Netfilter Extensions HOWTO

Fabrice MARIE <fabrice@netfilter.org>, liste de diffusion : `netfilter-devel@lists.samba.org` $Revision: 550 $, $Date: 2003-06-21 13:53:56 +0200 (Sat, 21 Jun 2003) $

Ce document décrit comment installer et utiliser les extensions de netfilter.

Table des matières

1 Introduction

Bonjour, c'est une excellente occasion pour moi pour remercier tous ceux qui passent beaucoup de temps à développer, tester, communiquer les bugs, et à utiliser netfilter. Donc merci à tous !

Ce HOWTO assume que vous avez lu et compris le

Linux 2.4 Packet Filtering HOWTO `http://www.netfilter.org/documentation/HOWTO/fr/packet-filtering-HOWTO.html` écrit par Rusty. Bien entendu, vous êtes sensé savoir compiler et installer un noyau correctement.

L'archive contenant `iptables` contient aussi les extensions de netfilter qui ne sont pas utilisées par les utilisateurs normaux, ou qui sont expérimentales ou finalement qui attendent d'être incluses dans le noyau. Ces extensions ne sont d'habitude pas compilées, sauf si vous le demandez.

Vous devriez trouver la dernière version de ce document sur la page web de

documentation de netfilter `http://www.netfilter.org/documentation/index.html#HOWTO`

Le but de ce HOWTO est d'aider les gens à commencer à utiliser les extensions de netfilter, en expliquant comment installer ces extensions et comment les utiliser.

Finalement, il y a une liste complète des patches inclus dans patch-o-matic (générée automatiquement par un script) :

Patch-O-Matic Listing - Summary `http://www.netfilter.org/documentation/pomlist/pom-summary.html`
.

2 Patch-O-Matic

2.1 Qu'est-ce que Patch-O-Matic ?

Les développeurs de Netfilter distribuent un jeu de patches qu'ils arrangent de telle manière qu'ils peuvent être utilisés par leur script 'patch-o-matic' (aussi appelé 'p-o-m' ou 'pom'). Le script va vous aider a sélectionner les patches que vous voulez appliquer et va les appliquer pour vous.

D'abord, vous devriez downloader la dernière version de netfilter à partir du CVS pour être sûr que vous utilisez les dernières extensions, pour cela lancez à partir du shell :

```
# cvs -d :pserver:cvs@pserver.netfilter.org:/cvspublic login

(quand on vous demande un mot de passe, tapez ‘cvs’).

# cvs -d :pserver:cvs@pserver.netfilter.org:/cvspublic co netfilter/userspace netfilter/patch-o-matic
```

Ça va créer le répertoire de base 'netfilter/', et va mettre tous les fichiers nécessaires dedans pour vous :

```
# ls -l netfilter/
total 3
drwxr-xr-x    2 root     root          160 Nov  7 14:48 CVS/
```

```
drwxr-xr-x   13 root     root          488 Nov  7 14:54 patch-o-matic/
drwxr-xr-x    9 root     root          864 Nov  7 14:48 userspace/
```

Faites attention à ce que votre noyau soit dans '/usr/src/linux/'. Si pour quelque raison que ce soit, le noyau que vous voulez patcher ne ce trouve pas dans '/usr/src/linux/', alors vous pouvez faire pointer la variable KERNEL_DIR vers le chemin correct :

```
# export KERNEL_DIR=/chemin/vers/source/linux
```

Faites attention à ce que les dépendances soient faites. Pour être sûr, tapez :

```
# cd /usr/src/linux/
# make dep
```

Ensuite vous pouvez retourner dans le répertoire de netfilter, dans le sous-répertoire 'patch-o-matic/'. Vous pouvez ensuite lancer p-o-m.

2.2 Utiliser Patch-O-Matic

En étant dans le répertoire 'patch-o-matic/', lancez p-o-m :

```
# ./runme extra

Welcome to Rusty's Patch-o-matic!

Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so I don't recommend applying them all!
-------------------------------------------------------

Already applied: 2.4.1 2.4.4
Testing... name_of_the_patch NOT APPLIED ( 2 missing files)
The name_of_the_patch patch:
   Here usually is the help text describing what
   the patch is for, what you can expect from it,
   and what you should not expect from it.
Do you want to apply this patch [N/y/t/f/q/?]
```

p-o-m va vous présenter la plupart des patches non appliqués un par un. Si un patch est déjà appliqué, il apparaitra sur la ligne 'Already Applied'. Si le patch n'est pas encore appliqué, son nom va apparaître, ainsi que un petit texte d'explication détaillant l'utilité du patch. p-o-m vous explique ce qui ce passe : 'NOT APPLIED (n missing files)' signifie que le patch n'a pas encore été appliqué, alors que 'NOT APPLIED (n reject out of n chunks)' signifie en général que :

1. Soit le patch ne peut pas être appliqué proprement...
2. ...soit le patch a déjà été inclus dans le noyau que vous essayez de patcher.

Finalement, il va vous demander si vous voulez appliquer ce patch ou non.

– Appuyez sur entrée si vous ne voulez pas appliquer le patch.
– Tapez 'y' puis entrée si vous voulez appliquer le patch, si la tentative échoue, alors p-o-m vous avertira, et vous demandera encore pour confirmation. Si le patch s'applique, alors vous verrez son nom apparaître sur la ligne 'Already Applied'.
– Tapez 't' puis entrée si vous voulez tester seulement si le patch pourrait s'appliquer correctement.
– Tapez 'f' puis entrée si vous voulez forcer p-o-m à appliquer le patch.
– Finalement, tapez 'q' puis entrée si vous voulez quitter p-o-m.

