Administration d'un pare-feu avec les iptables

1. Politiques

a. Principe des politiques de pare-feu

Un pare-feu peut fonctionner selon deux modes distinct : « tout ce qui n'est pas autorisé est interdit », ou « tout ce qui n'est pas interdit est autorisé ». Pour définir le comportement par défaut, les iptables permettent de définir pour chaque chaîne une action par défaut.

Définition de la politique par défaut des iptables

```
iptables -P chaine action
```

Où *chaine* représente le type de trafic (INPUT, OUTPUT et FORWARD), et action le comportement souhaité (DROP ou ACCEPT).

Exemple de définition de politique

Dans cet exemple, on interdit à tout trafic de sortir de l'hôte en appliquant une politique de rejet des paquets sortants.

```
root@test:~$ ping -c 1 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
64 bytes from 192.168.0.10: icmp_seq=1 ttl=64 time=0.880 ms
--- 192.168.0.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.880/0.880/0.880/0.000 ms
root@test:~$ iptables -P OUTPUT DROP
root@test:~$ ping -c 1 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
--- 192.168.0.10 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
root@test:~$
```

b. Configuration d'une politique de base

Si l'hôte à configurer est appelé à devenir un pare-feu, il est probable que tout trafic soit interdit par défaut. Cette configuration courante consiste à définir sur les trois chaînes INPUT, OUTPUT et FORWARD une politique de non-traitement des paquets.

Configuration d'une politique restrictive

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

2. Filtrage de paquets

a. Politique et règles

Après avoir configuré une politique qui décrit le comportement de base du pare-feu, il faut créer des règles spécifiques aux éléments de trafics que l'on souhaite laisser passer ou interdire. La philosophie du pare-feu est : on définit le comportement général avec les politiques, et on gère au cas par cas les comportements spécifiques avec des règles.

b. Création de règle

Pour chaque élément de trafic devant être autorisé ou interdit, il faudra créer une règle spécifique.

Syntaxe création d'une règle de gestion de trafic

```
iptables -A chaine -s ip_source -d ip_dest -p protocole --dport port -j action
```

iptables : création de règle	
-A *chaine*	On ajoute une règle dans la chaîne *chaine* (INPUT, OUTPUT ou FORWARD).
-s *ip_source*	Facultatif : l'adresse IP source d'où proviennent les paquets soumis à la règle. Si l'adresse est une adresse de réseau, préciser le masque.
-d *ip_dest*	Facultatif : l'adresse IP de destination vers laquelle vont les paquets soumis à la règle. Si l'adresse est une adresse de réseau, préciser le masque.
-p *protocole*	Indique le protocole utilisé dans le paquet soumis à la règle. Valeurs courantes : udp, tcp, icmp.
--dport *port*	Facultatif : indique le port de destination du paquet soumis à la règle.
-j *action*	Indique comment traiter le paquet soumis à la règle. (ACCEPT ou DROP).

Autorisation des ping sortant et entrant

Chaque type de flux doit faire l'objet d'une règle iptable.

```
alpha:~# iptables -A OUTPUT -p icmp -j ACCEPT
alpha:~# iptables -A INPUT -p icmp -j ACCEPT
alpha:~#
```

Autorisation du trafic http traversant en provenance d'un réseau

```
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp -dport 80 -j ACCEPT
alpha:~#
```

Une configuration erronée sur un pare-feu peut avoir des conséquences dramatiques. Il est recommandé pour vérifier sa bonne configuration d'utiliser un scanner de ports depuis une machine distante. La commande nmap -F suivie de l'adresse IP de la machine protégée permet de vérifier très rapidement (Fastmode) que les ports sont bien bloqués ou ouverts.

c. Gestion des règles

Les règles sont appliquées dans leur ordre de création et le système leur applique automatiquement un numéro d'ordre.

Affichage des numéros de règles effectives

```
iptables -L chaine --line-numbers -n
```

Où *chaine* représente la chaîne de traitement (INPUT, OUTPUT ou FORWARD). Le paramètre -n n'est pas obligatoire, mais accélère fortement l'affichage en dispensant la commande de tenter de résoudre les adresses en noms.

Suppression d'une règle

```
iptables -D chaine numéro
```

Où *numéro* représente le numéro de la ligne obtenu avec la commande précédente et où *chaine* représente la chaîne

de traitement (INPUT, OUTPUT ou FORWARD).

Insertion d'une règle

```
iptables -I chaine numéro conditions -j action
```

Où *conditions* représente les critères de sélection du paquet soumis à la règle (adresses IP, ports et protocoles).

Exemple de gestion de règles

La gestion dynamique des règles est tellement pénible que l'usage établi veut plutôt que l'on exploite un fichier de script comprenant toutes les règles, et qu'on le recharge complètement après modification.

```
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp dpt:23
2    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp dpt:53
3    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp dpt:80
alpha:~# iptables -D FORWARD 1
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp dpt:53
2    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp dpt:80
alpha:~# iptables -I FORWARD 1 -s 192.168.1.0/24 -p tcp --dport 22 -j ACCEPT
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp dpt:22
2    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp dpt:53
3    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp dpt:80
alpha:~#
```

d. Gestion des flux retours

Dans la plupart des applications réseau, un hôte envoie un paquet à destination d'un autre qui lui répond. On a donc une communication à double sens. Or, dans la configuration d'un pare-feu, on visualise bien les flux aller : par exemple, depuis un navigateur vers un serveur web sur le port 80, mais moins bien les réponses des serveurs qui se font sur un port aléatoire à l'initiative du client supérieur à 1024.

