Netfilter et Iptables

I Présentation

I.1 Rôle du logiciel

Le logiciel Netfilter est le pare-feu officiel sous Linux. C'est lui qui est chargé d'effectuer les opérations (de sécurité, mais pas uniquement) sur les paquets transitant sur les interfaces réseau gérées par le système d'exploitation.

Il fournit à Linux :

- des fonctions de **pare-feu** et notamment le contrôle des machines qui peuvent se connecter, sur quels ports, de l'extérieur vers l'intérieur, ou de l'intérieur vers l'extérieur du réseau ;
- de **traduction d'adresse (NAT)** pour partager une connexion internet, masquer des machines du réseau local, ou rediriger des connexions ;
- et d'historisation du trafic réseau.

Netfilter intercepte les paquets réseau à différents endroits du système (à la réception, avant de les transmettre aux processus, avant de les envoyer à la carte réseau, etc.). Les paquets interceptés passent à travers des chaînes qui vont déterminer ce que le système doit faire avec le paquet. En modifiant ces chaînes, on va pouvoir bloquer certains paquets et en laisser passer d'autres.

En principe, le paquet « **iptables** » est installé d'origine sous Debian mais si ce n'était pas le cas, rien de plus simple à faire :

```
root@debian:~# apt-get install iptables
```

Nous n'allons pas étudier dans ce document toutes les possibilités de NetFilter mais juste deux utilisations courantes.

I.2 Principe

Nous allons définir des règles qui vont déterminer la nature des opérations que l'on souhaite réaliser sur le paquet. Par exemple, les paquets qui entrent dans le routeur par l'interface eth0 et dont le port est 80 (HTTP), sont rejetés ou acceptés.

Ces règles seront stockées dans des tables en fonction de leur utilisation. Trois tables principales sont utilisées par Netfilter :

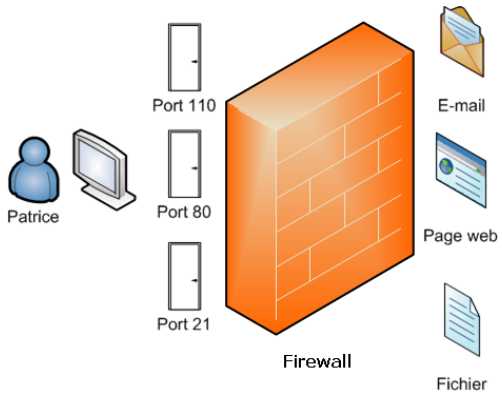
- la première est la table **filter** qui, comme son nom l'indique, permet de supprimer des paquets qui sont interdits ;
- la seconde est la table **nat** qui permet d'effectuer des modifications sur les adresses IP et les ports en vue de mettre en place un routage avec spécificité NAT ;
- la troisième est la table **mangle** qui permet d'effectuer toutes sortes de modifications et de marquage sur des paquets. Cette table ne sera pas étudiée dans ce document.

II Filtrage

II.1 Présentation des pare-feux

L'objectif du filtrage est de réaliser un pare-feu. Le fait de garder des agresseurs distants à l'extérieur d'un LAN représente un aspect important de la sécurité réseau, voire le *plus* important. L'intégrité d'un LAN devrait être protégée contre les utilisateurs distants malveillants grâce à l'utilisation de règles rigoureuses de pare-feu. C'est avant tout une question de sécurité. Seuls les programmes que vous considérez comme « sûres » et que vous utilisez, peuvent communiquer sur le réseau.

Par exemple, si je veux empêcher toute connexion FTP (parce que je considère que le FTP n'est pas sûr), je peux souhaiter bloquer le port 21 (utilisé par FTP). Par contre, les réponses à des connexions HTTP sont autorisées pour pouvoir naviguer sur le Web.



Avoir un pare-feu ne vous prémunit pas contre les virus (bien que sous Linux, ils restent rares). En revanche, cela rend la tâche particulièrement difficile aux pirates qui voudraient accéder à votre machine.