Une règle de bon sens est de lire attentivement le petit texte d'aide de chaque patch avant de l'appliquer. Comme il y a pour le moment BEAUCOUP de patches officiels (et certainement beaucoup d'autres non officiels), ce n'est pas une bonne idée de tous les appliquer ! Vous devriez vraiment choisir correctement pour appliquer seulement ceux dont vous avez besoin, même si ça veut dire recompiler le tout quand le besoin pour une autre extension arrive.

Patch-o-matic est principalement le script shell './runme'. Si vous le lancez la commande sans paramètre, le message d'aide suivant s'affichera :

```
Usage: ./runme [--batch] [--reverse] [--exclude suite/patch-file ...] suite|suite/patch-file

  --batch       batch mode, automatically applying patches
                (mode batch, applique les patches automatiquement).
  --reverse     back out the selected patches
                ('patch -R' les patches sélectionnés).
  --exclude     excludes the named patches
                (exclus les patches nommés).
```

Les patches sont contenus dans 'patch-o-matic/pending/', 'patch-o-matic/base', etc.. Ici, 'pending' et 'base' sont deux nom de suite. Tapez ls dans le répertoire 'patch-o-matic' pour voir toutes les suites disponibles. Exemples de commandes 'runme' :

```
./runme --batch pending
./runme --batch userspace/ipt_REJECT-fake-source.patch
```

La première commande va essayer d'appliquer tous les patches qui sont dans la suite submitted, et ensuite, tous les patches qui sont dans la suite pending (nous expliquons plus loin pourquoi deux suites d'un coup). La seconde commande va seulement appliquer le patch 'ipt_REJECT-fake-source.patch' de la suite userspace.

Les suites de patches les plus signifiantes sont (dans l'ordre qu'elles doivent être appliquées) :
- submitted
- pending
- base
- extra
- userspace

Quand vous demandez à './runme' d'appliquer les patches de la suite 'extra/', il va d'abord appliquer ceux des suites 'submitted/', 'pending/', et 'base/'. Chaque suite maintient son propre fichier nommé 'SUITE' qui décrit l'ordre dans lequel patch-o-matic doit tenter d'appliquer les patches. Par exemple, ce que j'ai décrit plus haut est très vite expliqué avec le contenu du fichier 'SUITE' de la suite 'userspace/' :

```
# cat userspace/SUITE
submitted pending base extra userspace
```

2.3 Et après ?

Une fois que vous avez appliqué tous les patches que vous vouliez, l'étape suivante est de recompiler votre noyau et de l'installer. Ce HOWTO ne va pas expliquer comment faire ça. A la place, vous devriez lire le *Linux Kernel HOWTO* http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html .

Pendant que vous configurez votre noyau, vous verrez apparaître de nouvelles options dans "Networking Options -> Netfilter Configuration". Choisissez les options dont vous avez besoin, recompilez puis installez votre nouveau noyau.

Une fois que votre nouveau noyau est installé, continuez et installez le package "iptables", à partir du répertoire 'userspace/' comme suit :

```
# make && make install
```

Et voilà ! Votre tout nouveau package iptables tout beau est maintenant installé ! Il est temps d'utiliser les nouvelles fonctionnalités offertes.

3 Nouveaux matches de netfilter.

Dans cette section, on essayera d'expliquer l'utilisation de ces nouveaux matches de netfilter. Les patches seront abordés par ordre alphabétique. Aussi, nous n'expliquerons pas les patches qui cassent les autres patches. Mais ça pourra venir ensuite.

De manière générale, pour les matches, vous pouvez voir le petit synopsis de chacun d'eux en tapant :

```
# iptables -m le_match_que_vous_voulez --help
```

Cela va afficher le texte d'aide normal d'iptables, et le petit synopsis spécifique à "le_match_que_vous_voulez" à la fin.

3.1 Le patch ah-esp

Ce patch par Yon Uriarte <yon@astaro.de> ajoute 2 nouveaux matches :

– "ah" : vous permet de matcher un paquet IPSEC basé sur son Security Parameter Index (SPI)
– "esp" : vous permet de matcher un paquet IPSEC basé sur son SPI.

Ce patch peut être très utile pour ceux d'entre vous qui utilisent IPSEC et qui voudraient discriminer les connexions basées sur leur SPI.

Par exemple, on va DROPper tous les paquets AH qui ont un SPI égal à 500 :

```
# iptables -A INPUT -p 51 -m ah --ahspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       ipv6-auth--  anywhere             anywhere           ah spi:500
```

Les options supportées pour le match ah sont :

–ahspi [!] spi[:spi]
-> match le spi (intervalle)

Le match esp marche exactement pareil :

```
# iptables -A INPUT -p 50 -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       ipv6-crypt--  anywhere             anywhere           esp spi:500
```

Les options supportées pour le match esp sont :

–espspi [!] spi[:spi]
-> match le spi (intervalle)

N'oubliez pas de spécifier le bon protocole en utilisant l'option "-p 50" ou "-p 51" (pour esp & ah respectivement) quand vous utilisez le match ah ou esp, sinon la règle de filtrage ne s'insèrera pas, pour une raison triviale...

