TABLE DES MATIERES

. : Bibliographie : .

Hors-Série Linux Magazine MISC n° 8 et 12 : http://www.miscmag.com/

Man Page Iptables (traduction française Christophe Donnier) :

http://www.delafond.org/traducmanfr/man/man8/iptables.8.html

Tutorial Iptables (Oskar Andreasson) : http://www.netfilter.org/documentation/

La couche *firewalling*

Un des apports majeurs du noyau Linux version 2.4 est sans contexte sa nouvelle couche de *firewalling*.

Les nouvelles fonctionnalités marquantes sont :
- architecture modulaire
- filtrage à états
- gestion complète de NAT (*Network Address Translation*)
- altération possible des en-têtes de paquets (*packet mangling*)

Netfilter
Cette couche, disponible dans les versions 2.4 du noyau Linux, regroupe l'ensemble des fonctions internes du noyau qui réalisent les opérations de *firewalling*. Netfilter possède une architecture modulaire.

Iptables
Netfilter est lié à l'utilitaire iptables qui, travaillant dans l'espace utilisateur, sert d'interface de configuration.

Les tables IP (*IP tables*)

Netfilter prend en charge un paquet arrivant et le passe à son système d'évaluation de règles que sont les tables IP.

Une table est constituée d'un nombre arbitraire et non limité de chaînes.

Une chaîne est une suite linéaire de règles.

Une règle est constituée d'un motif (*pattern*) destiné à reconnaître des paquets selon un nombre indéterminé de critères (*matches*) et d'une décision, appelée cible (*target*), à prendre en cas de reconnaissance du paquet.

Les tables IP sont gérées par un cœur (*core*) responsable de :
- l'enregistrement et le désenregistrement des tables
- l'enregistrement et le désenregistrement des critères de reconnaissance et des cibles auprès d'une table
- l'interface entre l'espace utilisateur et l'espace noyau pour la manipulation des règles dans une table
- l'évaluation des paquets lors de la traversée des tables

Il existe à ce jour trois tables : FILTER, NAT et MANGLE.

La table FILTER

C'est la table qui permet les opérations de filtrage IP. Les paquets y sont acceptés (**ACCEPT**), refusés (**DROP** ou **REJECT** avec renvoi d'un paquet erreur), logués (**LOG**) ou encore mis en queue (**QUEUE**), mais jamais modifiés.

Cette table contient trois chaînes de base :

- INPUT : les paquets à destination de la machine
- FORWARD : les paquets traversant (routés par) la machine
- OUTPUT : les paquets émis par la machine

Tout paquet traité par le système traversera **une et une seule** de ces trois chaînes.

La table NAT (*Network Address Translation*)

C'est la table qui permet les opérations de traduction d'adresses.
Cette table contient trois chaînes de base :

- PREROUTING
- OUTPUT
- POSTROUTING

Cette table dispose de cibles propres (SNAT, DNAT, MASQUERADE et REDIRECT) pour la mise en œuvre de NAT.

La table MANGLE

C'est la table qui permet d'altérer les en-têtes des paquets.
Cette table possède les cinq chaînes de base :