II.2 Les chaînes

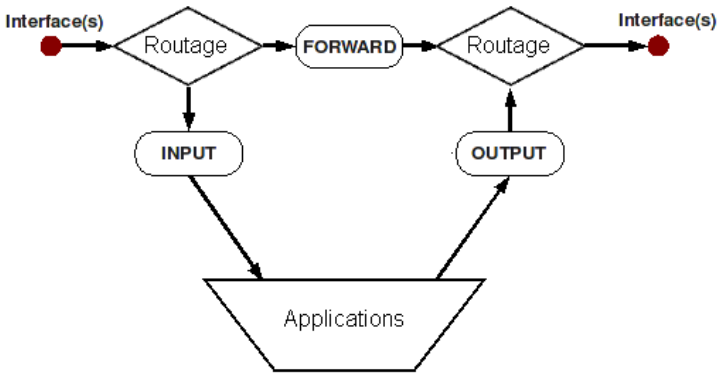
Dans son fonctionnement de base, Netfilter permet de jeter ou de laisser passer les paquets qui entrent et qui sortent.

Pour effectuer du filtrage, il faut ajouter des règles dans la table « **filter** ». Cette table est celle par défaut donc il n'est pas nécessaire de le préciser dans la commande.

Les règles vont s'appliquer selon des chaînes qui dépendent si le paquet entre, sort ou traverse le routeur.

Les trois chaînes principales sont :

- la chaîne **INPUT** : le paquet est à destination de la machine et s'apprête à contacter une application de la machine
- la chaîne **OUTPUT** : le paquet vient d'être émis par une application de la machine
- la chaîne **FORWARD** : le paquet vient d'être transféré depuis une interface vers une autre (un routage sera effectué avec le paquet)



II.3 Syntaxe

La syntaxe pour ajouter une règle à la table filter est la suivante :

```
iptables -t filter -A CHAINE Condition(s) -j ACCEPT|DROP|REJECT
```

Comme indiqué précédemment, CHAINE peut prendre pour valeur INPUT, OUTPUT ou FORWARD.

Le paquet doit répondre à une ou plusieurs conditions pour que la cible soit exécutée dessus. Elles peuvent être multiples. Elles définissent le type de paquet, par quelle interface il entre ou il sort du routeur.

L'option -j indique ce qu'il faut faire si le paquet correspond à la règle :

- ACCEPT : le paquet est autorisé à traverser le pare-feu ;
- DROP : le paquet n'est pas autorisé à traverser le pare-feu, le destinataire n'est pas prévenu que le paquet a été supprimé ;
- REJECT : le paquet n'est pas autorisé à traverser le pare-feu, le destinataire est prévenu que le paquet a été supprimé.

Les conditions permettent de définir les types des paquets à traiter, par exemple de quel réseau vient le paquet, du protocole de la couche 4, … Les conditions sont :

- ✗ -s suivi d'une adresse IP source (xxx.xxx.xxx.xxx) ou un réseau entier (xxx.xxx.xxx.xxx/nbr1) indique de quel réseau vient le paquet à traiter
- ✗ -d suivi d'une adresse IP destination (écriture identique) indique vers quel réseau va le paquet à traiter
- ✗ -i suivi d'une interface d' entrée (ethX) indique de quelle interface vient le paquet à traiter
- ✗ -o suivi d'une interface de sortie indique vers quelle interface va le paquet à traiter
- ✗ -p suivi d'un protocole de couche 4 (tcp, udp, icmp, etc.) indique le protocole de la couche supérieure
- ✗ -m suivi du même protocole de couche 4 permet d'ajouter des conditions supplémentaires sur le protocole :
  - ✔ --sport suivi d'un numéro de port source ou d'une plage de ports source (xxxxx-yyyyy pour tous les ports de xxxxx à yyyyy) ;
  - ✔ --dport suivi d'un numéro de port destination ou d'une plage.
- ✗ --state permet de spécifier l'état du paquet à matcher parmi les états suivants :
  - ✔ ESTABLISHED : paquet associé à une connexion déjà établie
  - ✔ NEW : paquet demandant une nouvelle connexion
  - ✔ INVALID : paquet associé à une connexion inconnue
  - ✔ RELATED : Nouvelle connexion mais liée, idéal pour les connexions FTP
- ✗ -f indique un paquet fragmenté
- ✗ --tcp-flags spécifie un flag tcp à matcher : SYN ACK FIN RST URG PSH ALL NONE
- ✗ --icmp-type spécifie un type de paquet icmp à matcher
- ✗ --mac-source spécifie l'adresse MAC à matcher

Le "!" peut être utilisé pour certaines commandes afin de spécifier le contraire (on peut le traduire par "sauf").

