'inversion de **ICMP Filter**, cliquez sur la case à cocher **Invert Filter** à droite. Seuls les types **ICMP** marqués sont désormais acceptés, tous les autres sont rejetés. Dans une zone utilisant la cible DROP, ils sont abandonnés.

1.11. PARAMÉTRAGE ET CONTRÔLE DES ENSEMBLES IP À L'AIDE DE FIREWALLD

Pour voir la liste des types d'ensembles IP pris en charge par **firewalld**, entrez la commande suivante en tant que root.

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

AVERTISSEMENT

Red Hat ne recommande pas l'utilisation de jeux d'adresses IP qui ne sont pas gérés par le biais de **firewalld**. Pour utiliser de tels ensembles d'adresses IP, une règle directe permanente est nécessaire pour référencer l'ensemble, et un service personnalisé doit être ajouté pour créer ces ensembles d'adresses IP. Ce service doit être lancé avant que **firewalld** ne démarre, sinon **firewalld** ne pourra pas ajouter les règles directes utilisant ces jeux. Vous pouvez ajouter des règles directes permanentes à l'aide du fichier **/etc/firewalld/direct.xml**.

1.11.1. Configuration des options du jeu IP à l'aide de la CLI

Les ensembles d'adresses IP peuvent être utilisés dans les zones **firewalld** en tant que sources et également en tant que sources dans des règles riches. Dans Red Hat Enterprise Linux, la méthode préférée consiste à utiliser les ensembles d'adresses IP créés avec **firewalld** dans une règle directe.

- Pour dresser la liste des jeux d'adresses IP connus de **firewalld** dans l'environnement permanent, utilisez la commande suivante en tant que **root**:

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

- Pour ajouter un nouveau jeu d'adresses IP, utilisez la commande suivante en utilisant l'environnement permanent comme **root**:

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

La commande précédente crée un nouveau jeu d'adresses IP avec le nom *test* et le type **hash:net** pour **IPv4**. Pour créer un jeu d'adresses IP à utiliser avec **IPv6**, ajoutez l'option **--option=family=inet6**. Pour que le nouveau paramètre prenne effet dans l'environnement d'exécution, rechargez **firewalld**.

- Dressez la liste du nouveau jeu d'adresses IP à l'aide de la commande suivante : **root**:

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

- Pour obtenir plus d'informations sur le jeu d'adresses IP, utilisez la commande suivante à l'adresse **root**:

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

Notez que le jeu d'adresses IP n'a pas d'entrées pour le moment.

- Pour ajouter une entrée à l'ensemble IP *test*, utilisez la commande suivante en tant que **root**:

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

La commande précédente ajoute l'adresse IP *192.168.0.1* au jeu d'adresses IP.

- Pour obtenir la liste des entrées actuelles du jeu d'adresses IP, utilisez la commande suivante à l'adresse **root**:

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

- Créez le fichier **iplist.txt** qui contient une liste d'adresses IP, par exemple :

```
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

Le fichier contenant la liste des adresses IP d'un ensemble d'adresses IP doit contenir une entrée par ligne. Les lignes commençant par un dièse, un point-virgule ou des lignes vides sont ignorées.

- Pour ajouter les adresses du fichier *iplist.txt*, utilisez la commande suivante en tant que **root**:

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

- Pour afficher la liste des entrées étendues de l'ensemble IP, utilisez la commande suivante à l'adresse **root**:

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

```
192.168.0.1
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

- Pour supprimer les adresses de l'ensemble IP et vérifier la liste des entrées mises à jour, utilisez les commandes suivantes comme **root**:

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

- Vous pouvez ajouter le jeu d'adresses IP en tant que source à une zone pour traiter tout le trafic provenant de n'importe quelle adresse répertoriée dans le jeu d'adresses IP avec une zone. Par exemple, pour ajouter le jeu d'adresses IP *test* en tant que source à la zone *drop* afin de supprimer tous les paquets provenant de toutes les entrées répertoriées dans le jeu d'adresses IP *test*, utilisez la commande suivante en tant que **root**:

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

Le préfixe **ipset:** dans la source indique **firewalld** que la source est un ensemble IP et non une adresse IP ou une plage d'adresses.

Seules la création et la suppression de jeux d'adresses IP sont limitées à l'environnement permanent. Toutes les autres options de jeux d'adresses IP peuvent également être utilisées dans l'environnement d'exécution sans l'option **--permanent**.

1.12. PRIORITÉ AUX RÈGLES RICHES

Par défaut, les règles riches sont organisées en fonction de leur action. Par exemple, les règles **deny** sont prioritaires sur les règles **allow**. Le paramètre **priority** des règles riches permet aux administrateurs de contrôler finement les règles riches et leur ordre d'exécution.

1.12.1. Comment le paramètre de priorité organise les règles en différentes chaînes

Dans une règle riche, le paramètre **priority** peut être défini comme n'importe quel nombre compris entre **-32768** et **32767**, les valeurs inférieures étant prioritaires.

Le service **firewalld** organise les règles en fonction de leur valeur prioritaire dans différentes chaînes :

- Priorité inférieure à 0 : la règle est redirigée vers une chaîne avec le suffixe **_pre**.
- Priorité supérieure à 0 : la règle est redirigée vers une chaîne avec le suffixe **_post**.
- Priorité égale à 0 : en fonction de l'action, la règle est redirigée vers une chaîne dont l'action est **_log**, **_deny** ou **_allow**.

À l'intérieur de ces sous-chaînes, **firewalld** trie les règles en fonction de leur valeur de priorité.

1.12.2. Définition de la priorité d'une règle riche

L'exemple suivant montre comment créer une règle riche qui utilise le paramètre **priority** pour enregistrer tout le trafic qui n'est pas autorisé ou refusé par d'autres règles. Vous pouvez utiliser cette règle pour signaler un trafic inattendu.

Procédure

- Ajoutez une règle riche avec une très faible priorité pour enregistrer tout le trafic qui n'a pas été pris en compte par d'autres règles :

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

La commande limite en outre le nombre d'entrées dans le journal à **5** par minute.

Vérification

- Affichez la règle **nftables** créée par la commande de l'étape précédente :

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

1.13. CONFIGURATION DU VERROUILLAGE DU PARE-FEU

Les applications ou services locaux peuvent modifier la configuration du pare-feu s'ils sont exécutés en tant que **root** (par exemple), **libvirt**). Cette fonction permet à l'administrateur de verrouiller la configuration du pare-feu de sorte qu'aucune application ou uniquement les applications ajoutées à la liste d'autorisation de verrouillage puissent demander des modifications du pare-feu. Les paramètres de verrouillage sont désactivés par défaut. S'ils sont activés, l'utilisateur peut être sûr qu'aucune modification indésirable n'est apportée à la configuration du pare-feu par des applications ou des services locaux.

1.13.1. Configuration du verrouillage à l'aide de l'interface de programmation

Vous pouvez activer ou désactiver la fonction de verrouillage à l'aide de la ligne de commande.

Procédure

1. Pour savoir si le verrouillage est activé, utilisez la commande suivante à l'adresse **root**:

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

La commande affiche **yes** avec l'état de sortie **0** si le verrouillage est activé. Dans le cas contraire, elle affiche **no** avec l'état de sortie **1**.

2. Pour activer le verrouillage, entrez la commande suivante à l'adresse **root**:

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

3. Pour désactiver le verrouillage, utilisez la commande suivante à l'adresse **root**:

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

1.13.2. Configuration des options de la liste d'autorisation de verrouillage à l'aide de l'interface de gestion

La liste des autorisations de verrouillage peut contenir des commandes, des contextes de sécurité, des utilisateurs et des identifiants d'utilisateur. Si une entrée de commande dans la liste d'autorisation se termine par un astérisque "*", toutes les lignes de commande commençant par cette commande seront prises en compte. Si l'astérisque n'est pas présent, la commande absolue, y compris les arguments, doit correspondre.

- Le contexte est le contexte de sécurité (SELinux) d'une application ou d'un service en cours d'exécution. Pour obtenir le contexte d'une application en cours d'exécution, utilisez la commande suivante :

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

  Cette commande renvoie toutes les applications en cours d'exécution. Faites passer la sortie par l'outil **grep** pour obtenir l'application qui vous intéresse. Par exemple :

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

- Pour dresser la liste de toutes les lignes de commande figurant dans la liste d'autorisation, entrez la commande suivante à l'adresse **root**:

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

- Pour ajouter une commande *command* à la liste des autorisations, entrez la commande suivante sous la forme **root**:

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

- Pour supprimer une commande *command* de la liste des autorisations, entrez la commande suivante sous la forme **root**:

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

- Pour savoir si la commande *command* figure dans la liste des autorisations, entrez la commande suivante sous la forme **root**:

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

  La commande affiche **yes** avec l'état de sortie **0** si elle est vraie. Elle affiche **no** avec l'état de sortie **1** dans le cas contraire.

- Pour dresser la liste de tous les contextes de sécurité figurant dans la liste d'autorisations, entrez la commande suivante à l'adresse **root**:

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

- Pour ajouter un contexte *context* à la liste des autorisations, entrez la commande suivante sous **root**:

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

- Pour supprimer un contexte *context* de la liste des autorisations, entrez la commande suivante sous **root**:

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

- Pour savoir si le contexte *context* figure dans la liste des autorisations, entrez la commande suivante en tant que **root**:

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

  Imprime **yes** avec l'état de sortie **0**, si vrai, imprime **no** avec l'état de sortie **1** sinon.

- Pour dresser la liste de tous les ID d'utilisateur figurant dans la liste d'autorisation, entrez la commande suivante à l'adresse **root**:

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

- Pour ajouter l'ID d'utilisateur *uid* à la liste d'autorisations, entrez la commande suivante sous **root**:

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

- Pour supprimer l'ID d'utilisateur *uid* de la liste des autorisations, entrez la commande suivante en tant que **root**:

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

- Pour savoir si l'ID de l'utilisateur *uid* figure dans la liste des autorisations, entrez la commande suivante :

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

  Imprime **yes** avec l'état de sortie **0**, si vrai, imprime **no** avec l'état de sortie **1** sinon.

