|  |  |
| --- | --- |
| No alt text provided for this image | **Pare-feu local avec nftables** |

# Introduction au filtrage de paquets

## Présentation

Chaque réseau local connecté à d'autres réseaux externes doit être soigneusement surveillé et inspecté à l'aide d'un filtrage de paquets ou d'un nœud de pare-feu fiable qui peut être une application ou un matériel dédié au périmètre du réseau.

À l'aide d'un pare-feu, chaque paquet entrant et sortant traversant un support réseau peut être sous contrôle et surveillance stricts en termes …

* de ports ;
* de protocoles ;
* de source   
  et
* de destination

destinés à être envoyés ou reçus par un autre nœud d'un réseau.

En cas de divergence dans les paquets réseau en raison d'attaques malveillantes, ce nœud de filtrage de paquets est capable de décider d'autoriser ou d'abandonner les paquets afin de les empêcher de traverser davantage un réseau, d'où le nom de pare-feu en fonction de certaines règles prédéfinies.

Concernant l'emplacement d'un pare-feu, il est largement reconnu qu'il doit être placé en première ligne de la couche de défense d'un réseau pour servir efficacement l'objectif susmentionné.

# Netfilter

Netfilter est un cadriciel (*framwork*) de filtrage de paquets à l'intérieur du noyau Linux 2.4.x et ses versions ultérieures.

Afin d’effectuer son travail, Netfilter fournit une application nommée Iptables, une application de filtrage de paquets commune associée à Netfilter dans le noyau Linux.   
Iptables est un système basé sur des tables permettant de définir des règles de pare-feu pour filtrer et surveiller les paquets entrants et sortants.

Techniquement, Netfilter fournit certaines opérations réseau ou un ensemble de points d’ancrage à l'intérieur du noyau Linux qui permettent à iptables de contrôler et de surveiller tous les paquets réseau traversant la pile réseau sous Linux.

Chaque programme qui fonctionne avec ces points d’ancrage sera activé pour interagir avec le flux de trafic lorsqu'un paquet déclenche un point d’ancrage spécifique.

nftables est un sous-système du noyau Linux assurant le filtrage et la classification des paquets réseau /datagrammes/trames. Il est disponible depuis le noyau Linux 3.13 publié le 19 janvier 2014.

Peu à peu, cependant, en raison des limitations d'iptables, de sa complexité élevée et de sa moindre évolutivité, un nouveau cadriciel a été introduit, nommé nftables, avec un nouvel outil de ligne de commande appelé nft, basé sur l'infrastructure Netfilter, y compris l'infrastructure hook.

# nftables

nftables est un cadriciel (*framework*) de classification de paquets open source et gratuit, publié en 2014 pour Linux.   
Il fournit le filtrage de paquets et la traduction d'adresses réseau (NAT).

nftables, fondamentalement, remplace et succède à iptables qui est également un programme de filtrage de paquets. nftables permet de définir des règles de filtrage et de journalisation de l'activité du trafic réseau.

nftables gagne en popularité dans la mesure où il présente certains avantages par rapport à iptables, notamment une évolutivité et de meilleures performances.

nftables fonctionne avec différentes familles d'adresses d'IPv4, IPv6 à ARP et netdev.

De plus, la syntaxe nftables est plus simple à utiliser, plus cohérente et compacte sans extensions spécifiques au protocole. Cela peut être visualisé en comparant les outils de commande nft et iptables pour ajouter une règle …  
**>> sudo nft add rule <Famille> <Nom de la table> <Nom de la chaine> ip daddr 10.0.0.1 accept  
>> sudo iptables -A <Nom de la chaine> -daddr 10.0.0.1 -j ACCEPT**

## Exécution de nftables

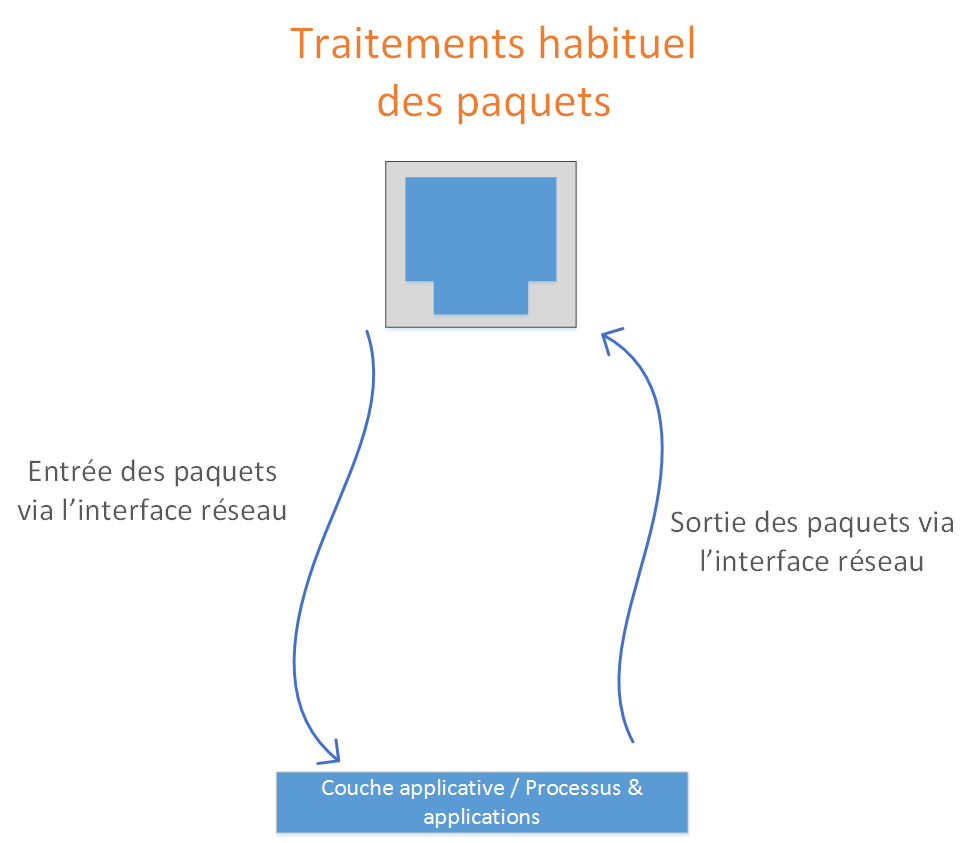
Les logiciels suivants sont nécessaires pour exécuter l'outil de ligne de commande nft …