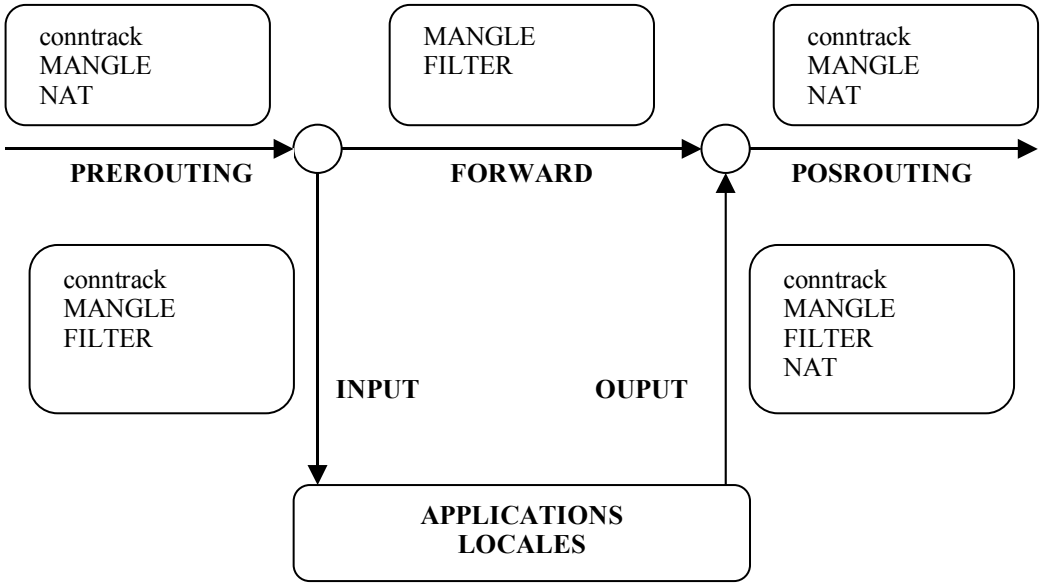
- PREROUTING
- INPUT
- FORWARD
- OUTPUT
- POSTROUTING

La table MANGLE modifie la structure qui représente le paquet dans la couche réseau du noyau. Il est donc possible d'agir sur les en-têtes mais aussi sur les champs des structures utilisées par le système.

Passage de paquets dans les tables

Netfilter apporte aux couches réseaux de niveau 3 une série de points d'entrée (*hooks*) et de retours (*callbacks*). Le code de filtrage est donc entièrement déporté en dehors de la couche réseau.

Lorsqu'un paquet traverse la couche réseau, il vient rencontrer les *hooks* de Netfilter. Pour chaque *hook*, le paquet est évalué dans la chaîne associée (si existante) pour toutes les tables successivement. Il y a donc un ordre (une priorité) dans lequel sont évaluées les tables.



Par exemple, lorsqu'un paquet arrive, il va d'abord être évalué dans la chaîne PREROUTING de la table MANGLE, puis dans la chaîne PREROUTING de la table NAT (la table FILTER ne possédant pas cette chaîne, on n'y entre pas).

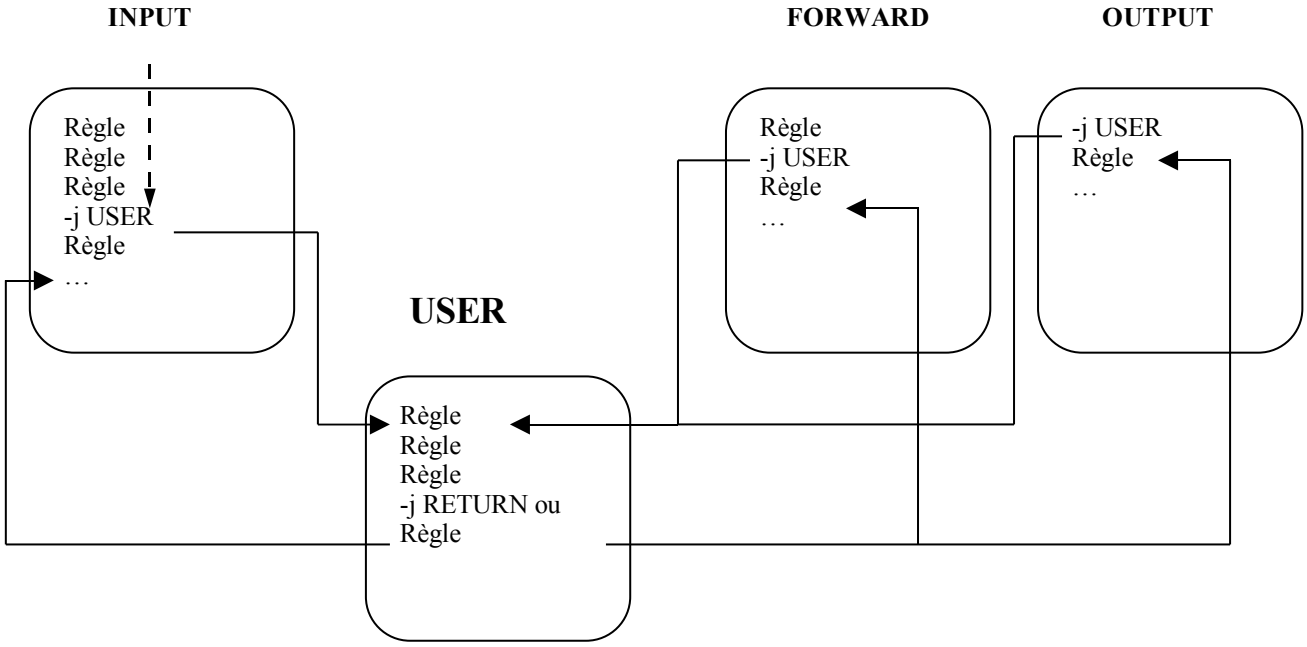
Les trois tables possèdent la chaîne OUTPUT, l'ordre sera d'abord MANGLE, puis FILTER et enfin NAT.