- Pour dresser la liste de tous les noms d'utilisateur figurant dans la liste d'autorisation, entrez la commande suivante à l'adresse **root**:

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

- Pour ajouter le nom d'utilisateur *user* à la liste des autorisations, entrez la commande suivante en tant que **root**:

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

- Pour supprimer le nom d'utilisateur *user* de la liste des autorisations, entrez la commande suivante en tant que **root**:

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

- Pour savoir si le nom d'utilisateur *user* figure dans la liste des autorisations, entrez la commande suivante :

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

Imprime **yes** avec l'état de sortie **0**, si vrai, imprime **no** avec l'état de sortie **1** sinon.

1.13.3. Configuration des options de la liste d'autorisation de verrouillage à l'aide de fichiers de configuration

Le fichier de configuration de la liste d'autorisations par défaut contient le contexte **NetworkManager** et le contexte par défaut **libvirt**. L'ID utilisateur 0 figure également sur la liste.

Les fichiers de configuration d'allowlist sont stockés dans le répertoire **/etc/firewalld/**.

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

Voici un exemple de fichier de configuration allowlist autorisant toutes les commandes de l'utilitaire **firewall-cmd**, pour un utilisateur appelé *user* dont l'identifiant est **815**:

```
<?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>
```

Cet exemple montre à la fois **user id** et **user name**, mais une seule option est nécessaire. Python est l'interpréteur et est ajouté à la ligne de commande. Vous pouvez également utiliser une commande spécifique, par exemple :

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

Dans cet exemple, seule la commande **--lockdown-on** est autorisée.

Dans Red Hat Enterprise Linux, tous les utilitaires sont placés dans le répertoire **/usr/bin/** et le répertoire **/bin/** est lié par symétrie au répertoire **/usr/bin/**. En d'autres termes, bien que le chemin d'accès à **firewall-cmd**, lorsqu'il est saisi sous la forme **root**, puisse se résoudre en **/bin/firewall-cmd**, **/usr/bin/firewall-cmd** peut désormais être utilisé. Tous les nouveaux scripts doivent utiliser le nouvel emplacement. Mais attention, si les scripts qui s'exécutent sous **root** sont écrits pour utiliser le chemin **/bin/firewall-cmd**, ce chemin de commande doit être ajouté à la liste des autorisations, en plus du chemin **/usr/bin/firewall-cmd**, traditionnellement utilisé uniquement par les utilisateurs qui ne sont pas de**root**.

Le * à la fin de l'attribut name d'une commande signifie que toutes les commandes qui commencent par cette chaîne correspondent. Si le * n'est pas présent, la commande absolue, y compris les arguments, doit correspondre.

1.14. PERMETTRE LE TRANSFERT DE TRAFIC ENTRE DIFFÉRENTES INTERFACES OU SOURCES À L'INTÉRIEUR D'UNE ZONE FIREWALLD

Le transfert intra-zone est une fonctionnalité de **firewalld** qui permet de transférer le trafic entre des interfaces ou des sources à l'intérieur d'une zone **firewalld**.

1.14.1. La différence entre le transfert à l'intérieur d'une zone et les zones dont la cible par défaut est ACCEPT

Lorsque le transfert intra-zone est activé, le trafic à l'intérieur d'une seule zone **firewalld** peut circuler d'une interface ou d'une source à une autre interface ou à une autre source. La zone spécifie le niveau de confiance des interfaces et des sources. Si le niveau de confiance est identique, la communication entre les interfaces ou les sources est possible.

Notez que si vous activez le transfert intra-zone dans la zone par défaut de **firewalld**, il ne s'applique qu'aux interfaces et aux sources ajoutées à la zone par défaut actuelle.

La zone **trusted** de **firewalld** utilise une cible par défaut définie sur **ACCEPT**. Cette zone accepte tout le trafic transféré et le transfert intra-zone n'est pas applicable pour elle.

Comme pour les autres valeurs cibles par défaut, le trafic transféré est abandonné par défaut, ce qui s'applique à toutes les zones standard à l'exception de la zone de confiance.

1.14.2. Utilisation du transfert intra-zone pour transférer le trafic entre un réseau Ethernet et un réseau Wi-Fi

Vous pouvez utiliser le transfert intra-zone pour transférer le trafic entre des interfaces et des sources situées dans la même zone **firewalld**. Par exemple, utilisez cette fonctionnalité pour transférer le trafic entre un réseau Ethernet connecté à **enp1s0** et un réseau Wi-Fi connecté à **wlp0s20**.

Procédure

1. Activer le transfert de paquets dans le noyau :

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

2. Assurez-vous que les interfaces entre lesquelles vous voulez activer le transfert intra-zone ne sont pas assignées à une zone différente de la zone **internal**:

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

3. Si l'interface est actuellement affectée à une zone autre que **internal**, réaffectez-la :

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

4. Ajouter les interfaces **enp1s0** et **wlp0s20** à la zone **internal**:

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

5. Activer le transfert intra-zone :

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

-

Vérification

Les étapes de vérification suivantes requièrent que le paquetage **nmap-ncat** soit installé sur les deux hôtes.

1. Connectez-vous à un hôte qui se trouve dans le même réseau que l'interface **enp1s0** de l'hôte sur lequel vous avez activé le transfert de zone.
2. Démarrez un service d'écho avec **ncat** pour tester la connectivité :

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

3. Connectez-vous à un hôte qui se trouve dans le même réseau que l'interface **wlp0s20**.
4. Se connecter au serveur d'écho fonctionnant sur l'hôte qui se trouve dans le même réseau que **enp1s0**:

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

5. Tapez quelque chose et appuyez sur **Entrée**, puis vérifiez que le texte est renvoyé.

Ressources supplémentaires

- **firewalld.zones(5)** page de manuel

1.15. CONFIGURATION DE FIREWALLD À L'AIDE DES RÔLES DE SYSTÈME

Vous pouvez utiliser le rôle système **firewall** pour configurer les paramètres du service **firewalld** sur plusieurs clients à la fois. Cette solution :

- Fournit une interface avec des paramètres d'entrée efficaces.
- Conserve tous les paramètres de **firewalld** en un seul endroit.

Après avoir exécuté le rôle **firewall** sur le nœud de contrôle, le rôle système applique immédiatement les paramètres **firewalld** au nœud géré et les rend persistants lors des redémarrages.

1.15.1. Introduction au rôle du système RHEL firewall

RHEL System Roles est un ensemble de contenus pour l'utilitaire d'automatisation Ansible. Ce contenu, associé à l'utilitaire d'automatisation Ansible, fournit une interface de configuration cohérente pour gérer à distance plusieurs systèmes.

Le rôle **rhel-system-roles.firewall** des rôles système RHEL a été introduit pour les configurations automatisées du service **firewalld**. Le paquetage **rhel-system-roles** contient ce rôle système, ainsi que la documentation de référence.

Pour appliquer les paramètres **firewalld** à un ou plusieurs systèmes de manière automatisée, utilisez la variable **firewall** System Role dans un cahier de jeu. Un cahier de jeu est une liste d'un ou plusieurs jeux écrits au format YAML basé sur le texte.

Vous pouvez utiliser un fichier d'inventaire pour définir un ensemble de systèmes qu'Ansible doit configurer.

Le rôle **firewall** vous permet de configurer de nombreux paramètres **firewalld**, par exemple :

- Zones.
- Les services pour lesquels les paquets doivent être autorisés.
- Octroi, rejet ou abandon de l'accès du trafic aux ports.
- Transfert de ports ou de plages de ports pour une zone.

Ressources supplémentaires

- **README.md** et **README.html** dans le répertoire **/usr/share/doc/rhel-system-roles/firewall/**
- Travailler avec des playbooks
- Comment constituer votre inventaire

1.15.2. Réinitialisation des paramètres de firewalld à l'aide du rôle de système RHEL du pare-feu

Avec le rôle système **firewall** RHEL, vous pouvez réinitialiser les paramètres **firewalld** à leur état par défaut. Si vous ajoutez le paramètre **previous:replaced** à la liste des variables, le rôle système supprime tous les paramètres existants définis par l'utilisateur et rétablit les valeurs par défaut de **firewalld**. Si vous combinez le paramètre **previous:replaced** avec d'autres paramètres, le rôle **firewall** supprime tous les paramètres existants avant d'en appliquer de nouveaux.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/reset-firewalld.yml***avec le contenu suivant :

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
  - name: Reset firewalld
    include_role:
      name: rhel-system-roles.firewall
```

```
    vars:
      firewall:
        - previous: replaced
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configuring-a-dmz.yml
```

Vérification

- Exécutez cette commande en tant que **root** sur le nœud géré pour vérifier toutes les zones :

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

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**
- **ansible-playbook(1)**
- **firewalld(1)**

1.15.3. Transférer le trafic entrant d'un port local vers un autre port local

Le rôle **firewall** vous permet de configurer à distance les paramètres **firewalld** avec un effet persistant sur plusieurs hôtes gérés.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/port_forwarding.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Forward incoming traffic on port 8080 to 443
    include_role:
      name: rhel-system-roles.firewall
```

```
    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. Exécutez le manuel de jeu :

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

Vérification

- Sur l'hôte géré, affichez les paramètres **firewalld**:

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

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

1.15.4. Configuration des ports à l'aide des rôles de système

Vous pouvez utiliser le rôle système RHEL **firewall** pour ouvrir ou fermer les ports du pare-feu local pour le trafic entrant et faire en sorte que la nouvelle configuration persiste lors des redémarrages. Par exemple, vous pouvez configurer la zone par défaut pour autoriser le trafic entrant pour le service HTTPS.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/opening-a-port.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Allow incoming HTTPS traffic to the local host
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
```

```
    - port: 443/tcp
      service: http
      state: enabled
      runtime: true
      permanent: true
```

