Linux Firewalling - IPTABLES

Aujourd'hui tout le monde sait ce que c'est qu'un firewall ainsi que son utilité sur un réseau, un serveur ou même un ordinateur personnel. En gros, c'est la partie du système qui s'occupe de gérer les accès à la machine et/ou au réseau et donc ainsi de les protéger des intrusions venant d'un autre réseau ou d'une autre machine.

Iptables est l'interface permettant de configurer NetFilter qui est un module du noyau linux fournissant la possibilité de contrôler, modifier, et filtrer les paquets IP qui passent par une interface réseau, et donc de faire du firewalling.

Ce document essaie d'expliquer les options de base de la commande iptables permettant la configuration du firewall d'une machine linux.

Nous supposons par la suite que nous avons iptables installé sur notre machine de test et que celle-ci dispose de deux interfaces réseaux : ppp0 qui est connectée à l'internet, et eth0 qui est connectée au réseau local.

[Bizarrement, je ne suis pas très inspiré aujourd'hui parce que ça fait quelques temps que je n'écris plus, j'espère que je ne vais pas raconter de bobards dans ce document qui traite d'un sujet qui me tient à cœur. J'utilise de plus en plus iptables, et la rédaction de ce document me permettra à moi-même de capitaliser le peu d'expérience que j'ai acquis en la matière. Bref, démarrons !]

I – Sauvegarde et Restauration avec iptables :

Lorsque vous arrêtez ou redémarrez votre programme iptables, les modifications que vous apportées à la configuration de votre firewall peuvent être perdues. Pour que ces modifications soit sauvegardées et donc réutilisées lors du prochain chargement du programme, il faut :
- éditer le fichier /etc/sysconfig/iptables-config
~]# *vi /etc/sysconfig/iptables-config*

```
# Save current firewall rules on stop.
#   Value: yes|no,  default: no
# Saves all firewall rules to /etc/sysconfig/iptables if firewall gets stopped
# (e.g. on system shutdown).
IPTABLES_SAVE_ON_STOP="no"

# Save current firewall rules on restart.
#   Value: yes|no,  default: no
# Saves all firewall rules to /etc/sysconfig/iptables if firewall gets
# restarted.
IPTABLES_SAVE_ON_RESTART="no"
```

- puis metre à "yes" ces deux options à "no".

Ainsi donc, les modifications seront sauvegardées dans le fichier /etc/sysconfig/iptables.

Deux programmes existent permettant de faire des backup et restauration de la configuration de iptables, ce sont *iptables-save* et *iptables-restore*.

Exemple de sauvegarde : (Sauvegarde dans /home/roger/iptables.save)
~]# *iptables-save > /home/roger/iptables.save*

Exemple de restauration : (Restauration à partir de /home/roger/iptables.save)
~]# *iptables-restore < /home/roger/iptables.save*

II – Les tables :

Il existe 4 types de tables indépendantes : **filter**, **nat**, **mangle**, **raw**.
Les règles du firewall sont définies dans ces tables. Un firewall n'utilise pas forcément toutes ces quatre tables.
L'option –t de la commande iptables permet de préciser la table sur laquelle on travaille, lorsque cette option n'est pas précisée, c'est la table par défaut (**filter**) qui est utilisée.

La table **filter** est utilisée pour faire des filtres (accepter, rejeter un paquet).
La table **nat** est utilisé pour faire du NAT (NAT destination et NAT de source), on dit qu'elle est consultée quand un paquet créant une nouvelle connexion arrive.
La table **mangle** est utilisée pour marquer le paquet, le modifier (notamment les bits de QOS) pour permettre de faire par exemple de la qualité de service. En anglais, ils disent : « This table is used for specialized packet alteration ».
La table **raw** qui est apparemment un peu plus sophistiquées et dont je ne parlerai pas dans cette première version de mon document.

Les tables contiennent donc des règles qui sont organisées dans des chaînes (chaînes de règles) :

III – Les chaînes :

Il existe 5 chaînes prédéfinies, mais l'administrateur peut également construire des chaînes personnelles. Chaque chaîne est une liste de règles qui peut correspondre à un ensemble de paquets :

INPUT : Concerne les entrées sur la machine locale (localhost), donc des connexions aux sockets de la machine elle-même. Elle ne peut pas être utilisée avec la table **nat**.
OUTPUT : Concerne les sorties de la machine locale, les paquets générés localement par le firewall lui-même en cours de configuration.
FORWARD : Concerne tous les paquets traversant la machine du/vers le réseau local en cours de routage à travers la machine. Cela suppose que notre machine a plusieurs interfaces et qu'elle sait acheminer des paquets entre différents réseaux. Son utilisation dans le **nat** n'aurait pas de sens.