Les chaînes

Les chaînes sont de deux types :

- les chaînes de base (les chaînes par défaut de la table)
- les chaînes utilisateur (crées pour les besoins spécifiques de l'administrateur)

Lorsqu'un paquet entre dans une table, il est envoyé à la chaîne de base associée au traitement en cours. Toutes les règles de cette table sont alors évaluées séquentiellement par comparaison du paquet avec le motif de reconnaissance défini. Si le paquet correspond au motif d'une règle, la décision associée est prise.



Une chaîne utilisateur est une chaîne crée sur demande. On y envoie les paquets en spécifiant son nom comme cible : on dit que la chaîne courante appelle la chaîne utilisateur. Un paquet la traverse comme une chaîne de base : une règle après l'autre. Si le paquet arrive en fin de chaîne ou atteint la cible RETURN, il est renvoyé à la chaîne appelante.

Les chaînes possèdent des compteurs de paquets. On peut les consulter pour voir le nombre de paquets ainsi que la quantité de données traitée par chaque règle d'une chaîne et sa politique.

Les chaînes de base possèdent une politique par défaut (*policy*). Lorsqu'un paquet arrive en bout de chaîne, il est donc possible de lui appliquer une action par défaut (cibles ACCEPT ou DROP seulement).

Les opérations possibles sur le chaînes :

-N CHAINE : crée la chaîne utilisateur CHAINE
-L CHAINE : liste les règles de la chaîne CHAINE (on peut ajouter l'option –v)
-F CHAINE : vide la chaîne CHAINE de ses règles
-Z CHAINE : remet à zéro les compteurs de la chaîne CHAINE
-E OLD_CHAINE NEW_CHAINE : renomme la chaîne OLD_CHAINE en NEW_CHAINE
-P CHAINE : fixe la politique par défaut de la chaîne (limitée aux chaînes de base)
-X CHAINE : supprime la chaîne utilisateur CHAINE

Le motif de reconnaissance

Pour évaluer un paquet, on le compare à un motif de reconnaissance contenu dans chaque règle. Ce motif est composé d'un nombre variable de critères selon la finesse désirée.

Les critères de base porte sur IP (adresse source et destination) et le protocole de niveau 4.

Les autres critères sont apportés par des modules appelés concordance (*matches*). L'utilisation d'une concordance se fait par l'option –m.

Chaque concordance possède ses options propres et sont donc utilisables pour affiner le filtrage.

Il existe de nombreuses concordances. En voici quelques unes :

state : concordance sur l'état des paquets
mac : concordance sur l'adresse MAC (source)
owner : concordance sur les paramètres du processus qui émet le paquet
time : concordance sur la plage horaire de réception ou d'envoi de paquets
ttl : concordance sur la velur du champ TTL d'un paquet IP
etc …

Pour les autres options, il faut consulter la page **man** d'iptables.

Les concordances sont valables pour toutes les tables, mais pas forcément dans toutes les chaînes.

Les cibles

Lorsqu'un paquet correspond au motif de reconnaissance d'une règle, une décision est prise.

Chaque règle ne peut avoir qu'une seule cible, mais éventuellement aucune pour réaliser des comptages de paquets par exemple.

Les cibles peuvent être propres à une table donnée et utilisable uniquement dans certaines chaînes de cette table : **ACCEPT**, DNAT, **DROP**, **LOG**, MARK, MASQUERADE, MIRROR, QUEUE, REDIRECT, **REJECT**, RETURN, SNAT, TOS, TTL et ULOG.

Construction d'une règle

Une règle se construit en quatre parties :

- une table d'application de la chaîne (FILTER par défaut)
- une chaîne d'application (une chaîne de base ou utilisateur)
- un motif de reconnaissance
- une cible représentant la décision à prendre

TABLE	CHAINE	MOTIF DE RECONNAISSANCE	CIBLE

iptables | -t FILTER | –A INPUT | -m state –state ESTABLISHED,RELATED | -j ACCEPT |

Les opérations permises par iptables pour placer une règle dans une chaîne sont :

-A CHAINE [REGLE] : place la règle REGLE en fin de chaîne CHAINE
-I CHAINE n [REGLE] : place la règle REGLE au rang n de la chaîne CHAINE (par défaut en tête de chaîne)
-R CHAINE n [REGLE] : remplace la règle de rang n par la règle REGLE dans la chaîne CHAINE
-D CHAINE n : supprime la règle de rand n de la chaîne CHAINE

Le suivi de session

Netfilter permet le filtrage à états de niveau 5 couramment appelé suivi de session. Chaque paquet traité par la machine sera affecté d'un état. Par contre, Netfilter ne prendra aucune décision en fonction de l'état d'un paquet : cette décision appartient à l'utilisateur en le spécifiant explicitement par la construction de ses règles.