L'option **permanent: true** rend les nouveaux paramètres persistants à travers les redémarrages.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/opening-a-port.yml
```

Vérification

- Sur le nœud géré, vérifiez que le port **443/tcp** associé au service **HTTPS** est ouvert :

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

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

1.15.5. Configuration d'une zone DMZ firewalld à l'aide du rôle de système firewalld RHEL

En tant qu'administrateur système, vous pouvez utiliser le rôle système **firewall** pour configurer une zone **dmz** sur l'interface **enp1s0** afin d'autoriser le trafic **HTTPS** vers la zone. Vous permettez ainsi à des utilisateurs externes d'accéder à vos serveurs web.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/configuring-a-dmz.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
   - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
```

```
  include_role:
    name: rhel-system-roles.firewall

  vars:
    firewall:
      - zone: dmz
        interface: enp1s0
        service: https
        state: enabled
        runtime: true
        permanent: true
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configuring-a-dmz.yml
```

Vérification

- Sur le nœud géré, affichez des informations détaillées sur la zone **dmz**:

```
# 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:
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

1.16. RESSOURCES SUPPLÉMENTAIRES

- **firewalld(1)** page de manuel
- **firewalld.conf(5)** page de manuel
- **firewall-cmd(1)** page de manuel
- **firewall-config(1)** page de manuel
- **firewall-offline-cmd(1)** page de manuel
- **firewalld.icmptype(5)** page de manuel
- **firewalld.ipset(5)** page de manuel

- **firewalld.service(5)** page de manuel
- **firewalld.zone(5)** page de manuel
- **firewalld.direct(5)** page de manuel
- **firewalld.lockdown-whitelist(5)**
- **firewalld.richlanguage(5)**
- **firewalld.zones(5)** page de manuel
- **firewalld.dbus(5)** page de manuel

CHAPITRE 2. DÉMARRER AVEC NFTABLES

Le cadre **nftables** classe les paquets et succède aux utilitaires **iptables**, **ip6tables**, **arptables**, **ebtables** et **ipset**. Il offre de nombreuses améliorations en termes de commodité, de fonctionnalités et de performances par rapport aux outils de filtrage de paquets précédents, notamment :

- Tableaux de consultation intégrés au lieu d'un traitement linéaire
- Un cadre unique pour les protocoles **IPv4** et **IPv6**
- Toutes les règles sont appliquées de manière atomique au lieu d'aller chercher, de mettre à jour et de stocker un ensemble complet de règles
- Prise en charge du débogage et du traçage dans l'ensemble de règles (**nftrace**) et surveillance des événements de traçage (dans l'outil **nft**)
- Syntaxe plus cohérente et plus compacte, pas d'extensions spécifiques au protocole
- Une API Netlink pour les applications tierces

Le cadre **nftables** utilise des tableaux pour stocker les chaînes. Les chaînes contiennent des règles individuelles pour effectuer des actions. L'utilitaire **nft** remplace tous les outils des cadres de filtrage de paquets précédents. Vous pouvez utiliser la bibliothèque **libnftnl** pour une interaction de bas niveau avec **nftables** Netlink API par le biais de la bibliothèque **libmnl**.

Pour afficher l'effet des modifications apportées au jeu de règles, utilisez la commande **nft list ruleset**. Comme ces utilitaires ajoutent des tables, des chaînes, des règles, des jeux et d'autres objets au jeu de règles **nftables**, sachez que les opérations sur le jeu de règles **nftables**, telles que la commande **nft flush ruleset**, peuvent affecter les jeux de règles installés à l'aide de la commande **iptables**.

2.1. MIGRER D'IPTABLES À NFTABLES

Si votre configuration de pare-feu utilise encore les règles **iptables**, vous pouvez migrer vos règles **iptables** vers **nftables**.

IMPORTANT

Les paquetages **ipset** et **iptables-nft** ont été dépréciés dans Red Hat Enterprise Linux 9. Cela inclut la dépréciation de **nft-variants** tels que les utilitaires **iptables**, **ip6tables**, **arptables**, et **ebtables**. Si vous utilisez l'un de ces outils, par exemple, parce que vous avez effectué une mise à niveau à partir d'une version antérieure de RHEL, Red Hat vous recommande de migrer vers l'outil de ligne de commande **nft** fourni par le paquetage **nftables**.

2.1.1. Quand utiliser firewalld, nftables ou iptables ?

Voici un bref aperçu des scénarios dans lesquels vous devez utiliser l'un des utilitaires suivants :

- **firewalld**: Utilisez l'utilitaire **firewalld** pour des cas d'utilisation simples de pare-feu. L'utilitaire est facile à utiliser et couvre les cas d'utilisation typiques de ces scénarios.
- **nftables**: Utilisez l'utilitaire **nftables** pour configurer des pare-feu complexes et critiques en termes de performances, par exemple pour l'ensemble d'un réseau.
- **iptables**: L'utilitaire **iptables** sur Red Hat Enterprise Linux utilise l'API du noyau **nf_tables** au

lieu de l'interface **legacy**. L'API **nf_tables** offre une compatibilité ascendante de sorte que les scripts qui utilisent les commandes **iptables** fonctionnent toujours sur Red Hat Enterprise Linux. Pour les nouveaux scripts de pare-feu, Red Hat recommande d'utiliser **nftables**.

IMPORTANT

Pour éviter que les différents services de pare-feu ne s'influencent mutuellement, exécutez un seul d'entre eux sur un hôte RHEL et désactivez les autres services.

2.1.2. Conversion des jeux de règles iptables et ip6tables en nftables

Utilisez les utilitaires **iptables-restore-translate** et **ip6tables-restore-translate** pour traduire les jeux de règles **iptables** et **ip6tables** en **nftables**.

Conditions préalables

- Les paquets **nftables** et **iptables** sont installés.
- Le système a configuré les règles **iptables** et **ip6tables**.

Procédure

1. Inscrivez les règles **iptables** et **ip6tables** dans un fichier :

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

2. Convertir les fichiers dump en instructions **nftables**:

```
# 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. Examinez et, si nécessaire, mettez à jour manuellement les règles générées sur **nftables**.
4. Pour permettre au service **nftables** de charger les fichiers générés, ajoutez ce qui suit au fichier **/etc/sysconfig/nftables.conf**:

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

5. Arrêtez et désactivez le service **iptables**:

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

Si vous avez utilisé un script personnalisé pour charger les règles **iptables**, assurez-vous que le script ne démarre plus automatiquement et redémarrez pour vider toutes les tables.

6. Activez et démarrez le service **nftables**:

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

Vérification

- Affichez l'ensemble de règles **nftables**:

```
# nft list ruleset
```

Ressources supplémentaires

- Chargement automatique des règles nftables au démarrage du système

2.1.3. Conversion de règles iptables et ip6tables en règles nftables

Red Hat Enterprise Linux fournit les utilitaires **iptables-translate** et **ip6tables-translate** pour convertir une règle **iptables** ou **ip6tables** en une règle équivalente pour **nftables**.

Conditions préalables

- Le paquet **nftables** est installé.

Procédure

- Utilisez l'utilitaire **iptables-translate** ou **ip6tables-translate** au lieu de **iptables** ou **ip6tables** pour afficher la règle **nftables** correspondante, par exemple :

```
# 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
```

Notez que certaines extensions ne prennent pas en charge la traduction. Dans ce cas, l'utilitaire imprime la règle non traduite préfixée par le signe **#**, par exemple :

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

Ressources supplémentaires

- **iptables-translate --help**

2.1.4. Comparaison des commandes iptables et nftables les plus courantes

Voici une comparaison des commandes courantes **iptables** et **nftables**:

- Liste de toutes les règles :

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

- L'inscription d'une certaine table et d'une certaine chaîne :

iptables	nftables
iptables -L	**nft list table ip filter**

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

La commande **nft** ne précrée pas les tables et les chaînes. Elles n'existent que si un utilisateur les a créées manuellement.

Liste des règles générées par firewalld :

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

2.1.5. Ressources supplémentaires

- iptables : Les deux variantes et leur relation avec nftables

2.2. RÉDACTION ET EXÉCUTION DE SCRIPTS NFTABLES

Le principal avantage de l'utilisation du cadre **nftables`** est que l'exécution des scripts est atomique. Cela signifie que le système applique l'intégralité du script ou empêche l'exécution en cas d'erreur. Cela garantit que le pare-feu est toujours dans un état cohérent.

En outre, avec l'environnement de script **nftables**, vous pouvez :

- Ajouter des commentaires
- Définir les variables
- Inclure d'autres fichiers de jeux de règles

Lorsque vous installez le paquetage **nftables**, Red Hat Enterprise Linux crée automatiquement les scripts ***.nft** dans le répertoire **/etc/nftables/**. Ces scripts contiennent des commandes qui créent des tables et des chaînes vides à différentes fins.

2.2.1. Formats de scripts nftables pris en charge

Dans l'environnement de script **nftables**, vous pouvez écrire des scripts dans les formats suivants :

- Le même format que la commande **nft list ruleset** affiche le jeu de règles :

```
#!/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
  }
}
```

- La syntaxe est la même que pour les commandes **nft**:

```
#!/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. Exécution de scripts nftables

Vous pouvez exécuter un script **nftables** en le passant à l'utilitaire **nft** ou en l'exécutant directement.

Procédure

- Pour exécuter un script **nftables** en le passant à l'utilitaire **nft**, entrez :

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

- Pour exécuter directement un script **nftables**:

  a. Pour chaque fois que vous effectuez cette opération :

     i. Veillez à ce que le script commence par la séquence shebang suivante :

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

IMPORTANT

Si vous omettez le paramètre **-f**, l'utilitaire **nft** ne lit pas le script et affiche : **Error: syntax error, unexpected newline, expecting string**.

     ii. Optionnel : Définissez le propriétaire du script sur **root**:

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

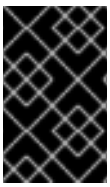
     iii. Rendre le script exécutable pour le propriétaire :

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

b. Exécutez le script :

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

Si aucune sortie n'est affichée, le système a exécuté le script avec succès.



IMPORTANT

Même si **nft** exécute le script avec succès, des règles mal placées, des paramètres manquants ou d'autres problèmes dans le script peuvent faire en sorte que le pare-feu ne se comporte pas comme prévu.

Ressources supplémentaires

- **chown(1)** page de manuel
- **chmod(1)** page de manuel
- Chargement automatique des règles nftables au démarrage du système

2.2.3. Utilisation de commentaires dans les scripts nftables

L'environnement de script **nftables** interprète comme un commentaire tout ce qui se trouve à droite d'un caractère **#** jusqu'à la fin d'une ligne.

Les commentaires peuvent commencer au début d'une ligne ou à côté d'une commande :

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

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

2.2.4. Utilisation de variables dans un script nftables

Pour définir une variable dans un script **nftables**, utilisez le mot-clé **define**. Vous pouvez stocker des valeurs individuelles et des ensembles anonymes dans une variable. Pour des scénarios plus complexes, utilisez des ensembles ou des cartes de verdict.

Variables à valeur unique

L'exemple suivant définit une variable nommée **INET_DEV** avec la valeur **enp1s0**:

```
define INET_DEV = enp1s0
```

Vous pouvez utiliser la variable dans le script en saisissant le signe **$** suivi du nom de la variable :

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

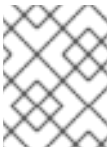
Variables contenant un ensemble anonyme

L'exemple suivant définit une variable qui contient un ensemble anonyme :

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

Vous pouvez utiliser la variable dans le script en écrivant le signe **$** suivi du nom de la variable :

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



NOTE

Les accolades ont une sémantique particulière lorsque vous les utilisez dans une règle, car elles indiquent que la variable représente un ensemble.

Ressources supplémentaires

- Utilisation des ensembles dans les commandes nftables
- Utilisation des cartes de verdict dans les commandes nftables

2.2.5. Inclusion de fichiers dans les scripts nftables

Dans l'environnement de script **nftables**, vous pouvez inclure d'autres scripts en utilisant l'instruction **include**.

Si vous ne spécifiez qu'un nom de fichier sans chemin absolu ou relatif, **nftables** inclut les fichiers du chemin de recherche par défaut, qui est défini sur **/etc** sur Red Hat Enterprise Linux.

Exemple 2.1. Inclure des fichiers du répertoire de recherche par défaut

Pour inclure un fichier du répertoire de recherche par défaut :

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

Exemple 2.2. Inclure tous les fichiers *.nft d'un répertoire

Pour inclure tous les fichiers se terminant par ***.nft** qui sont stockés dans le répertoire **/etc/nftables/rulesets/**:

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

Notez que l'instruction **include** ne correspond pas aux fichiers commençant par un point.

Ressources supplémentaires

- La section **Include files** dans la page de manuel **nft(8)**

2.2.6. Chargement automatique des règles nftables au démarrage du système

Le service systemd **nftables** charge les scripts de pare-feu inclus dans le fichier **/etc/sysconfig/nftables.conf**.

Conditions préalables

- Les scripts **nftables** sont stockés dans le répertoire **/etc/nftables/**.

Procédure

1. Modifiez le fichier **/etc/sysconfig/nftables.conf**.
   - Si vous avez modifié les scripts ***.nft** créés dans **/etc/nftables/** lors de l'installation du paquet **nftables**, décommentez la déclaration **include** pour ces scripts.
   - Si vous avez écrit de nouveaux scripts, ajoutez les instructions **include** pour inclure ces scripts. Par exemple, pour charger le script **/etc/nftables/*example*.nft** au démarrage du service **nftables**, ajoutez :

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

2. Facultatif : Démarrez le service **nftables** pour charger les règles du pare-feu sans redémarrer le système :

   ```
   # systemctl start nftables
   ```

3. Activer le service **nftables**.

   ```
   # systemctl enable nftables
   ```

Ressources supplémentaires

- Formats de scripts nftables pris en charge

2.3. CRÉATION ET GESTION DES TABLES, CHAÎNES ET RÈGLES NFTABLES

Vous pouvez afficher les ensembles de règles **nftables** et les gérer.

2.3.1. Notions de base sur les tableaux nftables

Sur le site **nftables**, une table est un espace de noms qui contient une collection de chaînes, de règles, d'ensembles et d'autres objets.

Une famille d'adresses doit être attribuée à chaque table. La famille d'adresses définit les types de paquets que cette table traite. Vous pouvez définir l'une des familles d'adresses suivantes lorsque vous créez une table :

- **ip**: Correspond uniquement aux paquets IPv4. Il s'agit de la valeur par défaut si vous ne spécifiez pas de famille d'adresses.
- **ip6**: Correspond uniquement aux paquets IPv6.
- **inet**: Correspond aux paquets IPv4 et IPv6.

- **arp**: Correspond aux paquets du protocole de résolution d'adresses (ARP) IPv4.
- **bridge**: Correspond aux paquets qui passent par un pont.
- **netdev**: Correspond aux paquets provenant de l'entrée.

Si vous souhaitez ajouter un tableau, le format à utiliser dépend de votre script de pare-feu :

- Dans les scripts en syntaxe native, utilisez :

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

- Dans les scripts shell, utilisez :

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

2.3.2. Les bases des chaînes nftables

Les tableaux sont constitués de chaînes qui, à leur tour, contiennent des règles. Les deux types de règles suivants existent :

- **Base chain**: Vous pouvez utiliser les chaînes de base comme point d'entrée pour les paquets provenant de la pile réseau.
- **Regular chain**: Vous pouvez utiliser les chaînes régulières comme cible **jump** pour mieux organiser les règles.

Si vous souhaitez ajouter une chaîne de base à un tableau, le format à utiliser dépend de votre script de pare-feu :

- Dans les scripts en syntaxe native, utilisez :

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

- Dans les scripts shell, utilisez :

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

  Pour éviter que l'interpréteur de commandes n'interprète les points-virgules comme la fin de la commande, placez le caractère d'échappement \ devant les points-virgules.

Les deux exemples créent **base chains**. Pour créer un **regular chain**, ne définissez aucun paramètre entre les crochets.

Types de chaînes
Vous trouverez ci-dessous les types de chaînes et un aperçu des familles d'adresses et des crochets avec lesquels vous pouvez les utiliser :

Type	Adresses des familles	Crochets	Description
filter	tous	tous	Type de chaîne standard
nat	**ip**, **ip6**, **inet**	**prerouting**, **input**, **output**, **postrouting**	Les chaînes de ce type effectuent une traduction d'adresse native basée sur les entrées de suivi de connexion. Seul le premier paquet traverse ce type de chaîne.
route	**ip**, **ip6**	**output**	Les paquets acceptés qui traversent ce type de chaîne entraînent une nouvelle recherche d'itinéraire si les parties concernées de l'en-tête IP ont été modifiées.

Priorités de la chaîne

Le paramètre de priorité spécifie l'ordre dans lequel les paquets traversent les chaînes ayant la même valeur de crochet. Vous pouvez donner à ce paramètre une valeur entière ou utiliser un nom de priorité standard.

La matrice suivante donne un aperçu des noms de priorité standard et de leurs valeurs numériques, ainsi que des familles d'adresses et des crochets avec lesquels vous pouvez les utiliser :

Valeur textuelle	Valeur numérique	Adresses des familles	Crochets
raw	**-300**	**ip**, **ip6**, **inet**	tous
mangle	**-150**	**ip**, **ip6**, **inet**	tous
dstnat	**-100**	**ip**, **ip6**, **inet**	**prerouting**
	-300	**bridge**	**prerouting**
filter	**0**	**ip**, **ip6**, **inet**, **arp**, **netdev**	tous
	-200	**bridge**	tous
security	**50**	**ip**, **ip6**, **inet**	tous
srcnat	**100**	**ip**, **ip6**, **inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**output**

Politiques de la chaîne

La politique de chaîne définit si **nftables** doit accepter ou abandonner les paquets si les règles de cette chaîne ne spécifient aucune action. Vous pouvez définir l'une des politiques suivantes dans une chaîne :

- **accept** (par défaut)
- **drop**

2.3.3. Règles de base de nftables

Les règles définissent les actions à effectuer sur les paquets qui passent une chaîne contenant cette règle. Si la règle contient également des expressions correspondantes, **nftables** n'effectue les actions que si toutes les expressions précédentes s'appliquent.

Si vous souhaitez ajouter une règle à une chaîne, le format à utiliser dépend de votre script de pare-feu :

- Dans les scripts en syntaxe native, utilisez :

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

- Dans les scripts shell, utilisez :

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

Cette commande shell ajoute la nouvelle règle à la fin de la chaîne. Si vous préférez ajouter une règle au début de la chaîne, utilisez la commande **nft insert** au lieu de **nft add**.

2.3.4. Gestion des tables, des chaînes et des règles à l'aide des commandes nft

Pour gérer un pare-feu **nftables** sur la ligne de commande ou dans des scripts shell, utilisez l'utilitaire **nft**.

IMPORTANT

Les commandes de cette procédure ne représentent pas un flux de travail typique et ne sont pas optimisées. Cette procédure montre seulement comment utiliser les commandes **nft** pour gérer les tables, les chaînes et les règles en général.

Procédure

1. Créez une table nommée **nftables_svc** avec la famille d'adresses **inet** afin que la table puisse traiter les paquets IPv4 et IPv6 :

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

2. Ajoutez une chaîne de base nommée **INPUT**, qui traite le trafic réseau entrant, à la table **inet nftables_svc**:

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

Pour éviter que l'interpréteur de commandes n'interprète les points-virgules comme la fin de la commande, échappez les points-virgules à l'aide du caractère \.

3. Ajoutez des règles à la chaîne **INPUT**. Par exemple, autoriser le trafic TCP entrant sur les ports 22 et 443 et, en tant que dernière règle de la chaîne **INPUT**, rejeter tout autre trafic entrant avec un message ICMP (Internet Control Message Protocol) de port inaccessible :

```
# 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
```

   Si vous saisissez les commandes **nft add rule** comme indiqué, **nft** ajoute les règles dans le même ordre à la chaîne que celui dans lequel vous exécutez les commandes.

4. Affiche l'ensemble de règles actuel, y compris les poignées :

```
# 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. Insérez une règle avant la règle existante avec la poignée 3. Par exemple, pour insérer une règle qui autorise le trafic TCP sur le port 636, entrez :

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

6. Ajoutez une règle après la règle existante avec la poignée 3. Par exemple, pour insérer une règle qui autorise le trafic TCP sur le port 80, entrez :

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

7. Affichez à nouveau l'ensemble de règles avec les poignées. Vérifiez que les règles ajoutées ultérieurement ont été ajoutées aux positions spécifiées :

```
# 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. Retirer la règle à l'aide de la poignée 6 :

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

   Pour supprimer une règle, vous devez spécifier la poignée.

9. Affichez le jeu de règles et vérifiez que la règle supprimée n'est plus présente :

```
# 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. Supprimer toutes les règles restantes de la chaîne **INPUT**:

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

11. Affichez l'ensemble de règles et vérifiez que la chaîne **INPUT** est vide :

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

12. Supprimer la chaîne **INPUT**:

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

Vous pouvez également utiliser cette commande pour supprimer les chaînes qui contiennent encore des règles.