* **Noyau Linux depuis la 3.13**, bien que des versions plus récentes du noyau soient recommandées ;
* **libmnl**  
  Bibliothèque Netlink minimaliste ;
* **libnftnl**  
  Bibliothèque d'espace utilisateur netlink de bas niveau ;
* **nft**  
  Outil de ligne de commande

## Points d’ancrage nftables

Un point d’ancrage est un point précis dans les flux réseau qui arrivent et sortent du système. Les points d’ancrages sont les éléments sur lesquels s'appliquent les règles et qui permettent donc de modifier le comportement des paquets en entrée ou sortie de l'interface en fonction des règles configurées.

Lorsqu'un paquet arrive ou sort du système, on peut imaginer qu'il n'effectue que le chemin suivant entre l'interface réseau et les applications.



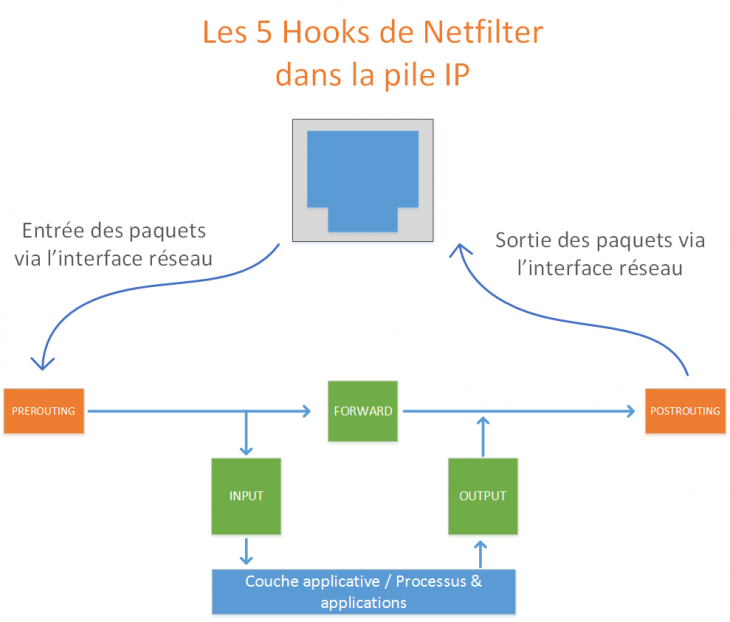
Chaque paquet qui entre dans un nœud, qu'il soit entrant ou sortant, déclenchera des points d’ancrage lorsqu'il traversera la pile réseau du noyau Linux.   
Le noyau Linux permet aux règles associées à ces points d’ancrage d'interagir avec le trafic réseau.

nftables, comme son parent, Netfilter, dispose de cinq points d’ancrage comprenant …

* le **pré-routage** (*prerouting*) ;
* l'**entrée** (*input*) ;
* la **sortie** (*output*) ;
* le **post-routage** (*post-routing*) et
* le **transfert** (*forward*).

Lorsque le flux de trafic entre dans un hôte local, il fait d'abord face au point d’ancrage de pré-routage (*prerouting*), puis au point d’ancrage d'entrée (*input*).   
Ensuite, le trafic généré par les processus de l’hôte local suit le point d’ancrage de sortie (*output*), puis le point d’ancrage de post‑routage (*postrouting*).

Les paquets destinés à son réseau mais qui ne sont pas adressés au nœud local feront face au point d’ancrage de transfert (*forward*) après avoir suivi le chemin de pré-routage puis de post-routage.   
Le point d’ancrage d'entrée (*input*), cependant, qui est un nouveau point d’ancrage dans nftables, est un point d’ancrage qui est placé avant tous les points d’ancrage (tout de suite après le point d’ancrage de pré-routage). Il peut ainsi filtrer le trafic de la couche 2 du modèle OSI.   
Avec ce point d’ancrage, donc, des politiques de filtrage précoces peuvent être définies.

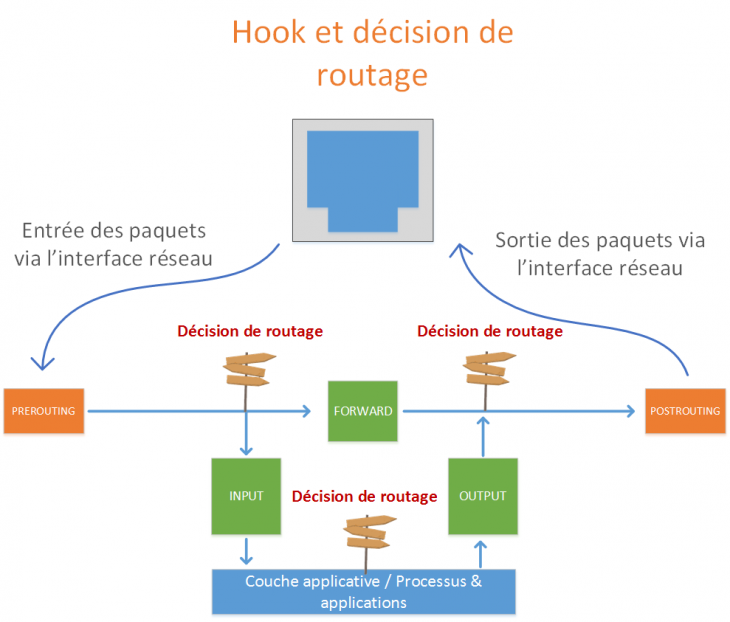


Pour chaque crochet positionné par NetFilter, on va pouvoir …

* Modifier le paquet, puis le rendre ou non au système ;
* Imposer au système de supprimer le paquet, cela correspond alors à un rejet du paquet ;
* Indiquer au système que le paquet est accepté pour ce hook, jusqu'au prochain.

