GNU / Linux — Firewall avec nftables

Comprendre les familles, les hooks, les tables, les chaînes et les règles

Objectif pédagogique

Comprendre où et quand nftables intercepte les paquets.

Construire mentalement l'ordre : famille → table → chaîne → hook → règle → action.

Lire une commande nft morceau par morceau, sans apprendre une ligne entière d'un bloc.

Retenir les commandes essentielles grâce à des mémos et des exemples concrets.

Contenu du support

Le contenu du cours fourni est conservé. Les signatures de bas de page ont été retirées.

Les encadrés intitulés « Mémo pédagogique », « Exemple simple » ou « Point de vigilance » sont des ajouts destinés à faciliter la compréhension.

Plan

Fil conducteur

Une famille indique quel type de trafic est géré.
Une table regroupe des chaînes.
Une chaîne est rattachée à un hook et contient des règles.
Une règle décrit ce que l'on recherche et l'action à effectuer.

1. Avantages par rapport à iptables

- Meilleures performances
- Un seul outil pour gérer toutes les couches (familles) réseau
  - IP
  - IPv6
  - ARP
  - Ethernet Bridge
- Meilleure gestion du rechargement dynamique des règles
- Classification de paquets plus rapide
- Plusieurs actions possibles par règle
  - Par exemple bloquer un paquet et logger avec une seule règle
  - 2 règles avec iptables

Mémo pédagogique

iptables utilisait plusieurs outils séparés selon les familles ; nftables rassemble ces usages avec une commande principale : nft.
Avec nftables, une seule règle peut cumuler plusieurs actions, par exemple journaliser puis bloquer.

2. Fonctionnement

- Appliquer des règles sur certains points d'ancrage (hooks)
- En fonction du type de famille qui est visé (IP, ARP, bridge...)

- Hooks situés à différentes étapes de la gestion de la trame
  - Depuis l'entrée de la couche 3 (filtrer directement sur des paquets de type ether de la couche 2 comme IPv4, ARP, PPPoE...)
  - Jusqu'à la couche 7 (filtrage plus fin, par exemple gestion du trafic SSH)

Définition simple : un hook

Un hook est un point de passage dans le traitement réseau du noyau.

Une chaîne de base peut être accrochée à ce point pour examiner les paquets au bon moment.

3. Les hooks

Hook	Contenu du support	Image mentale
ingress	Tous les paquets entrants dans le système passent par ce hook. Une interface d'écoute doit être spécifiée.	Filtrage précoce selon l'Ethertype.
egress	Tous les paquets sortants du système passent par ce hook. Une interface d'écoute doit être spécifiée.	Relié à aucune couche en particulier.
prerouting	Les paquets transitent ici juste après ingress, sauf ARP.	Décider si le paquet est local ou doit être transféré.
input	Tous les paquets à destination du système passent par ce hook.	Trafic destiné à la machine.
forward	Hook invoqué juste après prerouting.	Paquets redirigés vers une autre interface.
postrouting	Hook invoqué juste après forward.	Dernières opérations avant la sortie.
output	Tous les paquets sortants du système passent par ce hook.	Trafic créé localement.

Mémo express

INPUT = le paquet entre dans ma machine.

OUTPUT = le paquet est créé par ma machine et sort.

FORWARD = le paquet traverse ma machine.

PREROUTING = avant la décision de routage.

POSTROUTING = après la décision de routage, juste avant la sortie.

4. Familles et hooks associés

netdev	**Rôle :** Filtrage sur des paquets de type Ethertype, IPv4 et IPv6. **Hooks :** ingress, egress
arp	**Rôle :** Gestion des paquets ARP reçus et envoyés par le système. **Hooks :** input, output
bridge	**Rôle :** Gestion des paquets Ethernet passant par des interfaces Bridge. **Hooks :** prerouting, forward, postrouting, input, output
ip / ip6 / inet	**Rôle :** Gestion des paquets IPv4, IPv6 ou des deux sans distinction avec inet. **Hooks :** prerouting, forward, postrouting, input, output

