Routage et filtrage

1. Configuration d'un serveur Linux en tant que routeur

La fonction de routage est intégrée nativement dans le noyau Linux. Il n'y a donc pas de questions à se poser, toute machine Linux est un routeur en puissance. En revanche, cette fonction n'est pas active par défaut au démarrage. Il faut donc la configurer avant toute opération de routage.

a. Activation du routage sur un serveur Linux

Nous savons que tout système Linux présente un filesystem virtuel **/proc** qui permet d'observer en direct un certain nombre de comportements et paramètres. L'activation du routage se fait en modifiant le contenu du fichier **/proc/sys/net/ipv4/ip_forward**. Ce fichier contient un seul caractère, par défaut **0** pour indiquer que le routage est inactif.

Modification du fichier ip_forward pour activer le routage

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Une fois cette manipulation effectuée, la machine Linux est prête à router les paquets se présentant sur ses interfaces. Ce paramètre est volatile et sera perdu dès la machine éteinte. Toutefois, on peut évidemment annuler le routage en effectuant l'opération inverse.

Modification du fichier ip_forward pour désactiver le routage

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

Autre possibilité, la commande **sysctl** qui permet de modifier dynamiquement des paramètres fonctionnels du noyau. **sysctl** permet de modifier directement tous les fichiers se trouvant sous l'arborescence **/proc/sys**.

Activation du routage avec sysctl

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

Ces commandes sont effectives toute la durée de la session et doivent être retapées après chaque redémarrage. On peut bien entendu les placer dans un script de service appelé au démarrage, ou modifier le fichier **/etc/sysctl.conf**.

Activation permanente du routage dans le fichier /etc/sysctl.conf

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

b. Consultation de la table de routage

À ce stade, le routeur Linux est parfaitement capable de router les paquets. Toutefois, il ne pourra le faire que vers des réseaux connus, c'est-à-dire référencés dans sa table de routage.

La table de routage est maintenue en mémoire mais elle peut être consultée par quelques commandes.

Affichage de la table de routage par la commande route

```
route -n
```

Le paramètre -n est facultatif, mais il fait gagner beaucoup de temps à l'affichage car il dispense la commande de tenter de résoudre les adresses renvoyées en noms. Or, si l'adresse en question n'est pas renseignée dans une zone DNS inverse, cette requête se fait pour rien et il faut attendre plusieurs secondes pour que l'affichage arrive.

Affichage de la table de routage par la commande netstat

```
netstat -nr
```

Où l'option -r demande à la commande d'afficher la table de routage et -n de ne pas faire de résolution de noms. La

commande **netstat** a de nombreux usages, mais elle est souvent utilisée dans ce simple cadre de consultation de la table de routage.

Exemple d'affichages de table de routage

L'affichage de la table de routage est souvent le seul moyen simple de consulter la valeur de la passerelle par défaut.

```
beta:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
```

c. Gestion des routes statiques

Les seules entrées présentes automatiquement dans la table de routage sont les réseaux auxquels le routeur est directement connecté, ainsi que la passerelle par défaut. Le routeur peut donc exploiter ces entrées de la table de routage sans autre configuration. Si le routeur doit router des paquets vers d'autres réseaux, il faudra ajouter manuellement les routes dans la table de routage.

Ajout de route statique dans la table de routage

`route add -net` *`réseau_cible`* `netmask` *`masque`* `gw` *`routeur`*

Ajout de route statique : options et paramètres	
-net	La route ajoutée est celle d'un réseau. (La cible pourrait être un hôte seul même si cette configuration est moins fréquente.)
réseau_cible	L'adresse du réseau que la nouvelle route permet d'atteindre.
masque	Le masque de sous-réseaux associé à la nouvelle route.
gw *routeur*	Indique le routeur à emprunter pour atteindre le réseau cible.

Ajout de passerelle par défaut

`route add default gw` *`routeur`*

`route add -net 0.0.0.0 gw` *`routeur`*

Dans la deuxième syntaxe, 0.0.0.0 représente la route par défaut. Cette représentation de la route par défaut est universelle et applicable sur la quasi-totalité des systèmes exploitant une table de routage IP.

Bien entendu, il est possible de supprimer les routes statiques qui ne sont plus nécessaires ou enregistrées par erreur.

Suppression de route statique de la table de routage

`route del -net` *`réseau_cible`* `netmask` *`masque`*

Exemple d'ajout de route

```
beta:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
beta:~# route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.1.99
beta:~# route -n
```

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
10.0.0.0        192.168.1.99    255.0.0.0       UG    0      0        0 eth1
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
```

Exemple de suppression de route