N.B. : **les règles sont vérifiées dans l'ordre où elles sont inscrites lors de la configuration et dès qu'une correspondance a été trouvée, le reste des règles de la chaîne est ignoré.** Si aucune correspondance n'est trouvée, une règle par défaut est spécifiée grâce à l'action P (policy).

II.4 Exemples

- ✗ Visualisation des règles existantes avec leur numéro pour la table filter :

```
root@debian:~# iptables -L --line-numbers
```

- ✗ Ajout à la chaîne FORWARD de la table filter, l'ordre de rejeter tous les paquets TCP utilisant le port destination 80

```
root@debian:~# iptables -t filter -A FORWARD -p tcp -m tcp --dport 80 -j REJECT
```

- ✗ Réinitialisation de toutes les règles :
  - ✔ vide toutes les commandes iptables

```
root@debian:~# iptables -F
```

  - ✔ supprime toutes les commandes iptables

```
root@debian:~# iptables -X
```

- ✗ Suppression de la règle 2 de la chaîne OUTPUT :

```
root@debian:~# iptables -D OUTPUT 2
```

- ✗ Refus de toutes les actions sauf celles prédéfinies :

```
root@debian:~# iptables -P INPUT DROP
root@debian:~# iptables -P FORWARD DROP
root@debian:~# iptables -P OUTPUT ACCEPT
```

- ✗ Permission d'une connexion déjà ouverte de recevoir du trafic :

```
root@debian:~# iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

✗ Permission du trafic entrant par l'interface eth0 sur le port 22 (SSH) :

```
root@debian:~# iptables -A INPUT -p tcp -i eth0 --dport ssh -j ACCEPT
```

✗ Autorisation pour le PC à faire des pings sur des IP externes et à répondre aux requêtes "ping" :

```
root@debian:~# iptables -A OUTPUT -p icmp -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
```

✗ Autorisation des pings :

```
root@debian:~# iptables -A INPUT -p icmp -j ACCEPT
```

✗ Refus de tout trafic TCP sauf ce qui provient de l'adresse IP 10.42.42.42 :

```
root@debian:~# iptables -A INPUT -p tcp --source ! 10.42.42.42 -j DROP
```

✗ Rejet de tout paquet ICMP entrant :

```
root@debian:~# iptables -A INPUT -p icmp -j DROP
```

✗ Acceptation des paquets TCP venant du PC 192.168.42.42 :

```
root@debian:~# iptables -A INPUT -p tcp -s 192.168.42.42 -j ACCEPT
```

✗ Acceptation des paquets TCP vers le PC 10.1.0.1 traversant le routeur :

```
root@debian:~# iptables -A FORWARD -p tcp -d 10.1.0.1 -j ACCEPT
```

✗ Pour accepter tout ce qui se passe sur le réseau local 192.168.1.0/24 :

```
root@debian:~# iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
root@debian:~# iptables -A OUTPUT -d 192.168.1.0/24 -j ACCEPT
root@debian:~# iptables -A FORWARD -s 192.168.1.0/24 -j ACCEPT
```

✗ Acceptation des résolutions de nom (ie: le dns soit le port 53) :

```
root@debian:~# iptables -A INPUT -i ppp0 --protocol udp --source-port 53 -j
ACCEPT
root@debian:~# iptables -A OUTPUT -o ppp0 --protocol udp --destination-port 53 -j
ACCEPT
root@debian:~# iptables -A INPUT -i ppp0 --protocol tcp --source-port 53 -j
ACCEPT
root@debian:~# iptables -A OUTPUT -o ppp0 --protocol tcp --destination-port 53 -j
ACCEPT
```

✗ Acceptation du trafic web (on veut surfer soit le port 80) :