Mémo pédagogique

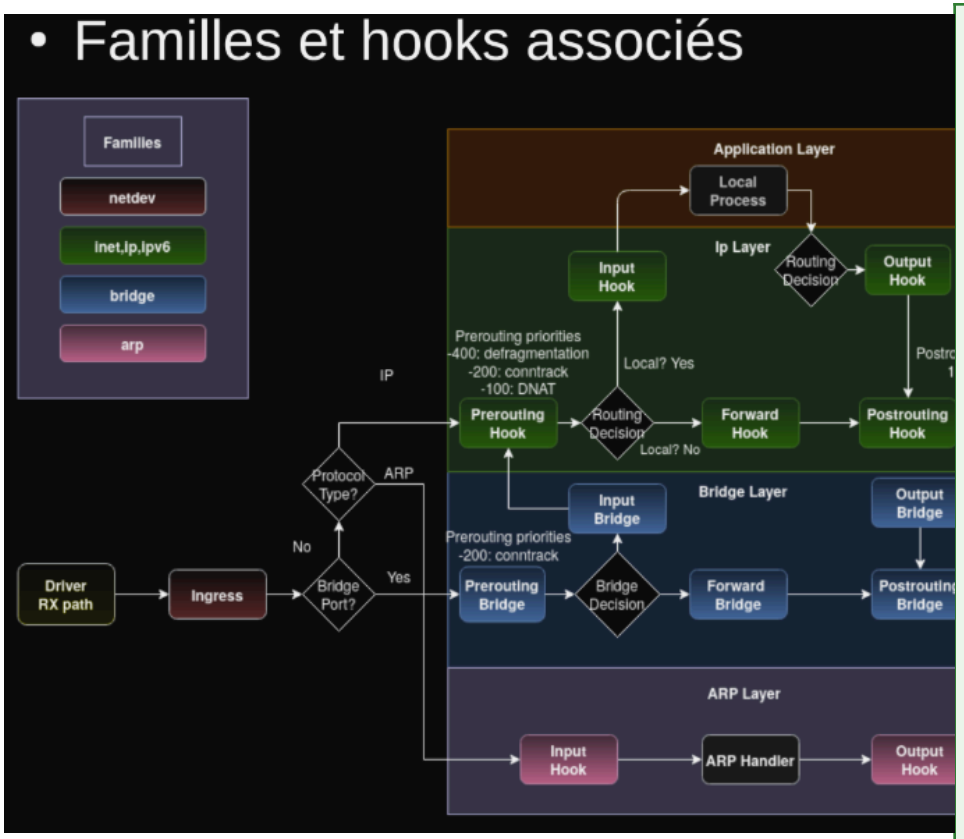
netdev = très tôt, au niveau des interfaces.

arp = uniquement les échanges ARP.

bridge = trafic Ethernet traversant un pont logiciel.

ip = IPv4 ; ip6 = IPv6 ; inet = IPv4 et IPv6 dans la même table.

Schéma : familles et hooks associés



Cheminement du trafic selon les familles netdev, inet/ip/ip6, bridge et arp.

Comment lire ce schéma

1. Le paquet arrive par le chemin RX du pilote, puis passe par ingress.
2. Selon son type et le port utilisé, il est dirigé vers la famille IP, Bridge ou ARP.
3. La famille IP utilise notamment prerouting, input, forward, output et postrouting.
4. La famille Bridge possède ses propres hooks équivalents.
5. La famille ARP utilise input et output autour du gestionnaire ARP.
6. Le paquet termine par egress puis par le chemin TX du pilote.

5. Configuration : trois parties

Élément	Définition du support	Compréhension
Tables	Conteneurs qui contiennent des chaînes.	Identifiées par leur famille et un nom défini par l'utilisateur.
Chaînes	Conteneurs rattachés à des hooks qui contiennent des règles.	Elles déterminent le point de passage traité.
Règles	Actions à appliquer à ces différents hooks.	Elles sélectionnent le trafic et décident de l'action.

Image mentale

Table = classeur.

Chaîne = intercalaire relié à un point de passage.

Règle = consigne écrite dans cet intercalaire.

6. Configuration des tables

Créer une table

```
nft add table [family] table_name

# Exemple
nft add table inet inet_table
```

- Si la famille n'est pas renseignée, la famille IP sera sélectionnée.
- Exemple : nft add table inet inet_table

Lister les tables

```
nft list tables [family]

# Exemple
nft list tables inet
table inet inet_table
```

- Si la famille n'est pas renseignée, les tables de toutes les familles seront listées.