3.2 Le patch condition

Ce patch par Stephane Ouellette <ouellettes@videotron.ca> ajoute un nouveau match qui vous permet d'autoriser ou de neutraliser un jeu de règles en utilisant les variables de condition stockées dans des fichiers '/proc'.

Notes :

- Les variables de condition sont stockées dans le répertoire '/proc/net/ipt_condition/'.
- Une variable de condition peut être seulement mise à "0" (VRAI) ou à "1" (FAUX).
- Une ou plusieurs règles peuvent êtres affectées par l'état d'une seule variable de condition.
- Un fichier proc de condition est crée automatiquement quand une nouvelle condition est référencée pour la premiere fois.
- Un fichier proc de condition est effacé automatiquement quand la dernière référence vers cette condition est effacée.

Les options supportées pour le match condition match sont :

–condition [!] conditionfile
-> match la condition booléenne.

Par exemple, si vous voulez interdire l'accès à votre serveur web pendant que vous faites de la maintenance, vous pouvez utiliser ce qui suit :

```
# iptables -A FORWARD -p tcp -d 192.168.1.10 --dport http -m condition --condition webdown -j REJECT --reject-with tcp-reset

# echo 1 > /proc/net/ipt_condition/webdown
```

La règle suivant va matcher seulement si la condition "webdown" est mise à "1".

3.3 Le patch conntrack

Ce patch par Marc Boucher <marc+nf@mbsi.ca> ajoute un nouveau match (un superset du match 'state') qui permet de matcher sur plus de critères en relation avec le suivi de connexions.

Par exemple, si vous voulez ACCEPTer toutes les connexions 'RELATED' mais seulement pour le protocole TCP, vous pouvez faire comme suit :

```
# iptables -A FORWARD -m conntrack --ctstate RELATED --ctproto tcp -j ACCEPT

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere           ctstate RELATED
```

Les options supportées pour le match conntrack sont :

[!] –ctstate [INVALID|ESTABLISHED|NEW|RELATED|SNAT|DNAT][,...]
-> L(es) 'état(s) à matcher. Les "nouveaux" états 'SNAT' et 'DNAT' sont virtuels, et matchent si l'adresse source originale diffère de la destination réponse, ou si la destination originale diffère de la réponse source.

[!] –ctproto proto

-> Le protocole à matcher ; Indiqué par son numéro (ex : 6) ou son nom (ex : 'tcp').

–ctorigsrc [!] address[/mask]

-> Spécifie l'adresse source originale.

–ctorigdst [!] address[/mask]

-> Spécifie l'adresse destination originale.

–ctreplsrc [!] address[/mask]

-> Spécifie l'adresse source de réponse.

–ctrepldst [!] address[/mask]

-> Spécifie l'adresse destination de réponse.

[!] –ctstatus [NONE|EXPECTED|SEEN_REPLY|ASSURED][,...]

-> Le(s) status à matcher.

[!] –ctexpire time[:time]

-> Match le temps restant (intervalle inclusif).

3.4 Le patch fuzzy

Ce patch par Hime Aguiar e Oliveira Jr. <hime@engineer.com> ajoute un nouveau module qui vous permet de matcher un paquet d'après un profile dynamique implémenté par voie d'un simple contrôleur de logique floue (Fuzzy Logic Controller, FLC).

Ce match implémente un TSK FLC (Takagi-Sugeno-Kang Fuzzy Logic Controller). L'idée de base est que l'on donne au match deux paramètres pour décrire l'intervalle de filtrage.

- Quand le débit des paquets est en dessous de 'lower-limit' la règle ne matchera jamais.
- Entre 'lower-limit' et 'upper-limit', plus le débit des paquets s'approche de 'upper-limit' et plus le module matchera.
- Finalement, quand le débit des paquets atteint 'upper-limit', le module atteint son maximum, et matche les paquets à 99%.

En prenant en compte que l'échantillonnage du débit est variable et qu'il est approximativement de 100ms (sur une machine très occupée), l'auteur du module pense que ce module présente une bonne réponse, qui s'adapte rapidement aux changement de trafic.

Par exemple, si vous voulez éviter les Denials Of Service, vous pourriez utiliser la règle suivante :

```
iptables -A INPUT -m fuzzy --lower-limit 100 --upper-limit 1000 -j REJECT
```

- En dessous du débit de 100 pps (paquets par seconde), le filtre est inactif.
- Entre 100 et 1000 pps, l'acceptance moyenne tombe de 100% (quand on est à 100pps) à 1% (quand on est a 1000pps).
- Au dessus de 1000pps, l'acceptance moyenne reste constante, à 1%.

Les options supportées pour le match fuzzy sont :

–upper-limit n

-> La limite supérieure.

–lower-limit n

-> La limite inférieure.

3.5 Le patch iplimit

Ce patch par Gerd Knorr <kraxel@bytesex.org> ajoute un match qui vous permet de restreindre le nombre de connexions TCP parallèles à partir d'une machine en particulier ou d'un réseau particulier.