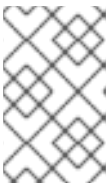
13. Affichez le jeu de règles et vérifiez que la chaîne **INPUT** a été supprimée :

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

14. Supprimez le tableau **nftables_svc**:

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

Vous pouvez également utiliser cette commande pour supprimer les tables qui contiennent encore des chaînes.



NOTE

Pour supprimer l'ensemble des règles, utilisez la commande **nft flush ruleset** au lieu de supprimer manuellement toutes les règles, chaînes et tables dans des commandes distinctes.

Ressources supplémentaires

nft(8) page de manuel

2.4. CONFIGURATION DU NAT À L'AIDE DE NFTABLES

Avec **nftables**, vous pouvez configurer les types de traduction d'adresses réseau (NAT) suivants :

- Mascarade
- Source NAT (SNAT)
- NAT de destination (DNAT)
- Redirection

IMPORTANT

Vous ne pouvez utiliser que des noms d'interface réels dans les paramètres **iifname** et **oifname**. Les noms alternatifs (**altname**) ne sont pas pris en charge.

2.4.1. Types de NAT

Il s'agit des différents types de traduction d'adresses réseau (NAT) :

Masquerading et NAT à la source (SNAT)

Utilisez l'un de ces types de NAT pour modifier l'adresse IP source des paquets. Par exemple, les fournisseurs d'accès à Internet n'acheminent pas les plages d'adresses IP privées, telles que **10.0.0.0/8**. Si vous utilisez des plages d'adresses IP privées dans votre réseau et que les utilisateurs doivent pouvoir accéder à des serveurs sur Internet, mappez l'adresse IP source des paquets provenant de ces plages vers une adresse IP publique.
La mascarade et le SNAT sont très semblables l'un à l'autre. Les différences sont les suivantes :

- Le masquage utilise automatiquement l'adresse IP de l'interface sortante. Par conséquent, il convient d'utiliser le masquage si l'interface sortante utilise une adresse IP dynamique.
- SNAT fixe l'adresse IP source des paquets à une IP spécifiée et ne recherche pas dynamiquement l'IP de l'interface sortante. Le SNAT est donc plus rapide que le masquage. Utilisez SNAT si l'interface sortante utilise une adresse IP fixe.

NAT de destination (DNAT)

Ce type de NAT permet de réécrire l'adresse et le port de destination des paquets entrants. Par exemple, si votre serveur web utilise une adresse IP d'une plage IP privée et n'est donc pas directement accessible depuis Internet, vous pouvez définir une règle DNAT sur le routeur pour rediriger le trafic entrant vers ce serveur.

Redirection

Ce type est un cas particulier de DNAT qui redirige les paquets vers la machine locale en fonction du crochet de la chaîne. Par exemple, si un service fonctionne sur un port différent de son port standard, vous pouvez rediriger le trafic entrant du port standard vers ce port spécifique.

2.4.2. Configuration du masquage à l'aide de nftables

Le masquage permet à un routeur de changer dynamiquement l'adresse IP source des paquets envoyés par une interface pour l'adresse IP de l'interface. Cela signifie que si l'interface se voit attribuer une nouvelle IP, **nftables** utilise automatiquement la nouvelle IP pour remplacer l'IP source.

Remplacer l'IP source des paquets quittant l'hôte via l'interface **ens3** par l'IP définie sur **ens3**.

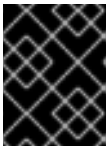
Procédure

1. Créer un tableau :

```
# nft add table nat
```

2. Ajoutez les chaînes **prerouting** et **postrouting** au tableau :

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



IMPORTANT

Même si vous n'ajoutez pas de règle à la chaîne **prerouting**, le cadre **nftables** exige que cette chaîne corresponde aux réponses des paquets entrants.

Notez que vous devez passer l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **postrouting** qui correspond aux paquets sortants sur l'interface **ens3**:

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

2.4.3. Configuration de la NAT à la source à l'aide de nftables

Sur un routeur, le NAT de source (SNAT) vous permet de changer l'IP des paquets envoyés par une interface en une adresse IP spécifique. Le routeur remplace alors l'IP source des paquets sortants.

Procédure

1. Créer un tableau :

```
# nft add table nat
```

2. Ajoutez les chaînes **prerouting** et **postrouting** au tableau :

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

IMPORTANT

Même si vous n'ajoutez pas de règle à la chaîne **postrouting**, le cadre **nftables** exige que cette chaîne corresponde aux réponses des paquets sortants.

Notez que vous devez passer l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **postrouting** qui remplace l'IP source des paquets sortants via **ens3** par **192.0.2.1**:

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

Ressources supplémentaires

- Transférer les paquets entrants sur un port local spécifique vers un autre hôte

2.4.4. Configuration de la NAT de destination à l'aide de nftables

Le NAT de destination (DNAT) permet de rediriger le trafic d'un routeur vers un hôte qui n'est pas directement accessible depuis l'internet.

Par exemple, avec DNAT, le routeur redirige le trafic entrant envoyé aux ports **80** et **443** vers un serveur web dont l'adresse IP est **192.0.2.1**.

Procédure

1. Créer un tableau :

```
# nft add table nat
```

2. Ajoutez les chaînes **prerouting** et **postrouting** au tableau :

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

IMPORTANT

Même si vous n'ajoutez pas de règle à la chaîne **postrouting**, le cadre **nftables** exige que cette chaîne corresponde aux réponses des paquets sortants.

Notez que vous devez passer l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **prerouting** qui redirige le trafic entrant vers les ports **80** et **443** sur l'interface **ens3** du routeur vers le serveur web avec l'adresse IP **192.0.2.1**:"

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

4. En fonction de votre environnement, ajoutez une règle SNAT ou de masquage pour modifier l'adresse source des paquets renvoyés par le serveur web à l'expéditeur :

   a. Si l'interface **ens3** utilise une adresse IP dynamique, ajoutez une règle de masquage :

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

   b. Si l'interface **ens3** utilise une adresse IP statique, ajoutez une règle SNAT. Par exemple, si l'interface **ens3** utilise l'adresse IP **198.51.100.1**:

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

5. Activer le transfert de paquets :

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

Ressources supplémentaires

- Types de NAT

2.4.5. Configuration d'une redirection à l'aide de nftables

La fonction **redirect** est un cas particulier de traduction d'adresse de réseau de destination (DNAT) qui redirige les paquets vers la machine locale en fonction du crochet de la chaîne.

Par exemple, vous pouvez rediriger le trafic entrant et transféré envoyé au port **22** de l'hôte local vers le port **2222**.

Procédure

1. Créer un tableau :

```
# nft add table nat
```

2. Ajoutez la chaîne **prerouting** à la table :

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

Notez que vous devez passer l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **prerouting** qui redirige le trafic entrant sur le port **22** vers le port **2222**:

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

Ressources supplémentaires

- Types de NAT

2.5. UTILISATION DES ENSEMBLES DANS LES COMMANDES NFTABLES

Le cadre **nftables** prend en charge les ensembles de manière native. Vous pouvez utiliser des ensembles, par exemple, si une règle doit correspondre à plusieurs adresses IP, numéros de port, interfaces ou tout autre critère de correspondance.

2.5.1. Utilisation d'ensembles anonymes dans nftables

Un ensemble anonyme contient des valeurs séparées par des virgules et placées entre crochets, telles que **{ 22, 80, 443 }**, que vous utilisez directement dans une règle. Vous pouvez également utiliser des ensembles anonymes pour les adresses IP et tout autre critère de correspondance.

L'inconvénient des ensembles anonymes est que si vous souhaitez modifier l'ensemble, vous devez remplacer la règle. Pour une solution dynamique, utilisez des ensembles nommés, comme décrit dans la section Utilisation d'ensembles nommés dans les tables nft .

Conditions préalables

- La chaîne **example_chain** et la table **example_table** de la famille **inet** existent.