Les états d'un paquet

L'utilisateur construit des règles qui se servent de l'état d'un paquet comme critère de filtrage. Pour cela, on utilise la concordance *state*.

L'état d'un paquet peut avoir quatre valeurs qui dépendant de l'état de la session à laquelle il appartient :

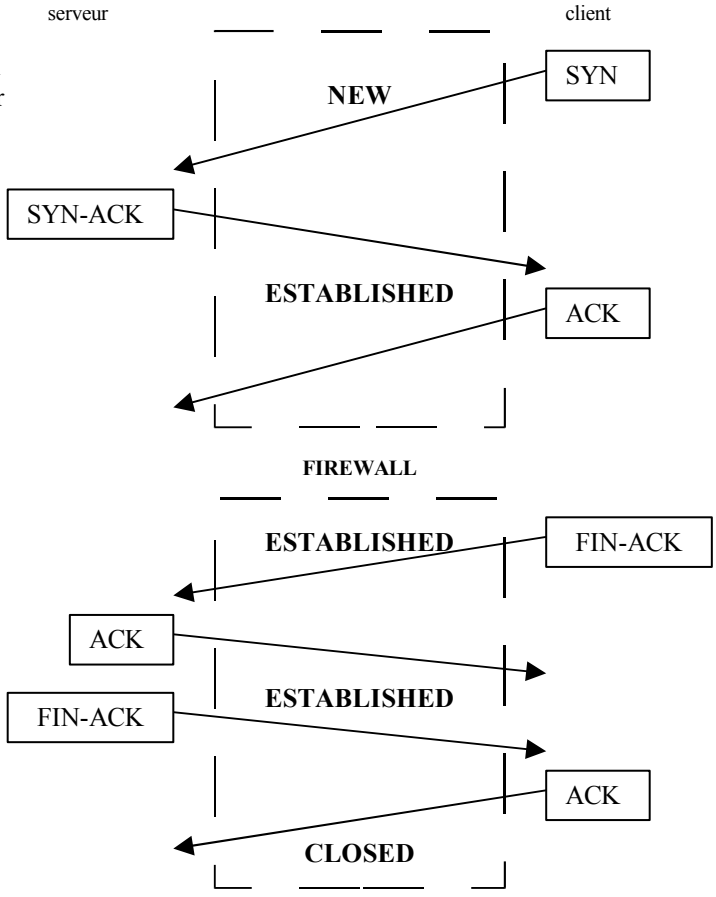
- NEW : le paquet ne correspond à aucun flux connu et s'il est accepté, une nouvelle session sera crée
- ESTABLISHED : le paquet fait partie d'une session existante
- RELATED : paquet attendu faisant partie d'une session en cours
- INVALID : paquet dont l'état n'a pu être déterminé et devra donc être détruit

Le suivi de protocoles

TCP

Le suivi des connexions TCP est réalisé en fonction de l'état des drapeaux (URG, ACK, PSH, RST, SYN et FIN).
Les états internes, définis par la RFC 793, sont visibles dans la tables des états (`/proc/net/ip_conntrack`). Les états visibles au niveau utilisateur pour les paquets seront **NEW** pour le premier et **ESTABLISHED** pour les suivants.