Chaque point d’ancrage (*Hook*) est relié à une chaine qu’il est possible de configurer de nftables …

* Point d’ancrage NF\_IP\_PRE\_ROUTING correspondant à la chaine PREROUTING dans nftables  
  Avec ce point d’ancrage, les paquets sont analysés dans leur forme brute, sans traitement préalable du système.   
  On peut alors déterminer si on autorise le paquet à entrer plus loin dans le système ou non ;
* Point d’ancrage NF\_IP\_LOCAL\_IN correspondant à la chaine INPUT de nftables.  
  Ici, les paquets sont prêt à être envoyés à la couche applicative, c’est-à-dire aux applications qui les traiterons   
  (exemple : un serveur web ou un service FTP) ;
* Point d’ancrage NF\_IP\_FOWARD correspondant à la chaine FORWARD de nftables  
  Lorsque ce point d’ancrage est utilisé, c'est que les paquets n'iront pas vers la couche applicative, mais seront redirigés vers une autre interface réseau.   
  Par exemple dans le cas où l’hôte est un routeur ;
* Point d’ancrage NF\_IP\_LOCAL\_OUT correspondant à la chaine OUTPUT de nftables  
  Il s'agit ici du même fonctionnement que la chaine INPUT mais en sortie de la couche applicative.   
  On va donc autoriser ou non un paquet à sortir vers l'interface réseau ;
* Point d’ancrage NF\_IP\_POSTROUTING qui correspond à la chaine POSTROUTING de nftables  
  On retrouve ici le même principe que la chaine PREROUTING mais pour la sortie des paquets.   
  Les paquets analysés sont ici de nouveau dans leur forme brute.



Entre les points d’ancrage PREROUTING et INPUT, il y a une décision de routage, c’est-à-dire que le système va analyser si le paquet traité est pour lui ou non …

* Si c'est le cas, il va l'envoyer vers la couche applicative où il croise le point d’ancrage INPUT ;
* Si ce n'est pas le cas, il va l'envoyer vers la sortie de l'interface réseau où il croisera le point d’ancrage FORWARD et ne passera pas par les points d’ancrage INPUT et OUTPUT.

Les points d’ancrage sont là pour appliquer un filtre ou une action (dans le cadre du NAT par exemple) quand les paquets réseaux ne passent pas nécessairement par la couche applicative.

Lorsque l'on fait du NAT, les paquets qui traversent un routeur, mais qui ne sont pas en destination de la couche applicative sont traités par les chaines PREROUTING, FORWARD puis POSTROUTING.

Un point d’ancrage d’une chaîne fait donc référence à une étape spécifique au cours de laquelle un paquet est traité à l’aide du noyau Linux sur la base de règles définies.   
Ces points d’ancrage sont …

* l'**entrée** ;
* le **pré-routage** ;
* l'**entrée** ;
* le **transfert** ;
* la **sortie**   
  et
* le **post-routage**.

Le point d’ancrage de pré-routage, d'entrée, de transfert, de sortie et de post-routage peut prendre en charge les familles d'adresses ip, ip6 et inet.   
Afin de prendre en charge la famille d'adresses arp, les points d’ancrage d'entrée et de sortie sont utilisés tandis que pour la famille netdev, le point d’ancrage d'entrée est utilisé.

**Pré-routage** (*prerouting*)  
Tous les paquets entrant dans un nœud sont traités par ce point d’ancrage.   
Il est invoqué avant le processus de routage et est utilisé pour le filtrage précoce ou la modification des attributs de paquet qui affectent le routage ;

**Entrée** (*input*)  
Ce point d’ancrage est exécuté après la décision de routage.   
Les paquets livrés à un système local sont traités par ce point d’ancrage ;

**Transfert** (*forward*)  
Ce point d’ancrage se produit également après la décision de routage.   
Les paquets qui ne sont pas dirigés vers l’hôte local sont traités par ce point d’ancrage ;

**Sortie** (*output*)  
Ce point d’ancrage contrôle les paquets provenant des processus d'une machine locale ;

**Post-routage** (*Postrouting*)  
Ce point d’ancrage est utilisé pour les paquets quittant un système local après la décision de routage ;

**Entrée** (*Ingress***)**   
(uniquement disponible dans la famille netdev)  
Depuis le noyau Linux 4.2, le trafic peut être filtré avant la couche 3 et bien avant le pré-routage, après que les paquets ont été transmis à partir d'un pilote NIC.