PREROUTING : (Avant routage) Permet d'altérer les paquets au fur et à mesure qu'ils arrivent avant routage, donc de modifier la destination des paquets. Sert à faire du DNAT et est principalement utilisée dans la table **nat**.
POSTROUTING : Permet d'altérer les paquets qui sont routés vers l'extérieur, donc de faire du SNAT en modifiant la source des paquets. Souvent utilisée dans la table **nat**.

III – Sens des chaînes dans les tables :

3.1 Table filter :

3 chaînes sont possibles :
- INPUT : for local sockets, interdire ou accepter les paquets destinés à la machine elle-même.
- FORWARD : Permet de mettre des filtres sur les paquets qui viennent d'ailleurs et doivent traverser la machine. (Donc la machine est un routeur)
- OUTPUT : filtre sur les paquets générés par la machine elle-même.

3.2 Table nat :

3 chaînes possibles :
- PREROUTING : Pour faire du DNAT, redirection de port ou d'IP sur une machine locale. Par exemple, elle reçoit une requête venant de l'Internet sur le port 80 et la transmet à une machine du réseau local sur un autre port (8080).
- POSTROUTING : Pour faire du SNAT, modifier la source (port et/ou IP)
- OUTPUT : Permet de faire du NAT pour la machine elle-même en modifiant les paquets générés locallement.

3.3 Table mangle :

For specialised packet alteration : Possibilité d'utiliser toutes les chaînes prédéfinies.
[A completer plus tard]

IV – Ordre d'analyse des chaînes (dans le cas de paquets en transit) :

Etape	Table	Chain	Commentaire
1			Trames sur le réseau physique (ex. Wifi)
2			Arrivée sur l'interface (i.e., wlan0)
3	mangle	PREROUTING	Chaîne normalement utilisée pour altérer les paquets (Changer le TOS) This chain is normally used for mangling packets, i.e., changing TOS and so on.
4	nat	PREROUTING	Principalement utilisée pour faire du DNAT. Avoid filtering in this chain since it will be bypassed in certain cases.

Etape	Table	Chain	Commentaire
5			Décision de routage : où doit aller le paquet ?
6	mangle	FORWARD	Pour des besoins très spécifiques, modifiant le paquet après la première décision de routage, mais avant la dernière décision prise avant l'envoi du paquet.
7	filter	FORWARD	Seuls les paquets forwardés passent par cette chaîne et c'est à ce niveau qu'il faut écrire tous les filtres pour ce genre de paquets.
8	mangle	POSTROUTING	Altérer les certains types spécifiques de paquets après toutes décisions de routage, mais lorsque le paquet est toujours sur la machine.
9	nat	POSTROUTING	Utilisée pour essentiellement pour faire du SNAT et du MASQUERADE. Avoid doing filtering here, since certain packets might pass this chain without ever hitting it.
10			Sortir par l'interface choisie par le routage (ex. eth0 ou ppp0).
11			Mise des trames sur le support physique (ex LAN).

V – Targets/Jumps (cibles) :

Permet de dire ce que le firewall doit faire avec un paquet qui rempli les critères définis par la règle : accepter le paquet, rejeter le paquet, le détruire, ou bien faire un saut vers une autre chaîne, …
C'est l'option –t de la commande iptables qui permet de spécifier le target.

Il existe plusieurs types de cibles (targets) prédéfinis. Nous ne regarderons que celles qui nous intéressent :

ACCEPT : Accepter le paquet.
DROP : Détruire ni plus ni moins le paquet sans même envoyer de message d'erreur en retour. (Lorsque mon ennemi m'envoi des paquets par exemple, je ne gaspille pas ma bande passante pour lui dire que j'ai refusé son paquet).
REJECT : (Uniquement dans les chaînes FORWARD, INPUT, OUTPUT) Rejeter le paquet en envoyant un ICMP à la source du paquet. Plus souple et plus souvent utilisé que DROP.
Exemple : iptables –A FORWARD –p tcp --dport 23 –j REJECT --reject-with tcp-reset
MASQUERADE : Permet de faire du MASQUERADE donc du SNAT en nattant avec l'adresse IP de l'interface out precisée.
Exemple : iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE
iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE --to-ports 1025-2600

SNAT : Permet de faire du SNAT, plus sophistiqué que MASQUERADE, possibilité de préciser l'IP de sortie, ou même une plage d'IP ainsi que des ports. Peut préciser aussi le réseau IP à natter.
Exemple : iptables –t nat –A POSTROUTING -p UDP –o ppp0 –j SNAT --to-source 196.200.55.10-196.200.55.20:1024-32000
DNAT : Utilisé pour faire du DNAT, permet de modifier l'IP et/ou le port de destination, et donc de faire de la redirection de port. Très intéressant pour les réseaux privés qui doivent fournir des services Internet.
Exemple : iptables –t nat –A PREROUTING –p tcp –d 196.200.55.20 --dport 80 –j DNAT --to-destination 192.168.0.3:8080
LOG : Log detailed informations about packets. Donc logguer les informations sur les paquets. Il existe aussi ULOG que je n'ai pas du tout testé.
Exemple : iptables –R INPUT 1 –p icmp –j LOG --log-level info