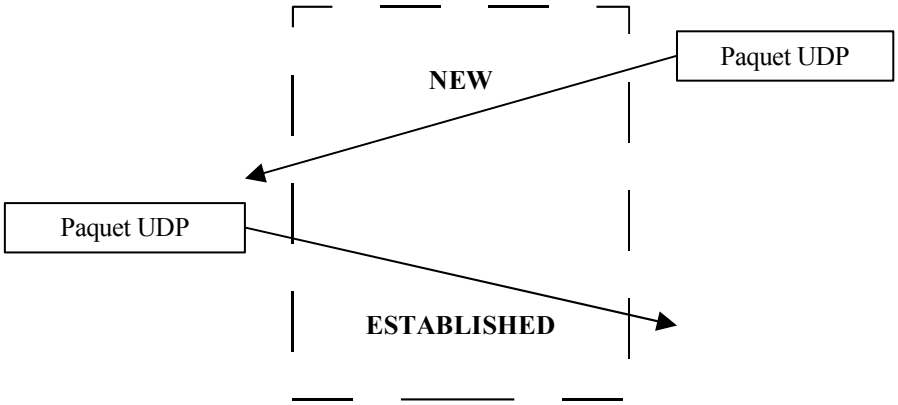
Remarques :
Tout paquet ayant un jeu de drapeaux valide au sens de la RFC sera accepté comme valide et pouvant donc recevoir un état.
Un paquet sans drapeau sera INVALID puisqu'au moins un drapeau doit être positionné.



UDP

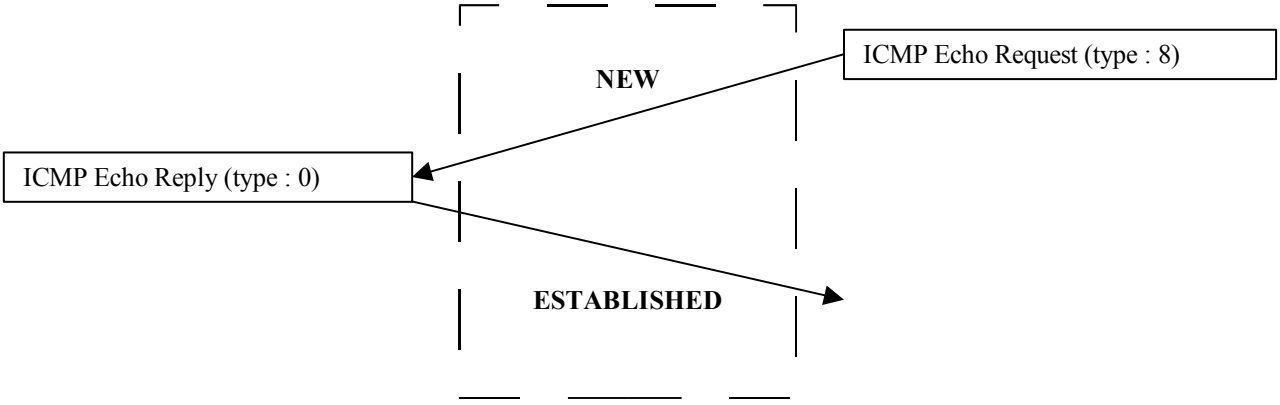
Le protocole UDP étant un protocole sans connexion, il n'est donc suivi qu'au niveau des *timeout*. Ainsi, le premier paquet crée la session (NEW) et les suivant auront l'état ESTABLISHED. Au bout d'un certain d'inactivité, la session est détruite.

ICMP



Le protocole ICMP fait apparaître deux type de paquets :

- les **requêtes** ICMP : elles sont au nombre de quatre (Echo [0-8], Timestamp [13-14], Address mask [17-18] et Information [15-16])



On peut donc appliquer les règles suivantes par exemple :

```
iptables -A INPUT -p ICMP -m state --icmp-type 8      -j ACCEPT          ou
iptables -A INPUT -p ICMP -m state --icmp-type echo-request -j DROP
```

Remarques :
Les ICMP ont un *timeout* de 30 s, le *timer* étant remis à zéro par chaque réponse. Une réponse non sollicitée est considérée comme INVALID.

- les **erreurs** ICMP qui sont associées à des sessions existantes. Ce type de paquet a l'état RELATED ou INVALID. On peut donc appliquer les règles suivantes par exemple :

```
iptables -A INPUT -p ICMP -m state --state RELATED    -j ACCEPT
iptables -A INPUT -p ICMP -m state --state INVALID    -j DROP
```

Le suivi applicatif

Certaines applications (FTP par exemple) font intervenir plusieurs flux au sein d'une même session. Le nouveau flux sera mis à l'état RELATED puisqu'il est associé à une session existante.

La traduction d'adresse réseau

La table supporte deux types d'action :

- la modification d'adresse source (**S**NAT)
- la modification d'adresse destination (**D**NAT)

Modification d'adresse source

Cette modification est réalisée juste avant l'envoi du paquet (POSTROUTING). En effet, il est plus judicieux de ne pas perdre la véritable adresse IP source avant la fin de tous les traitements.