Par exemple, on va limiter le nombre de connexions HTTP parallèles faites par une seule adresse IP à 4 :

```
# iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-above 4 -j REJECT

# iptables --list
Chain INPUT (policy ACCEPT)
target   prot opt source    destination
REJECT   tcp  --  anywhere  anywhere     tcp dpt:http flags:SYN,RST,ACK/SYN #conn/32 > 4 reject-with icmp-port-unreachable
```

Ou vous voudriez peut être limiter à 5 le nombre de connexions parallèles faites à partir d'un réseau de classe A par exemple :

```
# iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-mask 8 --iplimit-above 5 -j REJECT

# iptables --list
Chain INPUT (policy ACCEPT)
target   prot opt source    destination
REJECT   tcp  --  anywhere  anywhere     tcp dpt:http flags:SYN,RST,ACK/SYN #conn/8 > 5 reject-with icmp-port-unreachable
```

Les options supportées par le match iplimit sont :

[!] –iplimit-above n

-> match si le nombre de connexions TCP existantes (n') est (pas) au dessus de n.

–iplimit-mask n

-> netmask si vous voulez un réseau à la place d'une seule machine.

3.6 Le patch ipv4options

Ce patch par Fabrice MARIE <fabrice@netfilter.org> ajoute un nouveau match qui vous permet de matcher un paquet basé sur les options IP qui ont été mises.

Par exemple, nous allons DROPper tous les paquets qui ont l'option record-route et l'option timestamp mise :

```
# iptables -A INPUT -m ipv4options --rr -j DROP
# iptables -A INPUT -m ipv4options --ts -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere           IPV4OPTS RR
DROP       all  --  anywhere             anywhere           IPV4OPTS TS
```

Les options supportées pour le match ipv4options sont :

–ssrr

-> match l'option strict source routing.

–lsrr

-> match l'option loose source routing.

–no-srr

-> match les paquets sans source-routing du tout.

[!] –rr

-> match l'option record-route.

[!] –ts

-> match l'option timestamp.

[!] –ra

-> match l'option router-alert.

[!] –any-opt

-> match un paquet qui a au moins une option de mise (ou pas d'option du tout si ! est spécifié).

3.7 Le patch length

Ce patch par James Morris <jmorris@intercode.com.au> vous permet de matcher un paquet basé sur sa taille.

Par exemple, nous allons DROPper tous les pings qui ont une taille de paquet plus grande que 85 octets :

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m length --length 86:0xffff -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           icmp echo-request length 86:65535
```

Les options supportées pour le match length sont :

[!] –length length[:length]

-> match la taille du paquet (bornes inclues)

Les valeurs non présentes seront mises d'office à leur défaut. La valeur par défaut du minimum est 0, alors que la valeur par défaut du maximum est 65535.

3.8 Le patch mport

Ce patch par Andreas Ferber <af@devcon.net> ajoute un nouveau match qui vous permet de spécifier des ports avec un mélange entre des intervalles de ports, et des ports en eux même, aussi bien pour TCP que pour UDP.

Par exemple, si vous voulez bloquer ftp, ssh, telnet et HTTP en une ligne, faites comme suit :

```
# iptables -A INPUT -p tcp -m mport --ports 20:23,80 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere           mport ports ftp-data:telnet,http
```

Les options supportées pour l'option mport sont :

–source-ports port[,port :port,port...]

->

–sports port[,port :port,port...]

-> match le(s) port(s) source.

–destination-ports port[,port :port,port...]

-> match le(s) port(s) de destination.

–dports port[,port :port,port...]

-> match le(s) port(s) de destination.

–ports port[,port :port,port]

-> match le(s) port(s) source ou destination.

3.9 Le patch nth

Ce patch par Fabrice MARIE <fabrice@netfilter.org> ajoute un nouveau match qui vous permet de matcher le Nième paquet revu par la règle.

Par exemple, si vous voulez DROPper un paquet sur deux, faites comme suit :

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m nth --every 2 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           icmp echo-request every 2th
```

Des extensions écrites par Richard Wagner <rwagner@cloudnet.com> vous permettent de créer une méthode simple et rapide pour faire du load-balancing :

Par exemple, si vous voulez balancer la charge entre les 3 adresses IP 10.0.0.5, 10.0.0.6 et 10.0.0.7, alors vous pouvez faire comme suit :

```
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 0 -j SNAT --to-source 10.0.0.5
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 1 -j SNAT --to-source 10.0.0.6
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 2 -j SNAT --to-source 10.0.0.7

# iptables -t nat --list
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       all  --  anywhere             anywhere           every 3th packet #0 to:10.0.0.5
SNAT       all  --  anywhere             anywhere           every 3th packet #1 to:10.0.0.6
SNAT       all  --  anywhere             anywhere           every 3th packet #2 to:10.0.0.7
```

Les options supportées par le match nth sont :

–every Nth

-> match chaque Nième paquet.

[–counter] num

-> utiliser le compteur numéro (0-15, défaut 0).

[–start] num

-> Initialise le compteur à la valeur ‘num’ à la place de 0. ‘num’ doit être entre 0 et (Nth-1).

[–packet] num

-> Match sur le paquet numéro ‘num’. ‘num’ doit être entre 0 et (Nth-1). Si cette option est utilisée, alors il doit y avoir autant de règles avec cette option, qu’il y a de valeurs entre 0 et (Nth-1), de manière à couvrir l’intervalle.

3.10 Le patch pkttype.

Ce patch par Michal Ludvig <michal@logix.cz> ajoute un nouveau match qui vous permet de matcher un paquet basé sur son type : host/broadcast/multicast.