Dans les premiers âges des pare-feu, la solution consistait à autoriser tout trafic entrant dont le port était supérieur à 1024. Les pare-feu avaient alors davantage vocation à empêcher les gens de sortir plutôt que d'éviter les intrusions dans le réseau.

Depuis quelques années, les pare-feu dits « stateful » (à état) sont capables d'autoriser dynamiquement les flux retours du moment qu'ils sont la réponse à un flux en sortie explicitement autorisé.

Autorisation implicite des flux retours

```
iptables -A chaine -m state --state ESTABLISHED,RELATED -j ACCEPT
```

L'option -m state permet de réaliser un filtre en fonction de l'état du paquet traité. Les états acceptés : ESTABLISHED et RELATED représentent respectivement des paquets en réponse à un flux aller autorisé, et des paquets issus d'une nouvelle connexion, mais à l'initiative d'une connexion établie et autorisée (par exemple le trafic de données ftp relatif à un trafic de commandes ftp).

Exemple de configuration complète d'un pare-feu

On configure ici un pare-feu qui ne laisse rien passer, à l'exception des réponses aux trafics établis, ainsi que les protocoles nécessaires à la navigation Internet (http, https et dns).

```
alpha:~# iptables -P INPUT DROP
alpha:~# iptables -P OUTPUT DROP
alpha:~# iptables -P FORWARD DROP
alpha:~# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

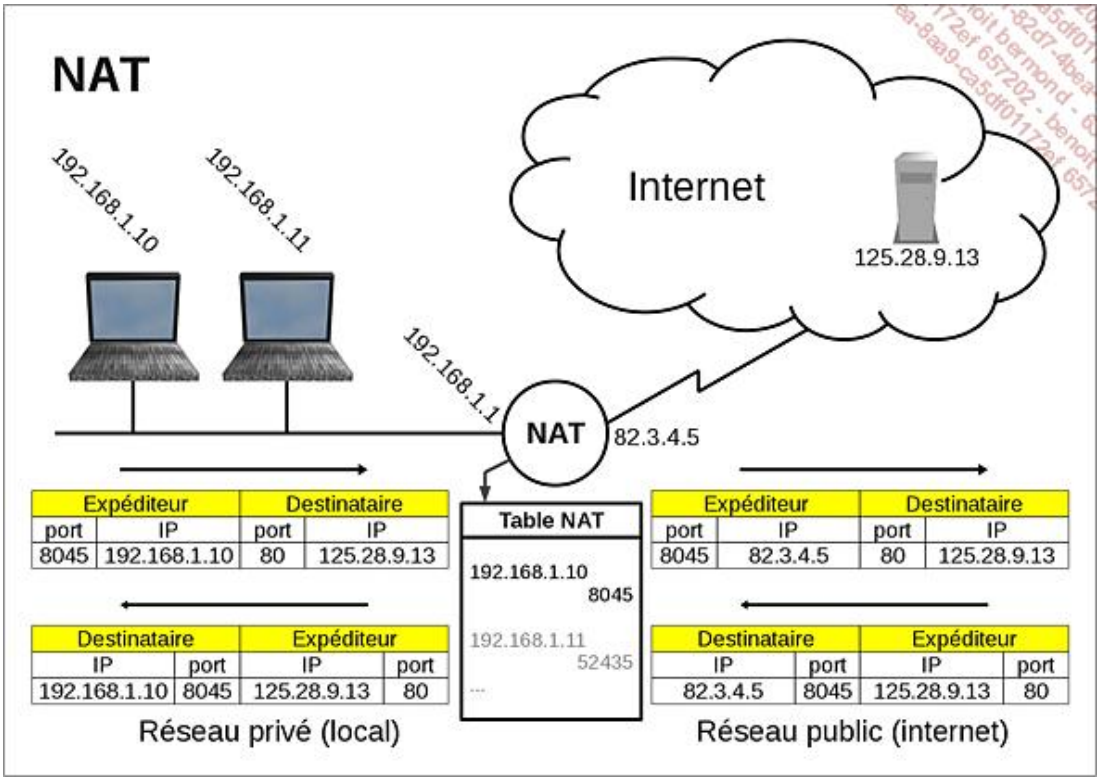
```
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp --dport 80 -j ACCEPT
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp --dport 443 -j ACCEPT
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p udp --dport 53 -j ACCEPT
```

Dans cet exemple, on configure un pare-feu qui ne laisse rien passer, à l'exception des réponses aux trafics établis, ainsi que les protocoles nécessaires à la navigation Internet (http, https et dns).

L'application **fail2ban** permet en cas de tentatives de connexion infructueuses à des applications ou au système lui-même de créer dynamiquement une règle qui bloquera toute communication de la part de l'attaquant.

3. Gestion du NAT

a. Rappel sur le principe du NAT



Le NAT consiste à réécrire l'en-tête IP d'un paquet qui passe d'un réseau public vers un réseau privé et inversement.

Les adresses IP publiques étant non routables sur l'Internet, un paquet qui proviendrait d'une adresse privée ne pourrait pas trouver de route retour, parce qu'aucun routeur n'accepterait de le renvoyer chez lui. De toute façon, les réseaux privés étant démultipliés à l'infini (il existe des millions de réseaux 192.168.1.0), il ne serait pas possible de maintenir dans les tables de routage des routeurs d'Internet une route cohérente vers le réseau d'origine.

La solution consiste donc pour sortir d'un réseau privé à remplacer l'adresse IP de l'expéditeur privé par l'adresse IP publique (unique sur Internet) du routeur réalisant le NAT. La traçabilité des translations (remplacement des adresses IP privées) se fait par rapport au port expéditeur utilisé : pour chaque translation réalisée, le routeur garde en mémoire le port expéditeur employé. Le paquet retour arrivant sur l'adresse publique du routeur et sur le port employé par l'expéditeur, l'adresse originelle du client est facilement retrouvée par le routeur NAT.

b. Diagnostic de la configuration NAT d'un routeur

Le NAT est géré dans une table spécifique appelée **NAT**. Toute configuration touchant au NAT se fera avec la commande **iptables** en précisant qu'on travaille sur la table NAT. Les chaînes traitées dans la table NAT sont **PREROUTING**, **POSTROUTING** et **OUTPUT**, représentant le trafic à modifier avant le routage, après, ou directement en sortie de la machine.

Affichages de la configuration NAT