# Installation

Pour installer le paquet de nftables, l’activer et le rendre actif au démarrage, on exécute les commandes suivantes …  
**>> sudo apt install nftables**

Pour vérifier le statut du service …  
**>> sudo systemctl status nftables.service  
 ● nftables.service – nftables  
 Loaded: loaded (/lib/systemd/system/nftables.service; disabled; vendor preset: enable>  
 Active: inactive (dead)  
 Docs: man:nft(8)  
 http://wiki.nftables.org**

**>> sudo systemctl start nftables.service  
>> sudo systemctl status nftables.service  
● nftables.service – nftables  
 Loaded: loaded (/lib/systemd/system/nftables.service; disabled; vendor preset: enable>  
 Active: active (exited) since Fri 2021-11-19 11:47:41 EST; 46s ago  
 Docs: man:nft(8)** [**http://wiki.nftables.org**](http://wiki.nftables.org) **Process: 594 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited, status=0/SUCC>  
 Main PID: 594 (code=exited, status=0/SUCCESS)  
 CPU: 15ms  
  
nov 19 11:47:41 debian11 systemd[1]: Starting nftables...  
nov 19 11:47:41 debian11 systemd[1]: Finished nftables.**

Commandes complémentaires … **>> sudo systemctl start nftables.service  
>> sudo systemctl enable nftables.service**

# Configuration initiale nftables

La configuration de nftables se compose de modules hiérarchiques comprenant …

* des tables ;
* des chaînes ;
* des ensembles   
  et
* des règles.

Toute la configuration de Netfiter par nftables passe par une seule commande … la commande nft.

## Vérification de la configuration actuelle

Afin de configurer nftables, il est d'abord nécessaire de vérifier le contenu actuel du fichier de configuration nftables.

Par défaut, les configurations nftables se retrouvent dans le fichier **/etc/nftables.conf** …  
**>> sudo nft list ruleset  
table inet filter {  
 chain input {  
 type filter hook input priority filter; policy accept;  
 }  
  
 chain forward {  
 type filter hook forward priority filter; policy accept;  
 }  
  
 chain output {  
 type filter hook output priority filter; policy accept;  
 }  
}**

Chaque règle possède un certain nombre de poignée (*handle*).   
Pour les afficher, on utilise le commutateur **-a** ou **--handle**.

Ces numéros sont importants car ils peuvent être utilisés pour …

* supprimer une règle spécifique ;
* ajouter une règle avant ou après une règle spécifique à l’aide un numéro de descripteur spécifique.

**>> sudo nft --handle list ruleset**ou **>> sudo nft -a list ruleset  
table inet filter { # handle 1  
 chain INPUT { # handle 1  
 type filter hook input priority filter; policy accept;  
 }  
  
 chain FORWARD { # handle 2  
 type filter hook forward priority filter; policy accept;  
 }  
  
 chain OUTPUT { # handle 3  
 type filter hook output priority filter; policy accept;  
 }  
}**

Pour afficher les tables …  
**>> sudo nft list tables**

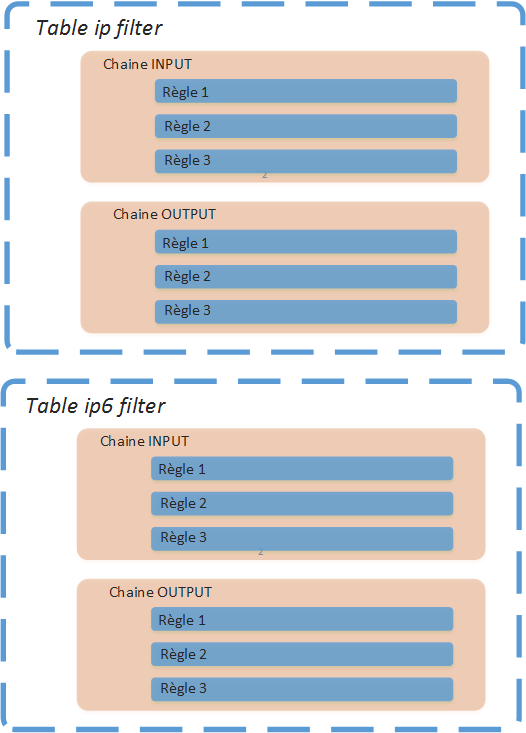
Pour lister les tables de la famille IP …  
**>> sudo nft list tables IP**

Pour afficher les tables IPv6 …  
**>> sudo nft list tables ip6**

## Structure de la table Netfilter

Les tables Netfilter sont organisées hiérarchiquement.

Les tables contiennent des chaînes et les chaînes contiennent des règles …



**Remarque** …  
Au cas où l’on aurait besoin d'effacer toutes les données actuelles du fichier de configuration et de créer ses propres modules nftables, on peut utiliser la commande ...  
**>> sudo nft flush ruleset**

# Modules nftables

## Tables

Une table est au sommet de l'ensemble de règles en tant que conteneur dans lequel les chaînes qui sont les conteneurs des règles sont conservées.   
Avec nftables, il ne s'agit que d'un simple conteneur que l'on peut qualifier d’organisationnel qui est défini par l'utilisateur.

Il n'y a pas de table prédéfinie dans nftables a contrario d'iptables dans lequel on peut trouver les tables filter, raw, NAT, mangle, …

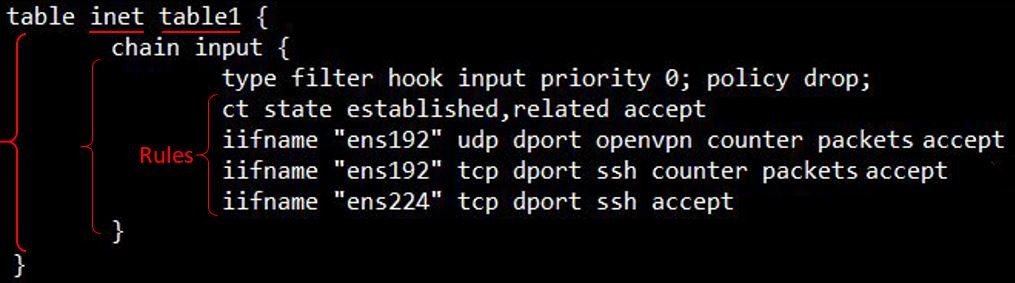
En termes d'opérations pouvant être effectuées sur une table, on peut citer …

* l'**ajout** ;
* la **suppression** ;
* l'**affichage** ou la liste   
  et
* le **nettoyage**.