Si par exemple vous voulez DROPper silencieusement tous les paquets broadcastés :

```
# iptables -A INPUT -m pkttype --pkt-type broadcast -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere           PKTTYPE = broadcast
```

Les options supportées par le match pkttype sont :

–pkt-type [!] packettype

-> Match le type de paquet, où le type de paquet est l'un des suivants :

host

-> dirige vers une machine en particulier.

broadcast

-> dirige vers tout le monde.

multicast

-> dirige vers un groupe de machines.

3.11 Le patch pool

Ce patch a été écrit par Patrick Schaaf <bof@bof.de>. Joakim Axelsson et Patrick sont en train de réécrire ce match, et donc ils vont remplacer cette section dès que le travail sera fini.

3.12 Le match psd

Ce match par Dennis Koslowski <dkoslowski@astaro.de> ajoute un nouveau match qui essaie de détecter les port-scans.

Dans sa forme la plus simple, le match psd peut être utilisé comme suit :

```
# iptables -A INPUT -m psd -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target  prot opt source    destination
DROP    all  --  anywhere  anywhere    psd weight-threshold: 21 delay-threshold: 300 lo-ports-weight: 3 hi-ports-weight: 1
```

Les options supportées pour le match psd sont :

[–psd-weight-threshold threshold]

-> Le poids limite de détection.

[–psd-delay-threshold delay]

-> Le délai limite de détection.

[–psd-lo-ports-weight lo]

-> Poids des ports privilégiés.

[–psd-hi-ports-weight hi]

-> Poids des ports non privilégiés.

3.13 Le patch quota

Ce patch par Sam Johnston <samj@samj.net> ajoute un nouveau match qui vous permet de mettre en place des quotas. Une fois que le quota a été atteint, la règle ne matche plus.

Par exemple, si vous voulez mettre un quota de 50Megs sur les données HTTP entrantes, faites comme suit :

```
# iptables -A INPUT -p tcp --dport 80 -m quota --quota 52428800 -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere           tcp dpt:http quota: 52428800 bytes
DROP       tcp  --  anywhere             anywhere           tcp dpt:http
```

Les options supportées par le match quota sont :

–quota quota

-> Le quota que vous voulez mettre en place.

3.14 Le patch random

Ce patch par Fabrice MARIE <fabrice@netfilter.org> ajoute un nouveau match qui vous permet de matcher un paquet de manière aléatoire, basé sur une probabilité.

Par exemple, si vous voulez DROPper 50% des pings de manière aléatoire, faites comme suit :

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m random --average 50 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source       destination
DROP       icmp --  anywhere     anywhere        icmp echo-request  random 50%
```

Les options supportées par le match random sont :

[–average percent]

-> La probabilité de match exprimée sous forme de pourcentage. Si omise, cette probabilité sera de 50% par défaut. Le pourcentage doit être entre 1 et 99 compris.

3.15 Le patch realm

Ce patch par Sampsa Ranta <sampsa@netsonic.fi> ajoute un nouveau match qui vous permet d'utiliser une clé realm (venant du code de routing) comme match similaire à ceux trouvés dans le packet classifier.

Par exemple, pour LOGguer tous les paquets sortants qui ont un realm égal à 10, faites comme suit :

```
# iptables -A OUTPUT -m realm --realm 10 -j LOG

# iptables --list
```

```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere           REALM match 0xa LOG level warning
```

Les options supportées pour le match realm sont :

–realm [!] value[/mask]
: -> Match le realm

3.16 Le patch recent

Ce patch par Stephen Frost <sfrost@snowman.net> ajoute un nouveau match qui vous permet de créer dynamiquement une liste d'adresses IP et ensuite de matcher des paquets par rapport à cette liste de différentes manières.

Par exemple, si vous voulez créer une liste 'lesmechants' contenant l'adresse IP des gens qui essayent de se connecter au port 139 sur votre firewall, pour pouvoir ensuite les bloquer pendant 60 secondes, faites comme suit :

```
# iptables -A FORWARD -m recent --name lesmechants --rcheck --seconds 60 -j DROP
# iptables -A FORWARD -p tcp -i eth0 --dport 139 -m recent --name lesmechants --set -j DROP