Supprimer une table

```
nft delete table [family] table_name

# Exemple
nft delete table inet inet_table
```

- Si la famille n'est pas renseignée, la table de la famille IP sera supprimée.
- La table doit être vide de toute règle pour pouvoir être supprimée.

Flusher une table

```
nft flush table [family] table_name

# Exemple
nft flush table inet inet_table
```

- Le flush va permettre de supprimer toutes les règles de la table choisie.
- Si la famille n'est pas renseignée, la famille IP sera sélectionnée.

À ne pas confondre

flush = vider les règles contenues dans la table.

delete = supprimer l'objet table lui-même.

7. Configuration des chaînes

Types de chaînes

Type	Familles indiquées dans le support	Hooks
filter	Toutes les familles	Tous les hooks
nat	ip, ip6, net	prerouting, input, output, postrouting
route	ip, ip6	output

- Type filter : toutes les familles et tous les hooks.
- Type nat : utilisée pour faire du NAT.
- Type route : utilisée pour rerouter les paquets.
- La priorité définit l'ordre d'exécution des règles pour le même hook, par ordre croissant.

Point de vigilance technique

Dans nftables, la famille combinée IPv4/IPv6 s'écrit inet. Le support contient la forme « net » dans la liste du type nat ; retiens inet pour la famille combinée.

Créer une chaîne

```
nft 'add chain [family] table_name chain_name { type type hook hook priority priority ; }'

# Exemple
nft 'add chain inet inet_table inet_chain_input { type filter hook input priority 0 ; }'
```

- Si la famille n'est pas renseignée, la famille IP sera sélectionnée.

Lister les chaînes

```
nft list chains [family]

# Exemple
nft list chains inet
```

- Si la famille n'est pas renseignée, les tables de toutes les familles seront listées.

Lister une chaîne

```
nft list chain [family] table_name chain_name

# Exemple
nft list chain inet inet_table inet_chain_input

table inet inet_table {
    chain inet_chain_input {
        type filter hook input priority filter; policy accept;
    }
}
```

- Seule la chaîne renseignée sera listée.

Handles : identifiants des objets

- Chaque chaîne et chaque règle possède un identifiant (ID). On va pouvoir s'en servir pour les modifier / supprimer.
- Cet affichage est possible en rajoutant l'option -a après la commande nft.

```
nft -a list chains inet

table inet inet_table {
    chain inet_chain_input { # handle 3
        type filter hook input priority filter; policy accept;
    }
}
nft delete chain [family] table_name chain_name

# Exemple
nft delete chain inet inet_table inet_chain_input
nft delete chain [family] table_name handle <handle>

# Exemple
nft delete chain inet inet_table handle 3
```

Mémo handle

Le handle est le numéro d'identité attribué par nftables.

Avant de supprimer ou remplacer un objet par son handle, affiche toujours les handles avec nft -a.

8. Configuration des règles

Créer une règle

```
nft add rule [family] table_name chain_name statement

# Exemple : refuser le ping sur le hook input
nft add rule inet inet_table inet_chain_input icmp type echo-request drop
```

- Si la famille n'est pas renseignée, la famille IP sera sélectionnée.
- La règle est ajoutée en dernier dans la liste.

Lister les règles

```
nft list ruleset [family]

# Exemple
nft list ruleset inet

table inet inet_table {
    chain inet_chain_input {
        type filter hook input priority filter; policy accept;
        icmp type echo-request drop
    }
}
```

- Si la famille n'est pas renseignée, les règles de toutes les familles seront listées.