Procédure

1. Par exemple, pour ajouter une règle à **example_chain** dans **example_table** qui autorise le trafic entrant vers les ports **22**, **80** et **443**:

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

2. Optionnel : Afficher toutes les chaînes et leurs règles sur **example_table**:

```
# 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. Utilisation d'ensembles nommés dans nftables

Le cadre **nftables** prend en charge les ensembles nommés mutables. Un ensemble nommé est une liste ou une plage d'éléments que vous pouvez utiliser dans plusieurs règles au sein d'une table. Un autre avantage par rapport aux ensembles anonymes est que vous pouvez mettre à jour un ensemble nommé sans remplacer les règles qui l'utilisent.

Lorsque vous créez un ensemble nommé, vous devez spécifier le type d'éléments qu'il contient. Vous pouvez définir les types suivants :

- **ipv4_addr** pour un ensemble contenant des adresses ou des plages IPv4, telles que **192.0.2.1** ou **192.0.2.0/24**.
- **ipv6_addr** pour un ensemble contenant des adresses ou des plages IPv6, telles que **2001:db8:1::1** ou **2001:db8:1::1/64**.
- **ether_addr** pour un ensemble contenant une liste d'adresses de contrôle d'accès au support (MAC), tel que **52:54:00:6b:66:42**.
- **inet_proto** pour un ensemble contenant une liste de types de protocoles Internet, tels que **tcp**.
- **inet_service** pour un ensemble contenant une liste de services Internet, tels que **ssh**.
- **mark** pour un ensemble contenant une liste de marques de paquets. Les marques de paquets peuvent être des valeurs entières positives de 32 bits (**0** à **2147483647**).

Conditions préalables

- La chaîne **example_chain** et la table **example_table** existent.

Procédure

1. Créez un ensemble vide. Les exemples suivants créent un ensemble pour les adresses IPv4 :

   - Pour créer un ensemble pouvant stocker plusieurs adresses IPv4 individuelles :

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

   - Pour créer un ensemble capable de stocker des plages d'adresses IPv4 :

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

     IMPORTANT

     Pour éviter que l'interpréteur de commandes n'interprète les points-virgules comme la fin de la commande, vous devez les faire précéder d'une barre oblique inverse.

2. Facultatif : Créez des règles qui utilisent l'ensemble. Par exemple, la commande suivante ajoute une règle à l'ensemble **example_chain** dans l'ensemble **example_table** qui supprimera tous les paquets provenant des adresses IPv4 de l'ensemble **example_set**.

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

   Comme **example_set** est toujours vide, la règle n'a actuellement aucun effet.

3. Ajouter des adresses IPv4 à **example_set**:

   - Si vous créez un ensemble qui stocke des adresses IPv4 individuelles, entrez :

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

   - Si vous créez un ensemble qui stocke des plages IPv4, entrez :

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

     Lorsque vous spécifiez une plage d'adresses IP, vous pouvez également utiliser la notation CIDR (Classless Inter-Domain Routing), comme **192.0.2.0/24** dans l'exemple ci-dessus.

2.5.3. Ressources supplémentaires

- La section **Sets** dans la page de manuel **nft(8)**

2.6. UTILISATION DES CARTES DE VERDICT DANS LES COMMANDES NFTABLES

Les cartes de verdict, également connues sous le nom de dictionnaires, permettent à **nft** d'effectuer une action basée sur les informations des paquets en associant des critères de correspondance à une action.

2.6.1. Utilisation de cartes anonymes dans nftables

Une carte anonyme est une instruction **{ *match_criteria* : *action* }** que vous utilisez directement dans une règle. La déclaration peut contenir plusieurs mappings séparés par des virgules.

L'inconvénient d'une table anonyme est que si vous souhaitez modifier la table, vous devez remplacer la règle. Pour une solution dynamique, utilisez des cartes nommées, comme décrit dans la section Utilisation de cartes nommées dans les tables nft .

Par exemple, vous pouvez utiliser une carte anonyme pour acheminer les paquets TCP et UDP des protocoles IPv4 et IPv6 vers des chaînes différentes afin de compter séparément les paquets TCP et UDP entrants.

Procédure

1. Créer un nouveau tableau :

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

2. Créez la chaîne **tcp_packets** dans **example_table**:

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

3. Ajoutez une règle à **tcp_packets** qui compte le trafic dans cette chaîne :

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

4. Créer la chaîne **udp_packets** dans **example_table**

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

5. Ajoutez une règle à **udp_packets** qui compte le trafic dans cette chaîne :

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

6. Créez une chaîne pour le trafic entrant. Par exemple, pour créer une chaîne nommée **incoming_traffic** dans **example_table** qui filtre le trafic entrant :

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

7. Ajouter une règle avec une carte anonyme à **incoming_traffic**:

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

La carte anonyme distingue les paquets et les envoie aux différentes chaînes de comptage en fonction de leur protocole.

8. Pour dresser la liste des compteurs de trafic, affichez **example_table**:

```
# 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 }
  }
}
```

Les compteurs des chaînes **tcp_packets** et **udp_packets** affichent le nombre de paquets et d'octets reçus.

2.6.2. Utilisation de cartes nommées dans nftables

Le cadre **nftables** prend en charge les cartes nommées. Vous pouvez utiliser ces cartes dans plusieurs règles au sein d'une table. Un autre avantage par rapport aux cartes anonymes est que vous pouvez mettre à jour une carte nommée sans remplacer les règles qui l'utilisent.

Lorsque vous créez une carte nommée, vous devez spécifier le type d'éléments :

- **ipv4_addr** pour une carte dont la partie de correspondance contient une adresse IPv4, telle que **192.0.2.1**.
- **ipv6_addr** pour une carte dont la partie de correspondance contient une adresse IPv6, telle que **2001:db8:1::1**.
- **ether_addr** pour une carte dont la partie de correspondance contient une adresse de contrôle d'accès au support (MAC), telle que **52:54:00:6b:66:42**.
- **inet_proto** pour une carte dont la partie de correspondance contient un type de protocole Internet, tel que **tcp**.
- **inet_service** pour une carte dont la partie correspondante contient un numéro de port de nom de service Internet, tel que **ssh** ou **22**.
- **mark** pour une carte dont la partie "match" contient une marque de paquet. Une marque de paquet peut être une valeur entière positive de 32 bits (**0** à **2147483647**).
- **counter** pour une carte dont la partie "correspondance" contient une valeur de compteur. La valeur du compteur peut être un nombre entier positif de 64 bits.
- **quota** pour une carte dont la partie "correspondance" contient une valeur de quota. La valeur du quota peut être un nombre entier positif de 64 bits.

Par exemple, vous pouvez autoriser ou rejeter des paquets entrants en fonction de leur adresse IP source. En utilisant une carte nommée, vous n'avez besoin que d'une seule règle pour configurer ce scénario, les adresses IP et les actions étant stockées dynamiquement dans la carte.

Procédure

1. Créez une table. Par exemple, pour créer une table nommée **example_table** qui traite les paquets IPv4 :

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

2. Créer une chaîne. Par exemple, pour créer une chaîne nommée **example_chain** dans **example_table**:

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

IMPORTANT

Pour éviter que l'interpréteur de commandes n'interprète les points-virgules comme la fin de la commande, vous devez les faire précéder d'une barre oblique inverse.

3. Créez une carte vide. Par exemple, pour créer une carte pour les adresses IPv4 :

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

4. Créez des règles qui utilisent la carte. Par exemple, la commande suivante ajoute une règle à **example_chain** dans **example_table** qui applique des actions aux adresses IPv4 qui sont toutes deux définies dans **example_map**:

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

5. Ajouter les adresses IPv4 et les actions correspondantes à **example_map**:

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

Cet exemple définit les correspondances entre les adresses IPv4 et les actions. En combinaison avec la règle créée ci-dessus, le pare-feu accepte les paquets provenant de **192.0.2.1** et laisse tomber les paquets provenant de **192.0.2.2**.

6. Facultatif : Améliorez la carte en ajoutant une autre adresse IP et une déclaration d'action :

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

7. Facultatif : Supprimer une entrée de la carte :

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

8. Facultatif : Affichez l'ensemble de règles :

```
# 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;
```

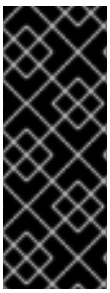
```
    ip saddr vmap @example_map
  }
}
```

2.6.3. Ressources supplémentaires

- La section **Maps** dans la page de manuel **nft(8)**

2.7. EXEMPLE : PROTECTION D'UN RÉSEAU LOCAL ET D'UNE ZONE DÉMILITARISÉE À L'AIDE D'UN SCRIPT NFTABLES

Utilisez le framework **nftables** sur un routeur RHEL pour écrire et installer un script de pare-feu qui protège les clients d'un réseau local interne et un serveur web dans une zone démilitarisée (DMZ) contre les accès non autorisés provenant d'Internet et d'autres réseaux.



IMPORTANT

Cet exemple n'est donné qu'à titre de démonstration et décrit un scénario avec des exigences spécifiques.

Les scripts de pare-feu dépendent fortement de l'infrastructure du réseau et des exigences en matière de sécurité. Utilisez cet exemple pour apprendre les concepts des pare-feu **nftables** lorsque vous écrivez des scripts pour votre propre environnement.

2.7.1. Conditions du réseau

Le réseau de cet exemple présente les conditions suivantes :

- Le routeur est connecté aux réseaux suivants :
  - L'Internet par l'interface **enp1s0**
  - L'interface interne du réseau local (LAN) **enp7s0**
  - La DMZ à travers **enp8s0**
- L'interface Internet du routeur dispose d'une adresse IPv4 statique (**203.0.113.1**) et d'une adresse IPv6 (**2001:db8:a::1**).
- Les clients du réseau local interne utilisent uniquement des adresses IPv4 privées de la plage **10.0.0.0/24**. Par conséquent, le trafic entre le réseau local et l'internet nécessite une traduction de l'adresse du réseau source (SNAT).
- Les PC administrateurs du réseau local interne utilisent les adresses IP **10.0.0.100** et **10.0.0.200**.
- La zone démilitarisée utilise les adresses IP publiques des plages **198.51.100.0/24** et **2001:db8:b::/56**.
- Le serveur web de la zone démilitarisée utilise les adresses IP **198.51.100.5** et **2001:db8:b::5**.
- Le routeur fait office de serveur DNS de cache pour les hôtes du réseau local et de la zone démilitarisée.

2.7.2. Exigences de sécurité pour le script du pare-feu

Voici les conditions requises pour le pare-feu **nftables** dans l'exemple de réseau :

- Le routeur doit pouvoir
  - Résoudre de manière récursive les requêtes DNS.
  - Effectuer toutes les connexions sur l'interface de bouclage.
- Les clients du réseau local interne doivent pouvoir :
  - Interroger le serveur DNS de mise en cache installé sur le routeur.
  - Accédez au serveur HTTPS dans la zone démilitarisée.
  - Accéder à n'importe quel serveur HTTPS sur Internet.
- Les PC des administrateurs doivent pouvoir accéder au routeur et à tous les serveurs de la zone démilitarisée à l'aide de SSH.
- Le serveur web dans la zone démilitarisée doit être capable de :
  - Interroger le serveur DNS de mise en cache installé sur le routeur.
  - Accéder à des serveurs HTTPS sur Internet pour télécharger des mises à jour.
- Les hôtes sur l'Internet doivent être en mesure de.. :
  - Accédez aux serveurs HTTPS dans la zone démilitarisée.
- En outre, les exigences de sécurité suivantes s'appliquent :
  - Les tentatives de connexion qui ne sont pas explicitement autorisées doivent être abandonnées.
  - Les paquets abandonnés doivent être enregistrés.

2.7.3. Configuration de la journalisation des paquets abandonnés dans un fichier

Par défaut, **systemd** enregistre les messages du noyau, tels que les paquets abandonnés, dans le journal. En outre, vous pouvez configurer le service **rsyslog** pour qu'il consigne ces entrées dans un fichier distinct. Pour éviter que le fichier journal ne grossisse à l'infini, configurez une politique de rotation.

Conditions préalables

- Le paquet **rsyslog** est installé.
- Le service **rsyslog** est en cours d'exécution.

Procédure

1. Créez le fichier **/etc/rsyslog.d/nftables.conf** avec le contenu suivant :

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

Avec cette configuration, le service **rsyslog** enregistre les paquets abandonnés dans le fichier **/var/log/nftables.log** au lieu de **/var/log/messages**.

2. Redémarrez le service **rsyslog**:

```
# systemctl restart rsyslog
```

3. Créez le fichier **/etc/logrotate.d/nftables** avec le contenu suivant pour faire pivoter **/var/log/nftables.log** si la taille dépasse 10 Mo :

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

Le paramètre **maxage 30** définit que **logrotate** supprime les journaux en rotation âgés de plus de 30 jours lors de la prochaine opération de rotation.

Ressources supplémentaires

- **rsyslog.conf(5)** page de manuel
- **logrotate(8)** page de manuel

2.7.4. Écriture et activation du script nftables

Cet exemple est un script de pare-feu **nftables** qui s'exécute sur un routeur RHEL et protège les clients d'un réseau local interne et un serveur web dans une zone démilitarisée. Pour plus de détails sur le réseau et les exigences relatives au pare-feu utilisé dans l'exemple, voir Conditions du réseau et Exigences de sécurité pour le script de pare-feu .

AVERTISSEMENT

Ce script de pare-feu **nftables** est uniquement destiné à des fins de démonstration. Ne l'utilisez pas sans l'adapter à votre environnement et à vos exigences de sécurité.

Conditions préalables

- Le réseau est configuré comme décrit dans la section Conditions du réseau.

Procédure

1. Créez le script **/etc/nftables/firewall.nft** avec le contenu suivant :

```
# 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. Inclure le script **/etc/nftables/firewall.nft** dans le fichier **/etc/sysconfig/nftables.conf**:

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

3. Activer le transfert IPv4 :

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

4. Activez et démarrez le service **nftables**:

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

Vérification

1. Facultatif : Vérifiez l'ensemble de règles **nftables**:

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

2. Essayez d'effectuer un accès que le pare-feu empêche. Par exemple, essayez d'accéder au routeur en utilisant SSH depuis la zone démilitarisée :

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

3. En fonction de vos paramètres de journalisation, vous pouvez effectuer une recherche :

   - Le journal **systemd** pour les paquets bloqués :

```
# 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 ...
```

   - Le fichier **/var/log/nftables.log** pour les paquets bloqués :

```
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. CONFIGURATION DE LA REDIRECTION DE PORT À L'AIDE DE NFTABLES

La redirection de port permet aux administrateurs de rediriger les paquets envoyés à un port de destination spécifique vers un autre port local ou distant.

Par exemple, si votre serveur web n'a pas d'adresse IP publique, vous pouvez définir une règle de transfert de port sur votre pare-feu qui transfère les paquets entrants sur les ports **80** et **443** du pare-feu vers le serveur web. Grâce à cette règle de pare-feu, les utilisateurs sur internet peuvent accéder au serveur web en utilisant l'IP ou le nom d'hôte du pare-feu.

2.8.1. Transférer les paquets entrants vers un autre port local

Vous pouvez utiliser **nftables** pour transférer des paquets. Par exemple, vous pouvez transférer les paquets IPv4 entrants sur le port **8022** vers le port **22** sur le système local.

Procédure

1. Créez une table nommée **nat** avec la famille d'adresses **ip**:

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

2. Ajoutez les chaînes **prerouting** et **postrouting** au tableau :

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

NOTE

Passez l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **prerouting** qui redirige les paquets entrants sur le port **8022** vers le port local **22**:

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

2.8.2. Transférer les paquets entrants sur un port local spécifique vers un autre hôte

Vous pouvez utiliser une règle de traduction d'adresse réseau de destination (DNAT) pour transférer les paquets entrants sur un port local vers un hôte distant. Cela permet aux utilisateurs sur Internet d'accéder à un service qui s'exécute sur un hôte avec une adresse IP privée.

Par exemple, vous pouvez transférer les paquets IPv4 entrants sur le port local **443** vers le même numéro de port sur le système distant avec l'adresse IP **192.0.2.1**.

Conditions préalables

- Vous êtes connecté en tant qu'utilisateur **root** sur le système qui doit transférer les paquets.

Procédure

1. Créez une table nommée **nat** avec la famille d'adresses **ip**:

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

2. Ajoutez les chaînes **prerouting** et **postrouting** au tableau :

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

NOTE

Passez l'option **--** à la commande **nft** pour éviter que le shell n'interprète la valeur négative de la priorité comme une option de la commande **nft**.

3. Ajoutez une règle à la chaîne **prerouting** qui redirige les paquets entrants sur le port **443** vers le même port sur **192.0.2.1**:

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

4. Ajoutez une règle à la chaîne **postrouting** pour masquer le trafic sortant :

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

5. Activer le transfert de paquets :

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

2.9. UTILISATION DE NFTABLES POUR LIMITER LE NOMBRE DE CONNEXIONS

Vous pouvez utiliser **nftables** pour limiter le nombre de connexions ou pour bloquer les adresses IP qui tentent d'établir un nombre donné de connexions afin d'éviter qu'elles n'utilisent trop de ressources système.

2.9.1. Limiter le nombre de connexions à l'aide de nftables

Le paramètre **ct count** de l'utilitaire **nft** permet aux administrateurs de limiter le nombre de connexions.

Conditions préalables

- La base **example_chain** de **example_table** existe.

Procédure

1. Créer un ensemble dynamique d'adresses IPv4 :

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

2. Ajoutez une règle qui n'autorise que deux connexions simultanées au port SSH (22) à partir d'une adresse IPv4 et qui rejette toutes les autres connexions à partir de la même IP :

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

3. Facultatif : Affichez le jeu créé à l'étape précédente :

```
# 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  }
  }
}
```

L'entrée **elements** affiche les adresses qui correspondent actuellement à la règle. Dans cet exemple, **elements** répertorie les adresses IP qui ont des connexions actives au port SSH. Notez que la sortie n'affiche pas le nombre de connexions actives ou si les connexions ont été rejetées.

2.9.2. Blocage des adresses IP qui tentent d'établir plus de dix nouvelles connexions TCP entrantes en l'espace d'une minute

Vous pouvez bloquer temporairement les hôtes qui établissent plus de dix connexions IPv4 TCP en une minute.

Procédure

1. Créez la table **filter** avec la famille d'adresses **ip**:

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

2. Ajoutez la chaîne **input** au tableau **filter**:

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

3. Ajoutez un ensemble nommé **denylist** à la table **filter**:

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

Cette commande crée un ensemble dynamique d'adresses IPv4. Le paramètre **timeout 5m** définit que **nftables** supprime automatiquement les entrées après cinq minutes afin d'éviter que l'ensemble ne se remplisse d'entrées périmées.

4. Ajoutez une règle qui ajoute automatiquement à l'ensemble **denylist** l'adresse IP source des hôtes qui tentent d'établir plus de dix nouvelles connexions TCP en l'espace d'une minute :

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

Ressources supplémentaires

- Utilisation d'ensembles nommés dans nftables

2.10. DÉBOGAGE DES RÈGLES NFTABLES

Le cadre **nftables** offre différentes options aux administrateurs pour déboguer les règles et déterminer si les paquets y correspondent.

2.10.1. Création d'une règle avec un compteur

Pour savoir si une règle est respectée, vous pouvez utiliser un compteur.

- Pour plus d'informations sur la procédure d'ajout d'un compteur à une règle existante, voir Ajout d'un compteur à une règle existante.

Conditions préalables

- La chaîne à laquelle vous souhaitez ajouter la règle existe.

Procédure

1. Ajoutez une nouvelle règle avec le paramètre **counter** à la chaîne. L'exemple suivant ajoute une règle avec un compteur qui autorise le trafic TCP sur le port 22 et compte les paquets et le trafic qui correspondent à cette règle :

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

2. Pour afficher les valeurs du compteur :

```
# 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. Ajouter un compteur à une règle existante

Pour savoir si une règle est respectée, vous pouvez utiliser un compteur.

- Pour plus d'informations sur une procédure qui ajoute une nouvelle règle avec un compteur, voir Création d'une règle avec le compteur .

Conditions préalables

- La règle à laquelle vous souhaitez ajouter le compteur existe.

Procédure

1. Affiche les règles de la chaîne, y compris leurs poignées :

```
# 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. Ajoutez le compteur en remplaçant la règle par le paramètre **counter**. L'exemple suivant remplace la règle affichée à l'étape précédente et ajoute un compteur :

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

3. Pour afficher les valeurs du compteur :

```
# 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. Surveillance des paquets correspondant à une règle existante

La fonction de traçage de **nftables**, combinée à la commande **nft monitor**, permet aux administrateurs d'afficher les paquets qui correspondent à une règle. Vous pouvez activer le traçage pour une règle et l'utiliser pour surveiller les paquets qui correspondent à cette règle.

Conditions préalables

- La règle à laquelle vous souhaitez ajouter le compteur existe.