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere           recent: CHECK seconds: 60
DROP       tcp  --  anywhere             anywhere           tcp dpt:netbios-ssn recent: SET
```

Les options supportées pour le match recent sont :

–name name
: -> Spécifie le nom de la liste sur laquelle il faut exécuter les commandes. si aucun nom n'est donné, alors le nom 'DEFAULT' sera utilisé.

[!] –set
: -> Ceci va ajouter l'adresse source du paquet à la liste. Si l'adresse source est déjà dans la liste, alors ça va mettre à jour l'entrée existante. Ça va toujours matcher ; ou ne matchera jamais si la négation ' !' a été spécifiée.

[!] –rcheck
: -> Ceci va vérifier si l'adresse source du paquet est pour le moment dans la liste, et va matcher si l'adresse y est, et ne matchera pas si l'adresse n'y est pas. L'effet inverse est obtenu en spécifiant la négation ' !'.

[!] –update
: -> Ceci va vérifier si l'adresse source du paquet est pour le moment dans la liste. Si elle y est, alors l'entrée va être mise à jour, et la règle va matcher. Si l'adresse n'est pas dans la liste pour le moment, alors la règle ne matchera pas. L'effet inverse est obtenu si la négation ' !' est spécifiée.

[!] –remove
: -> Cela va vérifier si l'adresse source du paquet est pour le moment dans la liste. Si elle y est, alors l'entrée sera retirée et la règle va matcher. Si l'adresse n'est pas trouvée dans la liste, la règle ne matchera pas. L'effet inverse est obtenu si la négation ' !' est spécifiée.

[!] –seconds seconds

-> Cette option doit être utilisée avec 'rcheck' ou 'update'. Quand cette option est utilisée, ça va faire en sorte que la règle ne matche que si l'adresse est pour le moment dans la liste et que l'adresse a été vue dans l'intervalle donné (en secondes). L'effet inverse est obtenu en spécifiant la négation ' !'.

[!] –hitcount hits

-> Cette option doit être utilisée avec 'rcheck' ou 'update'. Quand cette option est utilisée, cela va faire en sorte que la règle ne matche que si l'adresse est pour le moment dans la liste et qu'on a reçu un nombre de paquets supérieur ou égal a 'hits'. Cette option peut être utilisée en même temps que 'seconds' pour obtenir un match très précis qui requiert qu'un certain nombre de paquets soient vus dans une certaine limite de temps. L'effet inverse est obtenu si la négation ' !' est spécifiée..

–rttl

-> Cette option doit être utilisée avec 'rcheck' ou 'update'. Quand elle est utilisée, ça va faire en sorte que la règle ne matche que si l'adresse est dans la liste et que le TTL du paquet que l'on inspecte maintenant est le même que celui du paquet qui a créé l'entrée initialement dans la règle avec '–set'. Ça peut être utile si vous avez des problèmes avec des gens qui "spoof"ent leur adresse source pour parvenir a un DoS au travers de ce module.

3.17 Le patch record-rpc

Ce patch par Marcelo Barbosa Lima <marcelo.lima@dcc.unicamp.br> ajoute un nouveau match qui vous permet de matcher un paquet si l'adresse source de ce paquet fait une requête avant sur le portmapper, ou si c'est une nouvelle requête GET vers le portmapper, permettant un filtrage précis du protocole RPC.

Pour matcher les informations de suivi de connexions RPC, faites comme suit :

```
# iptables -A INPUT -m record_rpc -j ACCEPT

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
```

Le match record_rpc n'a aucune option pour le moment.

Ne vous en faites pas pour les informations de match qui ne s'affichent pas quand vous listez la règle, c'est juste que la fonction print() est vide :

```
/* Prints out the union ipt_matchinfo. */
static void
print(const struct ipt_ip *ip,
      const struct ipt_entry_match *match,
      int numeric)
{
}
```

3.18 Le patch string

Ce patch par Emmanuel Roger <winfield@freegates.be> ajoute un nouveau match qui vous permet de matcher une chaîne de caractères n'importe où dans le paquet.

Par exemple, si vous voulez matcher tous les paquets contenant la chaîne "cmd.exe" n'importe où dans le paquet et rediriger le paquet vers la queue en userland pour qu'il soit analysé ensuite par un IDS, faites comme suit :

```
# iptables -A INPUT -m string --string 'cmd.exe' -j QUEUE

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
QUEUE      all  --  anywhere             anywhere           STRING match cmd.exe
```

Utilisez ce match avec précautions ! Beaucoup de gens veulent utiliser ce match pour arrêter les worms (en DROPpant le paquet louche). C'est une erreur classique. Ça ne marche pas, et puis quand ça marche, ça peut être battu sans problème en utilisant n'importe quelle méthode d'évasion d'IDS.

Dans le même genre, beaucoup de gens veulent utiliser ce match comme moyen de filtrer des fonctions en particulier dans le protocole HTTP par exemple comme POST ou GET en DROPpant n'importe quel paquet qui contient la dite chaîne de caractères. Il faut que vous compreniez que ce travail est mieux effectué par un proxy filtrant en userland. De plus, n'importe quelle page HTML contenant le mot POST serait DROPpé avec cette méthode. Gardez à l'esprit que ce match a été crée dans l'intention de rediriger vers le userland les paquets louches afin d'avoir une meilleure analyse, c'est tout. DROPper un paquet en se basant uniquement sur ce match est une erreur.

Les options supportées pour le match string sont :

–string [!] string
-> Match une string *n'importe ou* dans le paquet.

3.19 Le patch time

Ce patch par Fabrice MARIE <fabrice@netfilter.org> ajoute un nouveau match qui vous permet de matcher un paquet basé sur son temps d'arrivée ou de départ (pour les paquets localement générés).

Par exemple, si vous voulez ACCEPTer tous les paquets qui arrivent entre 8H00 et 18H00 du lundi au vendredi, faites comme suit :

```
# iptables -A INPUT -m time --timestart 8:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source           destination
ACCEPT     all  --  anywhere         anywhere        TIME from 8:0 to 18:0 on Mon,Tue,Wed,Thu,Fri
```

Les options supportées par le match time sont :

–timestart value
-> minimum HH :MM

–timestop value
-> maximum HH :MM

–days listofdays
-> une liste de jour, à partir de la liste suivante (attention à la casse)
- Mon -> lundi
- Tue -> mardi
- Wed -> mercredi
- Thu -> jeudi
- Fri -> vendredi
- Sat -> samedi
- Sun -> dimanche

3.20 Le patch ttl

Ce patch par Harald Welte <laforge@gnumonks.org> ajoute un nouveau match qui vous permet de matcher un paquet basé sur son TTL.

Par exemple, si vous voulez LOGguer tous les paquets qui ont un TTL inférieur à 5, faites comme suit :