```
root@debian:~# iptables -A INPUT -i ppp0 --protocol tcp --source-port 80 -m state
--state ESTABLISHED -j LOG_ACCEPT
root@debian:~# iptables -A OUTPUT -o ppp0 --protocol tcp --destination-port 80 -m
state --state NEW,ESTABLISHED -j LOG_ACCEPT
```

✗ Acceptation de tout le trafic web (www soit 80) entrant :

```
root@debian:~# iptables -A INPUT -p tcp -i eth0 --dport 80 -j ACCEPT
```

✗ Autorisation du trafic entrant des segments qui essaient d'établir une connexion :

```
root@debian:~# iptables -A INPUT -p tcp --tcp-flags SYN,FIN,ACK SYN -j ACCEPT
```

III NAT

III.1 Présentation

La translation d'adresse réseau (NAT) translate n adresses privées d'un réseau en une adresse publique par exemple. Cela permet à n ordinateurs de se connecter à Internet avec une seule adresse publique (parfois plusieurs).

La table **nat** est celle qui va permettre de modifier les adresses IP et les ports afin de pouvoir réaliser du NAT. Elle va pouvoir modifier les adresses IP et ports sources (avec comme cible "SNAT" pour pouvoir faire accéder à internet les

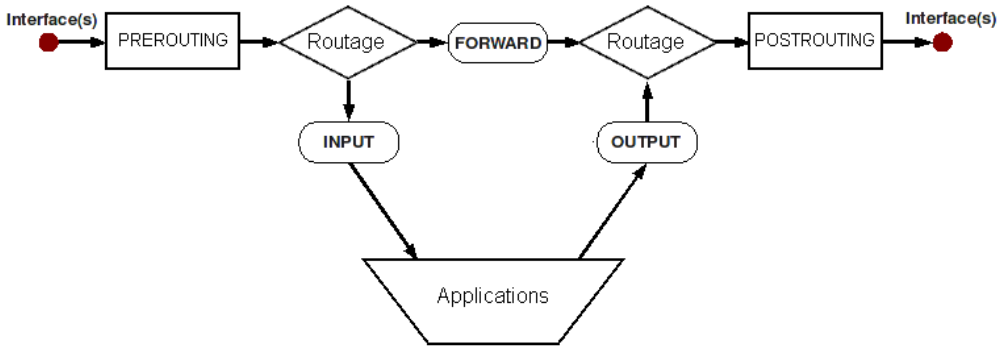
postes se trouvant dans le LAN NAT) ou bien les adresses IP et ports destination (avec comme cible "DNAT" pour pouvoir accéder à un poste se trouvant dans le LAN NAT depuis internet).

Deux cas de translation existent :

1. Les postes privés veulent accéder à Internet (adresse publique). Dans ce cas il faut préciser comme cible SNAT.
2. Les ordinateurs sur le réseau publique veulent accéder à un service d'une machine du réseau privé (exemple serveur WEB accessible depuis Internet). Il faut préciser la cible DNAT.

Il faut dans ce cas utiliser deux autres chaînes :

- ✗ la chaîne **PREROUTING** : le paquet arrive sur une interface réseau
- ✗ la chaîne **POSTROUTING** : le paquet s'apprête à sortir d'une interface réseau



III.2 Syntaxe

La syntaxe pour ajouter une règle à la table nat est la suivante :