Procédure

1. Affiche les règles de la chaîne, y compris leurs poignées :

```
# 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. Ajoutez la fonction de traçage en remplaçant la règle par les paramètres **meta nftrace set 1**. L'exemple suivant remplace la règle affichée à l'étape précédente et active le traçage :

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

3. Utilisez la commande **nft monitor** pour afficher le suivi. L'exemple suivant filtre la sortie de la commande pour n'afficher que les entrées contenant **inet example_table example_chain**:

```
# 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)
...
```

AVERTISSEMENT

En fonction du nombre de règles dont le traçage est activé et de la quantité de trafic correspondant, la commande **nft monitor** peut afficher un grand nombre de résultats. Utilisez **grep** ou d'autres utilitaires pour filtrer la sortie.

2.11. SAUVEGARDE ET RESTAURATION DU JEU DE RÈGLES NFTABLES

Vous pouvez sauvegarder les règles **nftables** dans un fichier et les restaurer ultérieurement. Les administrateurs peuvent également utiliser un fichier contenant les règles pour, par exemple, transférer les règles vers un autre serveur.

2.11.1. Sauvegarde du jeu de règles nftables dans un fichier

Vous pouvez utiliser l'utilitaire **nft** pour sauvegarder le jeu de règles **nftables** dans un fichier.

Procédure

- Pour sauvegarder les règles **nftables**:

- Dans un format produit par **nft list ruleset**:

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

- Au format JSON :

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

2.11.2. Restauration du jeu de règles nftables à partir d'un fichier

Vous pouvez restaurer le jeu de règles **nftables** à partir d'un fichier.

Procédure

- Pour restaurer les règles **nftables**:
  - Si le fichier à restaurer est au format produit par **nft list ruleset** ou contient directement des commandes **nft**:

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

  - Si le fichier à restaurer est au format JSON :

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

2.12. RESSOURCES SUPPLÉMENTAIRES

- Utilisation de nftables dans Red Hat Enterprise Linux 8
- Qu'est-ce qui vient après iptables ? Son successeur, bien sûr : nftables
- Firewalld : L'avenir, c'est nftables

CHAPITRE 3. UTILISATION DE XDP-FILTER POUR UN FILTRAGE PERFORMANT DU TRAFIC AFIN DE PRÉVENIR LES ATTAQUES DDOS

Par rapport aux filtres de paquets, tels que **nftables**, Express Data Path (XDP) traite et abandonne les paquets réseau directement au niveau de l'interface réseau. Par conséquent, XDP détermine l'étape suivante pour le paquet avant qu'il n'atteigne un pare-feu ou d'autres applications. Par conséquent, les filtres XDP nécessitent moins de ressources et peuvent traiter les paquets réseau à un taux beaucoup plus élevé que les filtres de paquets conventionnels pour se défendre contre les attaques par déni de service distribué (DDoS). Par exemple, lors des tests, Red Hat a éliminé 26 millions de paquets réseau par seconde sur un seul cœur, ce qui est nettement plus élevé que le taux d'élimination de **nftables** sur le même matériel.

L'utilitaire **xdp-filter** permet d'autoriser ou de bloquer les paquets réseau entrants à l'aide de XDP. Vous pouvez créer des règles pour filtrer le trafic en provenance ou à destination d'éléments spécifiques :

- IP addresses
- Adresses MAC
- Ports

Notez que, même si **xdp-filter** a un taux de traitement des paquets nettement plus élevé, il n'a pas les mêmes capacités que, par exemple, **nftables**. Considérez **xdp-filter** comme un utilitaire conceptuel pour démontrer le filtrage de paquets à l'aide de XDP. En outre, vous pouvez utiliser le code de l'utilitaire pour mieux comprendre comment écrire vos propres applications XDP.

IMPORTANT

Sur les architectures autres que AMD et Intel 64-bit, l'utilitaire **xdp-filter** est fourni en tant qu'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat, peuvent ne pas être complètes sur le plan fonctionnel et Red Hat ne recommande pas de les utiliser pour la production. Ces aperçus offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Consultez la section Portée de l'assistance pour les fonctionnalités de l'aperçu technologique sur le portail client de Red Hat pour obtenir des informations sur la portée de l'assistance pour les fonctionnalités de l'aperçu technologique.

3.1. EXCLUSION DES PAQUETS RÉSEAU QUI CORRESPONDENT À UNE RÈGLE XDP-FILTER

Vous pouvez utiliser **xdp-filter** pour laisser tomber les paquets du réseau :

- Vers un port de destination spécifique
- À partir d'une adresse IP spécifique
- À partir d'une adresse MAC spécifique

La politique **allow** de **xdp-filter** définit que tout le trafic est autorisé et que le filtre ne laisse tomber que les paquets réseau qui correspondent à une règle particulière. Par exemple, utilisez cette méthode si vous connaissez les adresses IP source des paquets que vous voulez laisser tomber.

Conditions préalables

- Le paquet **xdp-tools** est installé.
- Pilote de réseau qui prend en charge les programmes XDP.

Procédure

1. Load **xdp-filter** pour traiter les paquets entrants sur une certaine interface, telle que **enp1s0**:

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

   Par défaut, **xdp-filter** utilise la politique **allow**, et l'utilitaire ne bloque que le trafic qui correspond à une règle.

   Vous pouvez également utiliser l'option **-f *feature*** pour n'activer que certaines fonctionnalités, telles que **tcp**, **ipv4**, ou **ethernet**. Le fait de ne charger que les fonctionnalités requises au lieu de toutes augmente la vitesse de traitement des paquets. Pour activer plusieurs fonctionnalités, séparez-les par une virgule.

   Si la commande échoue avec une erreur, cela signifie que le pilote de réseau ne prend pas en charge les programmes XDP.

2. Ajoutez des règles pour rejeter les paquets qui correspondent à ces règles. Par exemple :

   - Pour supprimer les paquets entrants sur le port **22**, entrez :

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

     Cette commande ajoute une règle qui correspond au trafic TCP et UDP. Pour ne faire correspondre qu'un protocole particulier, utilisez l'option **-p *protocol*** pour ne correspondre qu'à un protocole particulier.

   - Pour abandonner les paquets entrants provenant de **192.0.2.1**, entrez :

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

     Notez que **xdp-filter** ne prend pas en charge les plages d'adresses IP.

   - Pour abandonner les paquets entrants provenant de l'adresse MAC **00:53:00:AA:07:BE**, entrez :

```
# xdp-filter ether 00:53:00:AA:07:BE -m src
```

Vérification

- La commande suivante permet d'afficher des statistiques sur les paquets abandonnés et autorisés :

```
# xdp-filter status
```

Ressources supplémentaires

- **xdp-filter(8)** page de manuel
- Si vous êtes un développeur et que vous êtes intéressé par le code de **xdp-filter**, téléchargez et installez le RPM source correspondant (SRPM) à partir du portail client de Red Hat.

3.2. SUPPRESSION DE TOUS LES PAQUETS RÉSEAU À L'EXCEPTION DE CEUX QUI CORRESPONDENT À UNE RÈGLE XDP-FILTER

Vous pouvez utiliser **xdp-filter** pour n'autoriser que les paquets réseau :

- Depuis et vers un port de destination spécifique
- Depuis et vers une adresse IP spécifique
- De et vers une adresse MAC spécifique

Pour ce faire, utilisez la politique **deny** de **xdp-filter** qui définit que le filtre laisse tomber tous les paquets du réseau à l'exception de ceux qui correspondent à une règle particulière. Par exemple, utilisez cette méthode si vous ne connaissez pas les adresses IP source des paquets que vous voulez laisser tomber.

AVERTISSEMENT

Si vous définissez la stratégie par défaut sur **deny** lorsque vous chargez **xdp-filter** sur une interface, le noyau bloque immédiatement tous les paquets provenant de cette interface jusqu'à ce que vous créiez des règles autorisant un certain trafic. Pour éviter d'être exclu du système, saisissez les commandes localement ou connectez-vous à l'hôte via une autre interface réseau.

Conditions préalables

- Le paquet **xdp-tools** est installé.
- Vous êtes connecté à l'hôte soit localement, soit à l'aide d'une interface réseau pour laquelle vous ne prévoyez pas de filtrer le trafic.
- Pilote de réseau qui prend en charge les programmes XDP.

Procédure

1. Load **xdp-filter** pour traiter les paquets sur une certaine interface, telle que **enp1s0**:

```
# xdp-filter load enp1s0 -p deny
```

   Vous pouvez également utiliser l'option **-f *feature*** pour n'activer que certaines fonctionnalités, telles que **tcp**, **ipv4**, ou **ethernet**. Le fait de ne charger que les fonctionnalités requises au lieu de toutes augmente la vitesse de traitement des paquets. Pour activer plusieurs fonctionnalités, séparez-les par une virgule.

Si la commande échoue avec une erreur, cela signifie que le pilote de réseau ne prend pas en charge les programmes XDP.

2. Ajoutez des règles pour autoriser les paquets qui correspondent à ces règles. Par exemple :

   - Pour autoriser les paquets vers le port **22**, entrez :

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

     Cette commande ajoute une règle qui correspond au trafic TCP et UDP. Pour ne faire correspondre qu'un protocole particulier, ajoutez l'option **-p *protocol*** à la commande.

   - Pour autoriser les paquets à **192.0.2.1**, entrez :

     ```
     # xdp-filter ip 192.0.2.1
     ```

     Notez que **xdp-filter** ne prend pas en charge les plages d'adresses IP.

   - Pour autoriser les paquets à l'adresse MAC **00:53:00:AA:07:BE**, entrez :

     ```
     # xdp-filter ether 00:53:00:AA:07:BE
     ```

IMPORTANT

L'utilitaire **xdp-filter** ne prend pas en charge l'inspection des paquets avec état. Cela implique que vous ne définissiez pas de mode à l'aide de l'option **-m *mode*** ou que vous ajoutiez des règles explicites pour autoriser le trafic entrant que la machine reçoit en réponse au trafic sortant.

Vérification

- La commande suivante permet d'afficher des statistiques sur les paquets abandonnés et autorisés :

  ```
  # xdp-filter status
  ```

Ressources supplémentaires

- **xdp-filter(8)** page de manuel.
- Si vous êtes un développeur et que vous êtes intéressé par le code de **xdp-filter**, téléchargez et installez le RPM source correspondant (SRPM) depuis le portail client de Red Hat.