Deux cibles permettent de réaliser cette modification :

- SNAT : la cible SNAT pour la spécification de l'adresse source

- MASQUERADE : cette cible est associée explicitement à l'interface de sortie (MASQUERADE n'est qu'un cas particulier de SNAT). Si cette interface tombe en panne ou change d'adresse IP (cas d'adresse IP dynamique), toutes les sessions en cours sont détruites (ce qui n'est pas le cas avec la cible SNAT qu'il faudra donc privilégiée dans la majorité des cas). La cible MASQUERADE est souvent utilisée dans le cas d'adresse dynamique (DHCP) et notamment pour partager une connexion internet.

Modification d'adresse destination

Cette modification se fait au niveau des chaînes PREROUTING et OUTPUT, c'est-à-dire avant un point de routage interne à Netfilter (*Routing Decision*) puisque la modification de l'adresse peut modifier la manière dont sera routé le paquet.

Deux cibles permettent de réaliser cette modification :

- DNAT : pour la spécification d'une adresse destination

- REDIRECT : qui envoie les paquets vers *localhost*, ce qui est équivalent à un DNAT sur 127.0.0.1.

Remarque :
La cible REDIRECT set en particulier à réaliser des proxy transparents, comme par exemple la redirection du trafic http vers un proxy local (squid par exemple) :

```
iptables -t NAT -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 3128
```

Autres fonctionnalités

La journalisation des paquets

Netfilter offre la possibilité de journaliser des paquets en utilisant les cibles LOG ou ULOG.
Le fonctionnement de ces cibles est particulier dans le fait qu'elles n'agissent pas sur le cheminement du paquet dans la chaîne : il n'est donc pas filtrer mais simplement journaliser.

La cible LOG

La cible est responsable de la remontée des messages vers le démon syslogd avec comme critère par défaut *warning* (modifiable à partir de l'option --log-level).

On utilise très souvent la concordance limit pour limiter le nombres de paquets logués. Par contre limit n'est pas indiquée pour faire de la gestion de bande passante puisque son fonctionnement ne répond pas à cette problématique (cf. l'outil tc *Traffic Control*).

Pour distinguer les différents messages dans un fichier de log, on utilisera l'option --log-prefix pour placer un préfixe dans le message logué.

Exemple :
Pour loguer toutes les demandes de connexion TCP en limitant l'enregistrement à 3 par seconde, on obtient :

iptables –A INPUT –m state --state NEW –m limit - -limit 3/second –l LOG --log-prefix "Nouvelle demande de connexion TCP :"

La cible ULOG

Cette cible permet de remonter les paquets logués vers un démon spécialisé (en espace utilisateur) qui serait à l'écoute. Celui ci se décomposera en deux parties :

- à l'entrée, triage des paquets remontés par la cible
- à la sortie, export des informations vers différentes resources (fichiers texte, base de données, etc …).

La mise en queue des paquets

La cible QUEUE permet d'envoyer des paquets en espace utilisateur. Le paquet sera alors mis en attente avant qu'il soit traité puis retourné par le démon.

Le marquage de paquets

Netfilter autorise le marquage interne d'un paquet (sans modification du paquet lui-même) en utilisant la cible MARK. Cela permettra de tracer (en utilisant la concordance mark) les paquets à travers la couche réseau et aussi de router ou gérer une qualité de service (QoS) en fonction des paramètres marqués.

Sauvegarder et restaurer les règles

Iptables fournit deux utilitaires :

- iptables-save pour sauvegarder les règles courantes dans un fichier
- iptables-restore pour charger les règles en mémoire

Exemple :

② Sauvegarde

La syntaxe de la commande est la suivante : **`iptables-save`** `[-c] [-t table]`

L'option `-c` permet de récupérer les valeurs courantes des compteurs.

```
iptables-save > /etc/ iptables-save
```

③ Restauration

La syntaxe de la commande est la suivante : **`iptables-restore`** `[-c] [-n]`

L'option `-n` permet de ne pas écraser les règles existantes.

```
iptables-restore < /etc/iptables-save
```

OU

```
iptables-save > /etc/iptables-save | iptables-restore
```

Remarques

La procédure d'installation de Netfilter/Iptables et des exemples de scripts sont fournies dans le tutoriel d'Oskar Andreasson.