```
iptables -t nat -A CHAINE Condition(s) -j SNAT|DNAT Paramètres_du_SNAT_ou_du_DNAT
```

CHAINE peut prendre pour valeur PREROUTING, POSTROUTING ou OUTPUT.

Les conditions peuvent être les mêmes que pour la table filter et surtout les deux suivantes :

- ✗ --to-destination utilisé en target pour le DNAT, permet de spécifier l'adresse de destination de la translation, on peut également spécifier un port s'il est différent du port source
- ✗ --to-source : Utilisé pour en target pour le SNAT, permet de spécifier l'adresse source de la translation

La cible peut être de deux types :

- ✗ SNAT : On substitue une adresse source dans un paquet sortant à son adresse source d'origine.
- ✗ DNAT : On substitue à l'adresse de destination des paquets provenant du réseau public, une adresse du réseau local privé.

La cible doit être suivie des paramètres qui indique la traduction d'adresse à réaliser.

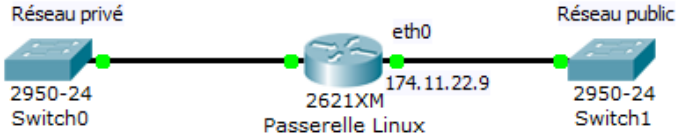
La cible SNAT doit obligatoirement s'exécuter au niveau de la chaîne POSTROUTING.

La cible DNAT doit obligatoirement s'exécuter au niveau de la chaîne PREROUTING.

III.3 Exemples

- ✗ Les adresses dans les paquets sortants du réseau privé seront substituées en une adresse publique (**PAT**) :

```
root@debian:~# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source
174.11.22.9          Adresse publique : 174.11.22.9
```



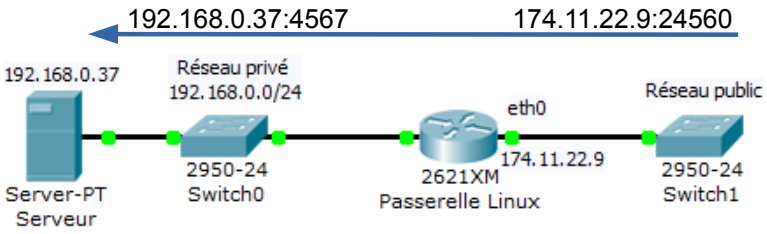
Cette commande ajoute une règle dans la chaîne POSTROUTING de la table nat qui modifie tous les paquets qui sortent par l'interface eth0. La modification lance la cible SNAT et change l'adresse IP source pour la valeur 174.11.22.9.

Remarque : Si vous avez deux réseaux privés connectés à la passerelle Linux en plus du réseau public, la commande est la même.

✗ Substitution de l'adresse de destination des paquets provenant du réseau public en une adresse du réseau local privé (**redirection de port**) :

Un paquet venant du réseau public et qui a pour adresse destination, l'adresse de la passerelle (interface eth0) et le port 24560 sera translaté. Ce paquet est envoyé au serveur du réseau privé 192.168.0.37 et le port devient 4567.

```
root@debian:~# iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 24560
-j DNAT --to-destination 192.168.0.37:4567
```



Cette commande ajoute une règle dans la chaîne PREROUTING de la table nat qui modifie tous les paquets qui entrent par l'interface eth0 et dont le port tcp utilisé est le 24560, la modification lance la cible DNAT et change l'adresse IP destination pour la valeur 192.168.0.37 et le port destination pour la valeur 4567.

✗ Visualisation des règles existantes avec leur numéro pour la table nat :

```
root@debian:~# iptables -t nat -L --line-numbers
```

✗ Suppression de la règle 1 de la chaîne PREROUTING :

```
root@debian:~# iptables -t nat -D PREROUTING 1
```

✗ Suppression de toutes les règles de la chaîne POSTROUTING de la table nat

```
root@debian:~# iptables -t nat -F POSTROUTING
```

✗ Suppression de la règle 1 de la chaîne PREROUTING :

```
root@debian:~# iptables -t nat -D PREROUTING 1
```

IV Sauvegarde et restauration des iptables

Toutes les commandes iptables entrées sont appliquées immédiatement. Mais si vous redémarrez l'ordinateur, il faudra les retaper. Pour retrouver les règles d'iptables au redémarrage de la machine, il faut donc penser à les sauvegarder et les restaurer.

La sauvegarde se réalise de cette manière :

```
root@debian:/etc# cd                              pour aller dans le répertoire /root
root@debian:~# iptables-save > sauvegarde-iptables
```

Pour restaurer les iptables, il faut lancer cette commande :

```
root@debian:~# iptables-restore < sauvegarde-iptables
```

Attention : Pour que les règles iptables soient restaurées automatiquement à chaque démarrage, il suffit d'ajouter la commande précédente dans le fichier **« /etc/rc.local »**.