Compte tenu de l'importance des tables, la présentation hiérarchique de la structure de nftables est la suivante …  
**Tables > Chaînes > Règles**

Il est possible de manipuler une table, une chaîne ou une règle à l'aide des outils de commande nft ou encore en les écrivant directement dans le fichier de configuration nftablrs.

La structure suivante est visible dans le fichier de configuration nftables.



### Familles d'adresses

Les familles d'adresses déterminent le type de paquets entrants et sortants traités par nftables.   
Pour chaque famille d'adresses, le noyau Linux contient des points d’ancrage spécifiques à différentes étapes des chemins de traitement des paquets.   
Il appelle nftables afin de prendre la décision d'autoriser ou d’abandonner un paquet uniquement si des règles pertinentes pour ces points d’ancrage, telles que l'entrée ou la sortie, sont définies.

Ces familles d'adresses sont les suivantes …

* **ip**  
  Famille d'adresses IPv4.
* **ip6** ;  
  Famille d'adresses IPv6 ;
* **inet**  
  Familles d'adresses IPv4 et IPv6 ;
* **arp**  
  Famille d'adresses ARP, gérant les paquets ARP IPv4 (couche 2 du modèle OSI) ;
* **bridge** (pont)  
  Famille d'adresses de pont, gérant les paquets traversant un périphérique de pont (couche 2 du modèle OSI) ;
* **netdev**  
  Famille d'adresses netdev, gérant les paquets du point d’ancrage (*hook*) d'entrée (couche 3 du modèle OSI).

La table de la famille netdev, en tant que moyen de point d’ancrage d'entrée, permet de filtrer plus tôt le trafic avant qu'il n'atteigne d'autres filtres en dessus de la couche 3 sur le modèle OSI.   
La famille netdev avec point d’ancrage d'entrée est une étape idéale pour supprimer les paquets résultant d'attaques DDOS, car ce point d’ancrage fonctionne très tôt dans le chemin des paquets du réseau.

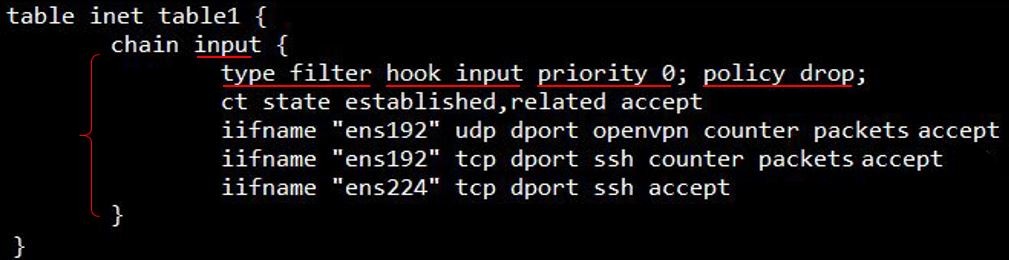
## Chaînes

Une chaîne est un conteneur de règles et se situe à l'intérieur d'une table existante.

Les chaînes peuvent être …

* une chaîne de base qui peut contrôler les paquets destinés à un nœud et a un point d’ancrage ;
* une chaîne sans base est une chaîne qui est utilisée pour l'organisation des chaînes et n'a pas de point d’ancrage   
  donc aucun contrôle sur les paquets.

Semblable à une table, toutes les activités opérationnelles peuvent être effectuées sur une chaîne en plus de renommer une chaîne.   
Les chaînes doivent être suivies d'un nom et d'une accolade ouverte et fermée.   
Elles sont également fournies avec un type, un point d’ancrage, une priorité et une stratégie qui doivent être définis lors de la création d'une chaîne.



**Types de chaînes**

Il existe trois types de chaine …

* **Filter**   
  Il s'agit d'un type de chaîne standard et prend en charge toutes les familles d'adresses,   
  à savoir arp, bridge, ip, ip6 et inet et les points d’ancrage ;
* **Route**  
  Cette chaine ne prend en charge que les familles d'adresses ip et ip6 et uniquement le point d’ancrage de sortie.   
  Si les parties pertinentes de l'en-tête IP ont été modifiées, une nouvelle recherche de route est effectuée ;
* **NAT**  
  Cette chaine peut effectuer la traduction d'adresses réseau et ne prend en charge que les familles d'adresses ip et ip6.   
  Les points d’ancrage de pré-routage, d'entrée, de sortie et de post-routage sont également pris en charge.

## Stratégies (*Policies*)

Les chaînes doivent avoir leurs politiques selon lesquelles les paquets sont traités pour être rejetés ou acceptés par défaut. Ces valeurs de stratégie peuvent être **Accepter**, qui est la stratégie par défaut, ou **Abandon**.

La politique d'acceptation signifie que tous les paquets réseau en fonction de leurs emplacements définis par le point d’ancrage doivent être acceptés par défaut.  
La politique de d’abandon signifie que par défaut, tous les paquets réseau doivent être supprimés en fonction de leurs emplacements définis par le point d’ancrage dans une chaîne.

Basé sur des règles définies à l'intérieur d'une chaîne, ils seront alors acceptés ou non.

## Règles (*Rules*)

Les règles sont les actions qui contrôlent les paquets entrants et sortants en fonction des points d’ancrage définis dans une chaîne.   
Si une règle à l'intérieur d'une chaîne correspond à un paquet basé sur l'étape dérivée de leurs points d’ancrage, le paquet est abandonné ou accepté.