```
# iptables -A INPUT -m ttl --ttl-lt 5 -j LOG

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere           TTL match TTL < 5 LOG level warning
```

Les options supportées par le match ttl sont :

–ttl-eq value
-> Match le TTL exact

–ttl-lt value
-> Match TTL < value

–ttl-gt value
-> Match TTL > value

4 Nouvelles targets de netfilter

Dans cette section, on essayera d'expliquer l'utilisation de ces nouvelles targets de netfilter. Les patches seront abordés par ordre alphabétique. Aussi, nous n'expliquerons pas les patches qui cassent les autres patches. Mais ça pourra venir ensuite.

De manière générale, pour les targets, vous pouvez voir le petit synopsis de chacune d'elles en tapant :

```
# iptables -j LA_TARGET_QUE_VOUS_VOULEZ --help
```

Ça va afficher le texte d'aide normal d'iptables, et le petit synopsis spécifique de "LA_TARGET_QUE_VOUS_VOULEZ" à la fin.

4.1 Le patch ftos

Ce patch par Matthew G. Marsh <mgm@paktronix.com> ajoute une nouvelle target qui vous permet de changer le TOS du paquet vers une valeur arbitraire.

Par exemple, si vous voulez changer le TOS de tous les paquets sortants à 15, faites comme suit :

```
# iptables -t mangle -A OUTPUT -j FTOS --set-ftos 15

# iptables -t mangle --list
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
FTOS       all  --  anywhere             anywhere           TOS set 0x0f
```

Les options supportées par la target FTOS sont :

–set-ftos value
-> Change le champs TOS du paquet à 'value'. Cette valeur peut être en décimal (ex : `32`) ou en héxa (ex : `0x20`).

4.2 Le patch IPV4OPTSSTRIP

Ce patch par Fabrice MARIE <fabrice@netfilter.org> ajoute une nouvelle target qui retire toute option IPv4 d'un paquet.

On peut l'utiliser comme suit :

```
# iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP

# iptables -t mangle --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
IPV4OPTSSTRIP  all  --  anywhere             anywhere
```

Cette target ne prend aucune option.

4.3 Le patch NETLINK

Ce patch Gianni Tedesco <gianni@ecsc.co.uk> ajoute une nouvelle target qui vous permet d'envoyer les paquets DROPpés vers le userspace via une socket netlink.

Par exemple, si vous voulez DROPper tous les pings et les envoyer à une socket netlink en userland à la place, faites comme suit :

```
# iptables -A INPUT -p icmp --icmp-type echo-request -j NETLINK --nldrop

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
NETLINK    icmp --  anywhere             anywhere           icmp echo-request nldrop
```

Les options supportées pour la target NETLINK sont :

–nldrop
-> DROP le paquet aussi.

–nlmark <number>
-> Marque le paquet

–nlsize <bytes>
-> Limite la taille du paquet.

Pour plus d'information sur les sockets netlink, vous pouvez vous référer au document

Netlink Sockets Tour http://www.skyfree.org/linux/kernel_network/netlink.html .

4.4 Le patch NETMAP

Ce patch par Svenning Soerensen <svenning@post5.tele.dk> ajoute une nouvelle target qui vous permet de créer un lien statique 1 :1 d'une adresse réseau, tout en gardant l'adresse de la machine intacte.

Par exemple, si vous voulez changer la destination de toutes les connexions entrantes, de 1.2.3.0/24 vers 5.6.7.0/24, faites comme suit :

```
# iptables -t nat -A PREROUTING -d 1.2.3.0/24 -j NETMAP --to 5.6.7.0/24

# iptables -t nat --list
```

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
NETMAP     all  --  anywhere             1.2.3.0/24          5.6.7.0/24
```

Les options supportées par la target NETMAP sont :

–to address[/mask]

-> L'adresse réseau à mapper.

4.5 Le patch ROUTE

Ce patch par Cédric de Launois <delaunois@info.ucl.ac.be> ajoute une nouvelle target qui vous d'établir des routes particulières qui ne sont pas supportées par la table de routage standard du noyau. La target ROUTE vous permet the router un paquet vers une interface ou un hôte, même si la destination normale du paquet est le routeur lui-même. La target ROUTE permet également de changer l'interface d'arrivée d'un paquet. Les paquets sont envoyés directement sur la ligne, sans traverser aucune autre table.

Cette target ne modifie jamais les paquets et est une target finale. Elle doit être utilisée dans la table 'mangle'.

Lorsque c'est possible, il est préférable d'utiliser la target MARK couplée à l'utilisation d'iproute2. Cette target ROUTE est cependant utile pour forcer l'utilisation d'une interface ou d'une passerelle, et pour changer l'interface d'arrivée d'un paquet. Certaines personnes l'utilisent aussi par facilité et pour simplifier leur règles (une règle pour router un paquet est plus facile qu'une règle MARK + une règle iproute2).

Les options supportées par la target ROUTE sont :

–oif ifname

Envoie le paquet en utilisant l'interface de nom 'ifname'. La machine de destination doit être sur le même lien, ou bien l'interface doit être un tunnel. Dans le cas contraire, la résolution arp ne peut se faire et le paquet est jeté.

–iif ifname

Modifie l'interface d'arrivée du packet en l'interface 'ifname'.

–gw ip

Route le paquet via cette passerelle. Le paquet est routé comme si son adresse IP de destination était cette ip.