Pour les utilisateurs de la distribution Mandrake, il est fort possible qu'ipchains soit chargé par défaut. Pour pouvoir utiliser iptables, il vous faut tout d'abord retirer ipchains, puis charger iptables :

```
modprobe -r ipchains
modprobe ip_tables
```

Options additionnelles

Il existe de nombreuses options du noyau à configurer pour un comportement optimal du pare-feu.
En voici quelques unes :

```
##########################################
## Configuration additionnelle du Kernel ##
##########################################
##
## linux/Documentation/filesystems/proc.txt
## linux/Documentation/networking/ip-sysctl.txt
##
##
## man ip
## man icmp
##
## il existe d'autres options comme :
## ip_always_defrag=1 : défragmente automatiquement les paquets
##

## Ignorer les messages ICMP buggés (RFC 1122)
if [ -e /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses ]; then
        echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
        echo "Ignorer les messages ICMP buggés          [OK]"
else
        echo "Ignorer les messages ICMP buggés          [FAILED]"
fi

## Activation de l'IP Forwarding.
if [ -e /proc/sys/net/ipv4/ip_forward ]; then
       echo "1" > /proc/sys/net/ipv4/ip_forward
       echo "Activation de l'IP Forwarding             [OK]"
else
       echo "Warning: /proc/sys/net/ipv4/ip_forward n'exsite pas"
       echo "Activation de l'IP Forwarding             [FAILED]"
fi

## Activation pour l'allocation d'adresse dynamique (DHCP) avec
/proc/sys/net/ipv4/
#if [ -e /proc/sys/net/ipv4/ip_dynaddr ]; then
#       echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#       echo "Activation de l'IP Dynaddr                [OK]"
#else
#       echo "Warning: /proc/sys/net/ipv4/ip_dynaddr n'exsite pas"
#       echo "Activation de l'IP Dynaddr                [FAILED]"
#fi

## Ne pas accepter les redirections ICMP.
## Désactivation sur toutes les interfaces.
#if [ -e /proc/sys/net/ipv4/conf/all/accept_redirects ]; then
#       echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
#fi
## Désactivation seulement sur l'interface externe.
if [ -e /proc/sys/net/ipv4/conf/$EXT_IF/accept_redirects ]; then
        echo "0" > /proc/sys/net/ipv4/conf/$EXT_IF/accept_redirects
        echo "Désactivation des redirections ICMP               [OK]"
else
        echo "Désactivation des redirections ICMP               [FAILED]"
fi
```

```
## Log des packets avec des adresses impossibles dans le systèmes de log.
if [ -e /proc/sys/net/ipv4/conf/all/log_martians ]; then
        echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
        echo "Log des adresses impossibles                    [OK]"
else
        echo "Log des adresses impossibles                    [FAILED]"
fi
## Ignore les requêtes ICMP sur toutes les interfaces
#if [ -e /proc/sys/net/ipv4/icmp_echo_ignore_all ]; then
#       echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all
#       echo "Ignore les requêtes ICMP          [OK]"
#else
#       echo "Ignore les requêtes ICMP          [FAILED]"
#fi
## Ignore les requêtes broadcast d'écho ICMP.
if [ -e /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts ]; then
        echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
        echo "Ignore les requêtes broadcast d'écho ICMP [OK]"
else
        echo "Ignore les requêtes broadcast d'écho ICMP [FAILED]"
fi
## Activation des rp_filter contre le spoofing.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
        for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
        echo "1" > $i; done
        echo "Activation des filtres contre le spoofing [OK]"
else
        echo "Activation des filtres contre le spoofing [FAILED]"
fi
## Désactivation des packets de source de routage.
if [ -e /proc/sys/net/ipv4/conf/all/accept_source_route ]; then
        for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
        echo "0" > $i; done
        echo "Désactivation des sources de routage              [OK]"
else
        echo "Désactivation des sources de routage              [FAILED]"
fi
## Désactivation du support de notification de congestion TCP.
#if [ -e /proc/sys/net/ipv4/tcp_ecn ]; then
#       echo "0" > /proc/sys/net/ipv4/tcp_ecn
#       echo "Désactivation de notif. de congestion TCP         [OK]"
#else
#       echo "Désactivation de notif. de congestion TCP         [FAILED]"
#fi
## Echelle locale pour les connexions TCP/UDP. (Machine à >128M pour connexions
>2000 simultanées)
#if [ -e /proc/sys/net/ipv4/ip_local_port_range ]; then
#       echo -e "32768\t61000" > /proc/sys/net/ipv4/ip_local_port_range
#fi
## Réglage du nombre maximum de connexion à tracquer. (Kernel Default: 2048)
if [ -e /proc/sys/net/ipv4/ip_conntrack_max ]; then
       echo "4096" > /proc/sys/net/ipv4/ip_conntrack_max
fi
## Proctection contre le "SYN Flooding"
if [ -e /proc/sys/net/ipv4/tcp_syncookies ]; then
        echo "1" >/proc/sys/net/ipv4/tcp_syncookies
        echo "Proctection contre le SYN Flooding                [OK]"
else
        echo "Proctection contre le SYN Flooding                [FAILED]"
fi
```

Petits tests entre amis

Quelque soit votre configuration, vous pouvez réaliser ces premiers tests pour appréhender la configuration d'iptables.
En effet, on se limitera à quelques tests sur l'interface *loopback* (**lo**).