Lister avec les handles

```
nft -a list ruleset [family]

# Exemple
nft -a list ruleset inet

table inet inet_table { # handle 1
    chain inet_chain_input { # handle 1
        type filter hook input priority filter; policy accept;
        icmp type echo-request drop # handle 2
    }
}
```

Remplacer une règle

```
nft replace rule [family] table_name chain_name handle <index> statement

# Exemple
nft replace rule inet inet_table inet_chain_input handle 2 drop
nft -a list ruleset inet

table inet inet_table { # handle 1
    chain inet_chain_input { # handle 1
        type filter hook input priority filter; policy accept;
        drop # handle 2
    }
}
```

Insérer une règle

```
nft insert rule [family] table_name chain_name handle <index> statement

# Exemple
nft insert rule inet inet_table inet_chain_input handle 2 tcp sport 9000 accept
```

- Si la famille n'est pas renseignée, la règle de la table de la famille IP sera insérée.
- La règle est insérée juste avant celle qui porte l'index saisi.

```
nft -a list ruleset inet

table inet inet_table { # handle 1
    chain inet_chain_input { # handle 1
        type filter hook input priority filter; policy accept;
        tcp sport 9000 accept # handle 3
        drop # handle 2
    }
}
```

Supprimer une règle

```
nft delete rule [family] table_name chain_name handle <handle>

# Exemple
nft delete rule inet inet_table inet_chain_input handle 2
nft -a list ruleset inet

table inet inet_table { # handle 1
    chain inet_chain_input { # handle 1
        type filter hook input priority filter; policy accept;
    }
}
```

9. Ordre des règles

- L'ordre de création des règles est très important : la première règle à laquelle le paquet correspondra sera appliquée.
- Exemple : on cherche à n'accepter que les pings.

Mauvais ordre

```
nft -a list ruleset

table inet inet_table { # handle 1
    chain inet_chain { # handle 1
        type filter hook input priority filter; policy accept;
        drop # handle 2
        icmp type echo-request accept # handle 3
    }
}
```

- Ici tous les paquets seront bloqués et donc la règle sur les paquets icmp de type echo-request entrants sera ignorée.

Bon ordre

```
nft -a list ruleset

table inet inet_table { # handle 1
    chain inet_chain { # handle 1
        type filter hook input priority filter; policy accept;
        icmp type echo-request accept # handle 2
        drop # handle 3
    }
}
```

Règle d'or

Les règles sont lues de haut en bas.

Place les exceptions autorisées avant la règle générale de blocage.

10. Options des règles : logs

- On peut générer des logs pour chaque règle.
- Contrairement à iptables, on peut le faire à la création de la règle.
- nft add rule [family] table_name chain_name statement log [prefix] policy
- Exemple : on refuse le ping et on log cette action avec le préfixe facultatif.

```
# Création de la table
nft add table inet inet_table

# Création de la chaîne
nft 'add chain inet inet_table inet_chain_input { type filter hook input priority 0 ; }'

# Création de la règle
nft 'add rule inet inet_table inet_chain_input icmp type echo-request log prefix "Attempting ping :" drop'

# Suivre les logs en temps réel
journalctl -f
```

Lecture simple

icmp type echo-request = sélectionner les demandes de ping.

log prefix "Attempting ping :" = écrire une trace avec ce préfixe.

drop = bloquer le paquet.

11. Options des règles : compteur

- Pour afficher le nombre de paquets et la quantité de données en octets qui transitent par une règle.
- nft add rule [family] table_name chain_name statement counter policy
- Exemple : on se sert du counter pour monitorer le flux sortant vers le port 443.

```
# Ajout de la chaîne pour filtrer sur le hook output
nft 'add chain inet inet_table inet_chain_output { type filter hook output priority 0 ; }'

# Ajout de la règle pour monitorer le trafic https
nft add rule inet inet_table inet_chain_output tcp dport 443 counter accept

# Après avoir généré du trafic
nft -a list ruleset inet

chain inet_chain_output { # handle 6
    type filter hook output priority filter; policy accept;
    tcp dport 443 counter packets 1241 bytes 104727 accept # handle 7
}
```

Mémo counter

packets 1241 = nombre de paquets ayant correspondu à la règle.

bytes 104727 = volume total de données correspondant à la règle.

12. Sauvegarde et restauration des règles

Sauvegarde des règles

- On affiche les règles complètes et on redirige la sortie dans un fichier.

```
nft list ruleset > nft_rules_backup.nft
nft list ruleset > /etc/nftables.conf
```

Restauration des règles

```
nft -f nft_rules_backup.nft
```

Pour aller plus loin

- man nft
- Connection Tracking

Attention

Une règle créée en ligne de commande n'est pas automatiquement persistante après un redémarrage.

La sauvegarde dans /etc/nftables.conf permet de préparer une configuration persistante selon le service nftables du système.