Une règle est évaluée de gauche à droite de telle sorte que lorsque la première instruction correspond, elle continue avec les parties suivantes d'une règle.  
Sinon, la règle suivante sera évaluée.

La structure d'une règle comprend des correspondances et des instructions qui sont les suivantes …  
**<correspondance> <déclarations>**

Par exemple …  
**iifname <Nom de l’interface> Stratégie: <Accepter ou Abandonner>**

### Correspondance

Les correspondances sont ces filtres qui permettent à une règle de filtrer certains paquets.

Voici brièvement quelques correspondances importantes avec leurs formats possibles …

* **ip saddr** <**Adresse IP source**>
* **Ip daddr** <**Adresse IP de destination**>
* **tcp/udp sport** <**Port source**>
* **tcp/udp dport** <**Port de destination**>
* **tcp flags** <**Drapeau**>
* **icmp type** <**Type**>
* **iifname** <**Nom de l’interface d’entrée**>
* **oifname** <**Nom** **de l’interface de sortie**>
* **protocol** <**Protocole**>

### Déclarations

Une instruction est l'action définie effectuée une fois qu'un paquet correspond à une ou plusieurs correspondances définies par une règle.   
Les déclarations comprennent des déclarations de verdict, de journal et de compteur.

**Déclarations de verdict**Une déclaration de verdict modifie le flux de contrôle dans l'ensemble de règles et émet des décisions de politique pour les paquets.

Les déclarations de verdict valides sont …

* **Accept**  
  Acceptation du paquet et arrête l'évaluation des règles restantes ;
* **Drop**  
  Suppression du paquet et arrêt de l'évaluation des règles restantes ;
* **Queue** (file d'attente)  
  Mise en file d'attente du paquet dans l'espace utilisateur et arrêt de l'évaluation des règles restantes ;
* **Continue** (continuer)  
  Continuation de l'évaluation de l'ensemble de règles avec la règle suivante ;
* **jump** **<Chaine>** (Saut)  
  Continuation à la première règle de <Chaine>.   
  Cette déclaration continuera à évaluer les prochaines règles pour finalement revenir à la dernière position ou une instruction return est émise ;
* **Return** (retour)  
  Retour de la chaîne actuelle et continuation vers la règle suivante de la dernière chaîne.   
  Dans une chaîne de base, cela équivaut à accepter ;
* **Goto** **<Chaine>** (Aller à)  
  Similaire à jump, mais après avoir terminé les règles dans <Chaine>, l'évaluation continuera à évaluer les chaînes suivantes au lieu d'attendre un retour à la dernière chaîne.

# Manipulation des objets

**Rappel …**Toute la configuration de Netfiter par nftables passe par une seule commande … la ommande nft.

**Préalable** …  
On peut effacer toutes les données actuelles du fichier de configuration et de créer ses propres modules nftables.  
Pour nettoyer …  
**>> sudo nft flush ruleset**

## Module d’aide

Pour afficher le module d’aide de nftables …  
**>> sudo nft --help**

**Remarque** …  
nftables est sensible à la casse … **INPUT** est différent d’**input**.

## Ajout d’un objet

### Ajout d’une table

Pour créer une table …  
**>> sudo nft add table <Famille> <Table>**  
**>> sudo nft add table ip filter**

### Ajout d’une chaine

Il existe deux types de chaines …

**Chaine simple**  
Une chaine simple n'est pas rattaché à un point d’ancrage.

Pour créer une chaine simple…  
**>> sudo nft add chain <Famille> <Table> <Chaine>  
>> sudo nft add chain ip filter INPUT  
>> sudo nft add chain ip filter OUTPUT**

**Chaine de base**Une chaine de base est une chaine va directement se rattacher à un point d’ancrage.  
Par exemple, lors de la création il faut préciser que la chaine doit se rattacher au point d’ancrage input.

Pour créer une chaine de base …  
**>> sudo nft add chain <Chaine> <Point d’ancrage> { type type> hook <Point d’ancrage> priority <Priorité> \; }  
>> sudo nft add chain ip filter INPUT { type filter hook input priority 0 \; }  
>> sudo nft add chain filter ip OUTPUT { type filter hook output priority 0 \; }**

On retrouve également une notion de priorité.   
La priorité permet de gérer les cas où plusieurs chaines sont positionnées sur le point d’ancrage INPUT par exemple. Chaque table peut avoir plusieurs chaines, et chaque chaine peut être positionnée sur un point d’ancrage.  
On peut avoir une chaine nommée INPUT\_1 et une autre chaine nommée INPUT\_2 pour des besoins organisationnels.   
La priorité la plus basse prend ainsi le pas sur les autres priorités.

**Remarque** …  
Il est possible d'avoir des priorités négatives (Exemple : -100) qui sont alors prioritaires sur une priorité 0.

**Vérification de la création d’une table**Pour vérifier la création de la table filter et des chaines INPUT et OUTPUT …  
**>> sudo nft --handle list ruleset  
ou  
>> sudo nft -a list ruleset**

**table ip filter { # handle 2  
 chain INPUT { # handle 1  
 }  
  
 chain OUTPUT { # handle 2  
 }**

**}**

### Ajout de règles

Pour **abandonner silencieusement les paquets** **vers une adresse externe** …  
**>> sudo nft add rule <Famille> <Table> <Chaine> <Famille> daddr <Adresse IP> <Déclaration>  
>> sudo nft add rule ip filter OUTPUT ip daddr 192.168.1.100 drop  
>> sudo nft --handle list ruleset  
table ip filter { # handle 2  
 chain INPUT { # handle 1  
 }  
  
 chain OUTPUT { # handle 2  
 ip daddr 192.168.1.100 drop # handle 3  
 }  
}**

Les compteurs de règles[[1]](#footnote-1) sont facultatifs avec nftables et le mot-clé counter doit être utilisé pour l'activer …  
**>> sudo nft rule <Famille> <Table> <Chaine> <Famille> daddr <Adresse IP> counter <Déclaration>  
>> sudo nft rule ip filter OUTPUT ip daddr 192.168.1.200 counter drop  
>> sudo nft --handle list ruleset  
  