Configurer ensuite votre syslog en ajoutant une ligne semblable :

kern.info /var/log/icmp.log

Vous trouverez ainsi vos log icmp dans /var/log/icmp.log

REDIRECT : Utilisé uniquement avec OUTPUT ou PREROUTING pour redirigé à la machine elle-même. Par exemple rediriger les requêtes http vers squid sur la même machine (Faire du proxy transparent).
Exemple : iptables –t nat –A PREROUTING --dport 80 –j REDIRECT --to-ports 8080
RETURN : Utilisé comme le RETURN du langage C qui permet de sortir d'une fonction, sauf qu'ici, ça permet de sortir de la chaîne comme si rien ne c'était passé et remonter peut être à la chaîne parente. Donc évite de tester les règles qui sont en dessous dans la même chaîne.
QUEUE : Passer le paquet à l'espace utilisateur, donc à un programme qui sait comment le traiter.

VI – Les options de la commande ipatbles :

La commande iptables permet d'écrire les règles de fonctionnement du firewall, et donc de le configurer.
En voici quelques options :
-A : ajoute chain rule-specification
-D : supprime chain rule-spec ou bien supprime chain rule-position
(Ex : iptables –D –t nat –A FORWARD 1)
-I : insère chain rule-position rule-spec
-R : remplace chain rule-position rule-spec
Exemple : iptables –R –A INPUT 1 –i lo –j DROP
-L : list [chain]
-F : flush [chain]
-N : Créer une nouvelle chain-utilisateur
-X : delete-chain [chain]
-p : permet de spécifier le protocole (TCP, ICMP, UDP, …)
-s : adresse IP source
-d : destination address
-j : jump to a target
-i : --in-interface (interface d'entrée)
-o : --out-interface

--dport : port destination
--sport : port source
--to-destination : spécifier une destination dans ce format : IP[:Port] (ex : 41.203.10.1:20)
--to-ports : vers ports spécifiés
--comment : Commentaire
-m : --match options, followed by the matching module name. L'option m est très sophistiquée et permet de faire beaucoup de choses. Ca permet de préciser un module, suivi des options du module.
Exemples :
a*) iptables –A FORWARD –i ppp0 –o eth0 –m state --state RELATED,ESTABLISHED -j ACCEPT*
Accepte de forwarder les paquets venant de l'interface ppp0 vers l'interface eth0 pour des connexions déjà établies ou relatives à d'autres connexions.
b) *iptables –A FORWARD –i eth0 –o ppp0 –m state --state NEW,RELATED,ESTABLISHED –j ACCEPT*
Accepte de forwarder les paquets venant de l'interface eth0 vers l'interface ppp0 pour des connexions nouvelles, déjà établies ou relatives à d'autres connexions.
c) *iptables –A INPUT –i lo –m comment --comment "je n'aime pas lo" –j DROP*
d) *iptables –A INPUT –i ppp0 –m addrtype --src-type ! UNICAST –j REJECT*
On rejette tout ce qui n'est pas unicast (not unicast)
e) *iptables –A FORWARD –i eth0 –o ppp0 –m conntrack --ctstate NEW,RELATED,ESTABLISHED ! --ctproto ICMP ! --ctorigsrc 127.0.0.1 –j ACCEPT*

Bonus : Activer du NAT sur un linux

A/ Dire au noyau qu'on veut que la machine forward les paquets :

~]# echo 1 /proc/sys/net/ipv4/ip_forward

Egalement éditer le fichier /etc/sysctl.conf, puis modifier la ligne où il y'a ip_forward :

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 0
```

Mettre le '0' à 1 ➔ net.ipv4.ip_forward = 1

B/ Activer le NAT avec iptables :

~]# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

C/ C'est tout. Démarrez votre iptables et le compte est bon. Si vous n'arrivez pas à sortir de votre réseau, c'est que vous avez des règles notamment dans la chaîne FORWARD qui vous bloquent. Il faut soit les supprimer, soit insérer des règles avant qui vous autorise à sortir.
Ex : *iptables -I FORWARD 1 -i ppp0 -o eth0 -m state -state RELATED,ESTABLISHED -j ACCEPT*

Sources :

http://www.netfilter.org/
http://iptables-tutorial.frozentux.net/iptables-tutorial.html
http://www.faqs.org/docs/iptables/
http://christian.caleca.free.fr/netfilter/iptables.htm
http://www.howtoforge.com/nat_iptables
http://support.imagestream.com/iptables_Firewall.html

Roger YERBANGA
www.rogeryerbanga.fr.st