```
iptables -t nat -L
iptables -t nat -S
```

c. Connexion d'un réseau privé à un réseau public

Dans cette configuration qui est aussi la plus courante, l'adresse IP d'expéditeur des hôtes du réseau privé est remplacée par l'adresse publique du routeur NAT.

Configuration du NAT

```
iptables -t nat -A POSTROUTING -o carte-ext -j action-nat
```

Nat avec iptables : options et paramètres	
-t nat	La règle concerne la table de NAT.
-A POSTROUTING	On ajoute une règle à la chaîne POSTROUTING, pour un traitement après routage.
-o *carte-ext*	Désigne la carte réseau par laquelle les paquets sortent du pare-feu.
-j *action-nat*	Désigne le mode d'action du NAT, supporte deux options : SNAT si l'adresse publique est fixe, et MASQUERADE si l'adresse publique est dynamique.

Exemple de configuration du NAT

```
alpha:~# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
alpha:~#
```

Dans cet exemple, eth1 est l'interface connectée au réseau public.

4. Scripts de configuration des règles de filtrage

a. Red Hat et les iptables

Les systèmes Red Hat et leurs dérivés proposent un service iptables qui permet d'appliquer une configuration de filtrage ou de NAT automatiquement. Le démarrage du service applique la configuration, et son arrêt annule tout filtrage. Ce fonctionnement est extrêmement pratique et permet de gérer un pare-feu Red-Hat de façon très confortable.

b. Création de service personnalisé de pare-feu avec les iptables

On constate assez vite que la création de règles de filtrage et de NAT avec les iptables a quelque chose de fastidieux. Par conséquent, après avoir déterminé les règles dont on a besoin, on aura tout intérêt à les placer dans un script.

Exemple de script de configuration de pare-feu

Ce type de script dispense d'avoir à gérer les règles une par une en cas de modification de la configuration. Il est beaucoup plus facile d'insérer une ligne dans le script que de décaler la numérotation des règles en mémoire. Toutefois, il faut annuler toute règle avant chaque application du script.

```
#!/bin/bash
# nom du fichier : /etc/parefeu_on
# Politique de base
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# NAT avec eth0 en interne et eth1 en sortie - adresse IP publique fixe
```

```
iptables -t nat -A POSTROUTING  -o eth1 -j SNAT --to-source 81.2.3.4
# gestion des paquets retours
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# trafic autorisé en sortie
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p udp --dport 53 -j ACCEPT
```

Bien entendu, il ne faudra pas oublier de le rendre exécutable.

Il sera également utile de créer un script d'annulation de toute règle de filtrage. Il peut en effet être utile d'autoriser plus ou moins provisoirement tout trafic, pour une mise à jour du pare-feu ou un usage applicatif ponctuel.

Exemple de script d'annulation de filtrage

```
#!/bin/bash
# nom du fichier : parefeu_off
# Effacement des règles
iptables -F
# Politique permissive
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

Enfin, on peut créer un script de gestion de service normalisé.

Exemple de script de service de pare-feu

Ce script est naturellement à placer dans le répertoire /etc/init.d.

```
#!/bin/bash
# nom du fichier : parefeu
case $1 in
start)
  /etc/parefeu_on
  ;;
stop)
  /etc/parefeu_off
  ;;
status)
  iptables -L
  ;;
*)
  echo "Syntaxe : /etc/init.d/parefeu start|stop|status
  ;;
esac
```