…  
chain OUTPUT { # handle 2  
 ip daddr 192.168.1.100 drop # handle 3  
 ip daddr 162.168.1.100 counter packets 0 bytes 0 drop # handle 7  
 }  
…**

Pour **ajouter une règle pour un réseau** …  
**>> sudo nft add rule <Famille> <Table> <chaine> <Famille> daddr <Adresse du reseau/CIDR> counter <Déclaration>  
>> sudo nft add rule ip filter OUTPUT ip daddr 192.168.1.0/24 counter accept  
>> sudo nft --handle list ruleset  
…  
chain OUTPUT { # handle 2  
 ip daddr 192.168.1.100 drop # handle 3  
 ip daddr 162.168.1.100 counter packets 0 bytes 0 drop # handle 7  
 ip daddr 192.168.1.0/24 counter packets 0 bytes 0 # handle 8  
 }  
…**

Pour **abandonner silencieusement les paquets vers le port 80** …  
**>> sudo nft add rule <Famille> <Table> <Chaine> <Protocole> dport <Numéro de port> <Déclaration>  
>> sudo nft add rule ip filter INPUT tcp dport 80 drop  
>> sudo nft --handle list ruleset  
…  
chain INPUT { # handle 4  
 tcp dport 80 drop # handle 9  
 }  
…**

Pour **accepter une demande d'écho ICMP** …  
**>> sudo nft add rule <Famille> <Table> Chaine> icmp type echo-request <Déclaration>  
>> sudo nft add rule filter OUTPUT icmp type echo-request accept  
>> sudo nft --handle list ruleset  
…  
chain INPUT { # handle 1  
 tcp dport 80 drop # handle 10  
 icmp type echo-request accept # handle 11  
 }  
…**

Pour **combiner le filtrage**, il suffit de spécifier plusieurs fois la syntaxe ip …  
**>> sudo nft add rule <Famille> <Table> <chaine> ip protocol icmp ip daddr <Adresse IP> counter <Déclaration>  
>> sudo nft add rule ip filter OUTPUT ip protocol icmp ip daddr 192.168.1.100 counter drop  
>> sudo nft --handle list ruleset  
…  
chain OUTPUT { # handle 2  
 ip daddr 192.168.1.100 drop # handle 3  
 ip daddr 162.168.1.100 counter packets 0 bytes 0 drop # handle 7  
 ip daddr 192.168.1.0/24 counter packets 0 bytes 0 # handle 8  
 ip protocol icmp ip daddr 192.168.1.100 counter packets 0 bytes 0 drop # handle 13  
 }  
…**

Il est possible d'insérer ou d'ajouter une règle à une position spécifique.   
Pour ce faire, on doit obtenir la poignée (*handle*) de la règle dans laquelle on souhaite en insérer ou en ajouter une nouvelle.

Pour insérer une règle à une position précise …  
**>> sudo nft add rule <Table> <chaine> position <Numéro de poignée> <Famille>daddr <Adresse IP> <Déclaration>  
>> sudo nft add rule filter OUTPUT position 8 ip daddr 192.168.1.111 drop**

Si on souhaite uniquement faire correspondre un protocole …  
**>> sudo nft insert rule <Table> <chaine> <Famille>protocole <Protocol> counter  
>> sudo nft insert rule filter OUTPUT ip protocol tcp counter**

### Suivi de connexion

Pour accepter tous les paquets entrants d'une connexion établie …  
**>> sudo nft insert rule <Table> input ct state established <Déclaration>  
>> sudo nft insert rule filter INPUT ct state established accept**

**Remarque importante** …  
On doit toujours terminer par ajouter à la fin une règle incluant la déclaration drop afin de bloquer tous les autres types de paquets.   
La position des déclarations drop ont une importance.   
S'ils étaient positionnés en début de chaque chaine, le pare-feu ne fonctionnera pas correctement.   
Il faut bien penser à toujours autoriser, puis bloquer le reste en terminant avec une chaine ayant comme déclaration drop, cela pour chaque chaine

### Exemple

On souhaite autoriser les ports 80 et 443 en entrée.

Voici les commandes à saisir …  
**>> sudo nft add rule filter INPUT tcp dport 80 accept  
>> sudo nft add rule filter INPUT tcp dport 443 accept  
>> sudo nft add rule filter INPUT drop  
>> sudo nft add rule filter OUTPUT tcp sport 443 accept  
>> sudo nft add rule filter OUTPUT tcp sport 443 accept  
>> sudo nft add rule filter OUTPUT drop**

## Suppression d’objet

### Vidange d’une table

La vidange d'une table supprime toutes les chaînes et règles de celle-ci, mais laisse la table elle-même en place.

Pour vider une table …  
**>> sudo nft flush table <Table>  
>> sudo nft flush table filter**

### Suppression d’une table

La suppression d'un tableau supprime la table et tout son contenu.   
Pour supprimer une table …  
**>> sudo nft delete table <Table>  
>> sudo nft delete table filter**

**Remarque** …  
Si la table n'est pas vide, c’est-à-dire s'il y a des chaines et des tables de configurées à l'intérieur, nftables va refuser de la supprimer et retourner un message d’erreur.

### Vidange d’une chaine

Le vidage d'une chaîne en supprime toutes les règles mais ne supprime pas la chaîne elle-même, y compris ses propriétés.