13. Pour aller plus loin : accepter seulement SSH, HTTP et HTTPS

```
nft add table inet filter
nft 'add chain inet filter input { type filter hook input priority 0 ; }'
nft 'add chain inet filter forward { type filter hook forward priority 0 ; }'
nft 'add chain inet filter output { type filter hook output priority 0 ; }'
nft add rule inet filter input ct state {established, related} accept
nft add rule inet filter input ct state invalid drop
nft add rule inet filter input iifname lo accept
nft add rule inet filter input tcp dport 22 accept
nft add rule inet filter input tcp dport 80 accept
nft add rule inet filter input tcp dport 443 accept
nft add rule inet filter input drop
nft add rule inet filter forward drop
```

Résultat

```
table inet filter {
   chain input {
      type filter hook input priority filter; policy accept;
      # allow established/related connections
      ct state {established, related} accept
      # early drop of invalid connections
      ct state invalid drop
      # allow from loopback
      iifname "lo" accept
      # allow ssh,http,https
      tcp dport 22 accept
      tcp dport 80 accept
      tcp dport 443 accept
      # everything else
      drop
   }
   chain forward {
      type filter hook forward priority filter; policy accept;
      drop
   }
   chain output {
      type filter hook output priority filter; policy accept;
   }
}
```

Lecture pédagogique

Les connexions déjà établies ou liées sont autorisées.

Les connexions invalides sont bloquées rapidement.

La boucle locale est autorisée.

Les ports 22, 80 et 443 sont autorisés.

Tout le reste du trafic entrant est bloqué.

Le forwarding est bloqué.

14. Pour aller plus loin : bannir un block d'IP

```
nft add table inet filter
nft 'add chain inet filter input { type filter hook input priority filter ; }'
nft 'add chain inet filter forward { type filter hook forward priority filter ; }'
nft 'add chain inet filter output { type filter hook output priority filter ; }'
nft add rule inet filter input ip saddr 192.168.2.0/24 drop
nft add rule inet filter forward drop
```

Résultat

```
table inet filter {
   chain input {
      type filter hook input priority filter; policy accept;
      ip saddr 192.168.2.0/24 drop
   }
   chain forward {
      type filter hook forward priority filter; policy accept;
      drop
   }
   chain output {
      type filter hook output priority filter; policy accept;
   }
}
```

Traduction

ip saddr 192.168.2.0/24 = sélectionner les paquets dont l'adresse IP source appartient à ce réseau.

drop = bloquer silencieusement ces paquets.

15. Pour aller plus loin : nftables et Fail2Ban

- Utiliser nftables pour logger une activité ping dans le journal et fail2ban pour répérer cettte activité et prendre des mesures.

```
nft add rule inet filter input icmp type echo-request log prefix "Attempting ping :" drop
```

Fichier filter custom de Fail2Ban

```
/etc/fail2ban/filter.d/nftables-ping-custom.conf

[INCLUDES]
before = common.conf
[Definition]
failregex = Attempting ping.*SRC=<HOST>
```

Fichier jail.conf

```
[nftablespingcustom]
filter = nftables-ping-custom
enabled = true
```

Chaîne logique

nftables journalise le ping et le bloque.
Fail2Ban analyse les journaux.
Le filtre repère l'adresse source grâce à <HOST>.
La jail active le filtre et permet à Fail2Ban de prendre des mesures.

16. Fiche mentale finale

Notion	Définition	Exemple
Famille	Type de trafic traité	inet, ip, ip6, netdev, bridge, arp
Table	Conteneur de chaînes	nft add table inet filter
Chaîne	Conteneur de règles rattaché à un hook	hook input, output, forward...
Hook	Point d'ancrage dans le chemin du paquet	input, output, forward, prerouting...
Règle	Critères + action	tcp dport 22 accept
Handle	Identifiant d'un objet	nft -a list ruleset
Policy	Action par défaut de la chaîne	policy accept / drop

Ordre à réciter

Famille → Table → Chaîne → Hook → Règle → Action.
Les règles sont lues dans l'ordre : la première correspondance gagne.

Commande de vérification principale

nft list ruleset : afficher toute la configuration.
nft -a list ruleset : afficher toute la configuration avec les handles.