```
beta:~# route del -net 10.0.0.0 netmask 255.0.0.0
beta:~# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
beta:~#
```

2. Iptables

Les iptables sont utilisées pour gérer le filtrage de paquets IP au sein d'un système Linux. Elles exploitent une commande unique : **iptables**, et se configurent par l'application successive de règles de gestion de paquets. Les iptables peuvent filtrer le trafic en transit dans un routeur Linux, mais aussi le trafic entrant et sortant de tout serveur ou poste de travail à une seule interface.

Si les iptables constituent un outil très puissant de gestion du trafic, la médaille a son revers et leur configuration est tout sauf intuitive. Avec une approche structurée, on peut toutefois assez rapidement appréhender leur fonctionnement. Les paragraphes ci-dessous exposent les concepts fondamentaux des iptables, afin de les utiliser plus tard dans des configurations de pare-feu.

a. Les tables

Les iptables s'appuient sur des tables associées à un mode fonctionnel. Selon le type de règle que l'on souhaite ajouter au fonctionnement des iptables, on précisera la table associée. Les tables principales utilisées sont **filter** pour le filtrage de paquets et **nat** pour la translation d'adresses entre un réseau privé et un réseau public.

La table **filter** est la table par défaut. Aussi, quand on établit une règle iptables dans un but de filtrer les paquets est-elle sous-entendue et donc non précisée.

La table **nat** sert à la translation d'adresses et doit être systématiquement précisée quand elle est invoquée.

b. Les chaînes

Une chaîne iptables représente un type de trafic du point de vue de sa circulation dans une machine. Les chaînes permettent de préciser si une règle doit s'appliquer à du trafic qui entre dans une machine, qui en sort ou qui la traverse.

La chaîne **INPUT** désigne le trafic entrant, la chaîne **OUTPUT** désigne le trafic sortant, et la chaîne **FORWARD** désigne le trafic qui traverse la machine, entrant par une interface et sortant par une autre. Attention, même si un paquet qui traverse le routeur est d'un point de vue physique respectivement entrant, traversant et sortant, iptables le considérera comme traversant seulement (chaîne FORWARD). Les chaînes INPUT et OUTPUT sont réservées au trafic à destination ou en provenance explicite de l'hôte soumis aux règles.

Une autre chaîne appelée **POSTROUTING** et utilisée dans la configuration du NAT a pour objet d'appliquer un traitement à un paquet après une opération de routage.

Les chaînes sont toujours indiquées en majuscules dans une syntaxe iptables.

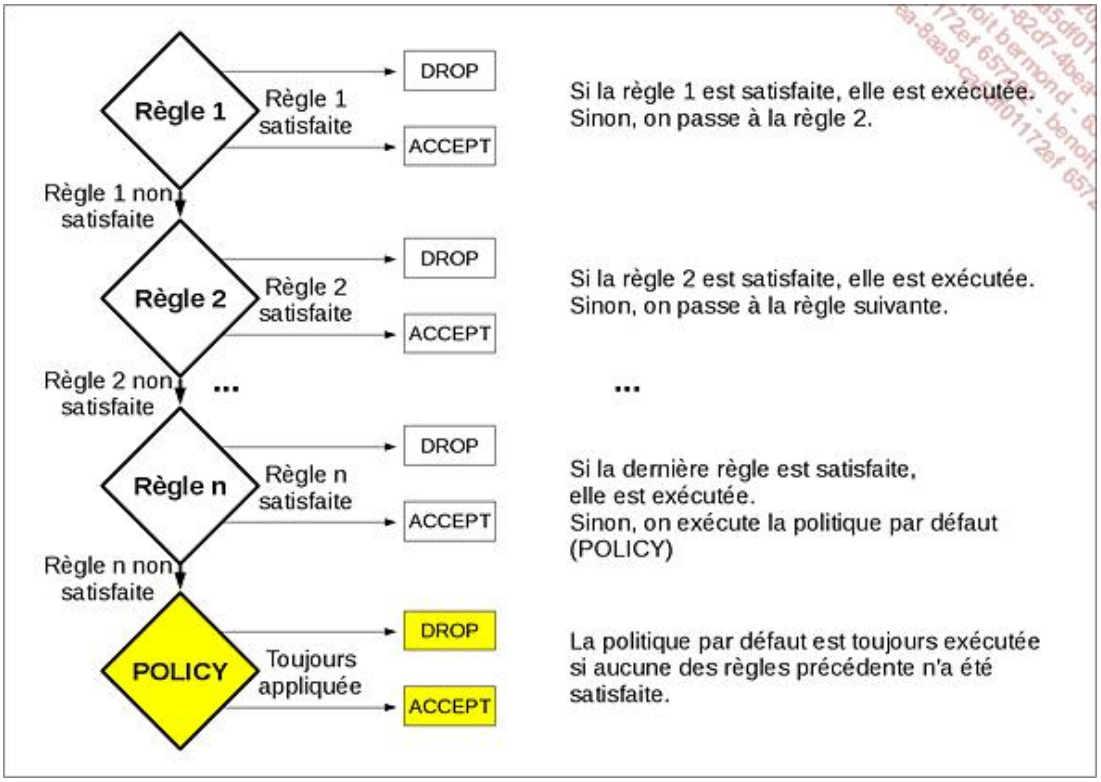
c. Les actions

Quand une règle est satisfaite, une action est engendrée par le système sur le paquet testé. Les principales actions sont **ACCEPT** qui laisse passer le paquet et **DROP**, qui le détruit.

Dans une syntaxe iptables, l'action (**target** dans le manuel en ligne) est annoncée par le paramètre **-j**.

Les actions sont toujours indiquées en majuscules dans une syntaxe iptables.

d. Le traitement des règles



Les règles sont appliquées une par une à tout paquet filtré. Si une règle est satisfaite, une action est engagée sur le paquet et le traitement s'arrête. Si une règle n'est pas satisfaite, la règle suivante est testée. Dans le cas où aucune des règles n'est satisfaite, le paquet subit un traitement par défaut paramétré dans une règle spécifique appelée politique (policy).

Il est possible d'afficher les règles appliquées dans l'ordre pour chacune des chaînes.

Affichage des règles effectives

```
iptables -L
```

Exemple d'affichage des règles

Cet exemple affiche les règles en vigueur sur un système Linux non configuré. On y voit la politique appliquée pour chacune des chaînes, et on constate l'absence de règles de filtrage.

```
alpha:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

La commande **iptables -L** affiche une interprétation des règles en vigueur. Si on souhaite connaître les syntaxes qui ont permis d'établir ces règles, il est préférable d'utiliser l'option **-S**.

Exemple d'affichage des règles selon les syntaxes

*L'option -**S** est particulièrement utile quand on est confronté à un système configuré par un tiers et qu'on ne sait pas quelles sont les commandes qui ont conduit à une configuration.*

```
alpha:~# iptables -S
```

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
alpha:~#
```