Pour vider une chaine …  
La commande est sous la forme :nft flush chain <table name> <chain name>  
**>> sudo nft flush chain <Table> <Chaine>   
>> sudo nft flush chain filter OUTPUT**

### Suppression d’une chaine

Pour supprimer une chaine …  
**>> sudo nft delete chain <Table> <Chaine>   
>> sudo nft delete chain filter OUTPUT**

**Remarque** …  
On ne peut **supprimer qu'une chaîne vide**.  
Il faut donc vider la chaîne avant d'entrer la commande de suppression.

### Suppression de règles

Pour supprimer une règle spécifique, on doit identifier au préalable connaitre le numéro de de poignée (*handle*).

Pour supprimer une règle …  
**>> sudo nft delete rule <Table> <Chaine> handle <Numéro de poignée>  
>> sudo nft delete rule filter OUTPUT handle 7**

Il est également possible de supprimer une règle sans désigner sa poignée …  
**>> sudo nft delete rule <Table> <Chaine> <Famille><Adresse IP> counter  
>> sudo nft delete rule filter OUTPUT ip saddr 192.168.1.100 counter**

## IPv6

La manipulation des objets IPv9 suit en toute les règles d’iPv4.

Pour ajouter une règle …  
**>> sudo nft add rule <Famille> <Table> <Chaine> <Famille> daddr <Adresse IP> <Déclaration>  
>> sudo nft add rule ip6 filter OUTPUT ip daddr 192.168.1.100 drop**

L’affichage des règles peut se faire avec …  
**>> sudo nft list table <Famille> <Table>  
>> sudo nft list table ip6 filter**

Pour accepter la configuration IPv6 dynamique et la découverte de voisins …  
**>> sudo rule ip6 <Table> <Chaine> icmpv6 type nd-neighbor-solicit <Déclaration>  
>> sudo nft add rule ip6 filter INPUT icmpv6 type nd-neighbor-solicit accept**

**>> sudo rule ip6 <Table> <Chaine> icmpv6 type nd-router-solicit <Déclaration>  
>> sudo nft add rule ip6 filter INPUT icmpv6 type nd-router-advert accept**

**Mode interactif**

**Fichier de configuration /etc/nftables.conf**

## Sauvegarde des règles nftables

Pour sauvegarder les règles nftables, il suffit à rediriger ce qui devrait être affiché sur le terminal dans un fichier.

Pour sauver la configuration complète …  
**>> sudo nft list rulseset > nftables.rules**

Pour sauvegarder le contenu d’une table …  
**>> sudo nft list table <Table> > <Fichier>**  
**>> sudo nft list table filter > nftables.filter.rules**

Remarque …  
Le fichier peut avoir le suffixe que l’on souhaite.   
Pour rappel, le système d'extension n'existe pas sous Linux, ils sont totalement optionnels.

## Restauration des règles nftables

Pour restaurer les règles nftables, on utlise la commande nft avec le commutateur -f.  
Cette commande permet d'importer des règles depuis un fichier.  
Pour importer le contenu d’une table …  
**>> sudo nft list table <Table> < <Fichier>**  
**>> sudo nft -f < nftables.rules  
>> sudo nft -f < nftables.filter.rules**

## Application des règles au démarrage

On peut vouloir appliquer ses règles nftables dès le démarrage de l'interface réseau.   
Il suffit d’indiquer, à la fin du montage de l’interface réseau, l'exécution d'une commande qui va importer les règles d'un fichier nftables.

Il faut sauvegarder la configuration nftables dans le fichier /root/nftables/iptables.rules.

Par la suite, il faut éditer le fichier d l’interface réseau …  
**>> sudo nano /etc/network/interfaces  
auto eth0  
iface ens33 inet static  
 address 192.168.1.100  
 netmask 255.255.255.0  
 pre-up nft -f /root/nftables/iptables.rules**

La dernière ligne débutant par pre-up … va exécuter la commande qui suit une fois que l'interface sera montée, assurant ainsi que les règles nftables soient restaurées au démarrage.

## Fichier /etc/nftables.conf

Il est possible d’écrire un fichier de configuration afin de modifier à la demande un ensemble de ables, chaines et règles.

Pour créer ce fichier …  
**>> sudo nano /etc/nftables.conf  
#!/usr/sbin/nft -f  
  
# Remise du pare-feu à zéro  
flush ruleset  
  
# Filtre IPv4  
table ip filter {  
 chain input {  
 type filter hook input priority 0;  
 ct state established accept  
 ct state related accept  
 meta iif lo accept  
 tcp dport ssh counter packets 0 bytes 0 accept  
 counter packets 5 bytes 5 log drop  
 }  
 chain output {  
 type filter hook output priority 0;  
 ct state established accept  
 ct state related accept  
 meta oif lo accept  
 ct state new counter packets 0 bytes 0 accept  
 }  
}  
  
# Filtre IPv6  
table ip6 filter {  
 chain input {  
 type filter hook input priority 0;  
 ct state established accept  
 ct state related accept  
 meta iif lo accept  
 tcp dport ssh counter packets 0 bytes 0 accept  
 icmpv6 type { nd-neighbor-solicit, echo-request, nd-router-advert, nd-neighbor-advert } accept  
 counter packets 5 bytes 5 log drop  
 }  
 chain output {  
 type filter hook output priority 0;  
 ct state established accept  
 ct state related accept  
 meta oif lo accept  
 ct state new counter packets 0 bytes 0 accept  
 }  
}**

# Référence

Document adapté du site …  
IT-connect

Publié sous licence Common Creative

1. Un compteur permet de comptabiliser à la fois le nombre total de paquets et le nombre total d'octets que l’interface a reçus ou envoyés depuis sa dernière réinitialisation.   
   Avec nftables, on doit spécifier explicitement un compteur pour chaque règle que l’on souhaite compter. [↑](#footnote-ref-1)