La procédure d'installation de Netfilter/Iptables et des exemples de scripts sont fournies dans le tutoriel d'Oskar Andreasson : http://www.netfilter.org/documentation/

Pour les utilisateurs de la distribution **Mandrake**, il est fort possible qu'ipchains soit chargé par défaut. Pour pouvoir utiliser iptables, il vous faut tout d'abord retirer ipchains, puis charger iptables :

```
modprobe -r ipchains
modprobe ip_tables
```

① Vous pouvez vérifier les modules chargés en tapant **`lsmod`** et d'autre part la configuration par défaut d'ipatbles au démarrage en tapant **`iptables -L -v`** (cela permet de voir les politiques par défaut).

① On interdit tout ce qui vient de l'interface loopback

```
iptables -A INPUT -i lo -j DROP
ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) from 192.168.1.2 : 56(84) bytes of data.
--- 192.168.1.2 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
iptables -L -v
Chain INPUT (policy ACCEPT 18 packets, 2346 bytes)
pkts bytes target     prot opt in     out     source               destination
   7   588 DROP       all  --  lo     any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 25 packets, 2934 bytes)
pkts bytes target     prot opt in     out     source               destination

iptables -D INPUT 1
```

Remarque importante : cette règle a même interdit l'accès login à la machine !

② On interdit tout ce qui sort de l'interface loopback

```
iptables -A OUTPUT -o lo -j DROP
ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
ping: sendto: Operation not permitted
ping: sendto: Operation not permitted
ping: sendto: Operation not permitted

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss

iptables -L -v
Chain INPUT (policy ACCEPT 32 packets, 4328 bytes)
pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 53 packets, 6092 bytes)
pkts bytes target     prot opt in     out     source               destination
   4   336 DROP       all  --  any    lo      anywhere             anywhere

iptables -D OUTPUT 1
```

③ On interdit tout ce qui rentre du réseau

```
iptables -A INPUT -s 0/0 -j DROP
iptables -L -v
Chain INPUT (policy ACCEPT 54 packets, 7266 bytes)
pkts bytes target     prot opt in     out     source               destination
   0     0 DROP       all  --  any    any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 78 packets, 9282 bytes)
pkts bytes target     prot opt in     out     source               destination

ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss

iptables -D INPUT 1
```

④ On interdit tout ce qui rentre ou qui sort du réseau 192.168.1.0

```
iptables -A INPUT -s 192.168.1.0/24 -j DROP
ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) from 192.168.1.2 : 56(84) bytes of data.
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=42 usec
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=32 usec
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
iptables -D INPUT 1

iptables -A OUTPUT -d 192.168.1.0/24 -j DROP
ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) from 192.168.1.2 : 56(84) bytes of data.
ping: sendto: Operation not permitted
ping: sendto: Operation not permitted
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
iptables -D OUTPUT 1
```

⑤ On interdit toutes les requêtes echo (ping)

```
iptables -A INPUT -p icmp --icmp-type 8 -j DROP
ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
iptables -L -v
Chain INPUT (policy ACCEPT 47 packets, 6446 bytes)
pkts bytes target prot opt in  out source   destination
  3  252   DROP   icmp --  any any anywhere anywhere     icmp echo-request
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in  out source   destination
Chain OUTPUT (policy ACCEPT 71 packets, 8462 bytes)
pkts bytes target prot opt in  out source   destination
iptables -D INPUT 1
```

⑥ On interdit toutes les réponses echo (ping)

```
iptables -A INPUT -p icmp --icmp-type 0 -j DROP
ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) from 192.168.1.2 : 56(84) bytes of data.
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
iptables -L -v
Chain INPUT (policy ACCEPT 77 packets, 9186 bytes)
pkts bytes target prot opt in  out source   destination
  4   336  DROP   icmp --  any any anywhere anywhere           icmp echo-reply
```