Supposons par exemple que vous voulez rediriger les paquets ssh vers un serveur situé à l'intérieur de votre réseau, sans modifier ces paquets (ceci exclu l'utilisation de la technique de redirection de port). Une solution est d'utiliser un tunnel ipip et la target ROUTE pour rerouter les paquets ssh vers le vrai serveur ssh, qui a la même adresse IP que le routeur. Il n'est pas possible de rerouter ces paquets en utilisant la table de routage du noyau car le noyau délivre localement les paquets ayant une adresse de destination appartenant au routeur lui-même.

Un peu d'art ASCII :

```
          eth0  +-------+ 192.168.0.1         192.168.0.2 +----+
----------------|routeur|---------------------------------|hôte|
IP: 150.150.0.1 +-------+                                 +----+
                   | | tunl1              IP: 150.150.0.1 | |
                   | +----------------------------------+ |
                   +--------------------------------------+
                                tunnel IPIP
```

Pour l'exemple ci-dessus, vous pouvez faire comme suit :

```
# iptables -A PREROUTING -t mangle -i eth0 -p tcp --dport 22 -j ROUTE --oif tunl1
# iptables -A PREROUTING -t mangle -i tunl1 -j ROUTE --oif eth0

# iptables -L PREROUTING -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ROUTE      tcp  --  anywhere             anywhere           tcp dpt:ssh ROUTE oif tunl1
ROUTE      all  --  anywhere             anywhere           ROUTE oif eth0
```

Autre exemple : si vous voulez facilement et rapidement répartir la charge entre deux passerelles 10.0.0.1 et 10.0.0.2, faites comme suit :

```
# iptables -A PREROUTING -t mangle -m random --average 50 -j ROUTE --gw 10.0.0.1
# iptables -A PREROUTING -t mangle -j ROUTE --gw 10.0.0.2

# iptables -L PREROUTING -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ROUTE      all  --  anywhere             anywhere           random 50% ROUTE gw 10.0.0.1
ROUTE      all  --  anywhere             anywhere           ROUTE gw 10.0.0.2
```

4.6 Le patch SAME

Ce patch par Martin Josefsson <gandalf@wlug.westbo.se> ajoute une nouvelle target similaire à SNAT et qui donne au client la même adresse à chaque connexion.

Par exemple, si vous voulez modifier l'adresse source de toutes les connexions pour qu'elles soient 1.2.3.4-1.2.3.7, faites comme suit :

```
# iptables -t nat -A POSTROUTING -j SAME --to 1.2.3.4-1.2.3.7

# iptables -t nat --list
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SAME       all  --  anywhere             anywhere           same:1.2.3.4-1.2.3.7
```

Les options supportées par la target SAME sont :

–to <ipaddr>-<ipaddr>
: -> L'adresse vers laquelle il faut mapper la connexion.

–nodst
: -> N'utilise pas l'adresse IP destination dans la sélection de la source.

4.7 Le patch tcp-MSS

Ce patch par Marc Boucher <marc+nf@mbsi.ca> ajoute une nouvelle target qui vous permet d'examiner et de changer le MSS dans les paquets TCP SYN, pour contrôler la taille maximum pour cette connexion.

Comme l'explique Marc lui même, c'est un hack, utilisé pour résoudre les problèmes engendrés par ces ISPs têtus comme des mules qui bloquent les paquets ICMP Fragmentation Needed à tort.

Sont utilisation typique est :

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
TCPMSS     tcp  --  anywhere             anywhere           tcp flags:SYN,RST/SYN TCPMSS clamp to PMTU
```

Les options supportées par la target tcp-MSS sont (et sont mutuellement exclusives) :

–set-mss value

mettre la valeur du MSS explicitement à la valeur donnée.

–clamp-mss-to-pmtu

Bloque la valeur du MSS à (path_MTU - 40)

4.8 Le patch TTL

Ce patch par Harald Welte <laforge@gnumonks.org> ajoute une nouvelle target qui vous permet de changer le TTL d'un paquet vers une valeur explicite, ou d'incrémenter/décrémenter le TTL par une valeur donnée.

Par exemple, si vous voulez changer le TTL de toutes les connexions sortantes à 126, faites comme suit :

```
# iptables -t mangle -A OUTPUT -j TTL --ttl-set 126

# iptables -t mangle --list
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
TTL        all  --  anywhere             anywhere           TTL set to 126
```

Les options supportées par la target TTL sont :

–ttl-set value

-> Met le TTL à la valeur <value>

–ttl-dec value

-> Décrémente le TTL par <value>

–ttl-inc value

-> Incrémente le TTL par <value>

4.9 Le patch ulog

Ce patch par Harald Welte <laforge@gnumonks.org> ajoute une nouvelle target qui fournit un mécanisme un peu plus avancé que la target LOG standard pour logguer les paquets. Le répertoire 'libipulog/' contient une librairie pour recevoir les messages ULOG en userland.

Harald maintient une

page web `http://www.gnumonks.org/projects/ulogd` contenant de la documentation sur l'utilisation de la target ULOG ainsi que sur la librairie, et donc on ne l'expliquera pas ici..

5 Les nouveaux patches de suivi de connexion

Dans cette section, nous allons montrer les différents patches à disposition pour permettre un meilleur suivi de connexions. Pour les utiliser, chargez simplement les modules correspondants (avec les bonnes options si nécessaire).

5.1 Le patch amanda-conntrack-nat.

Ce patch par Brian J. Murrell <netfilter@interlinx.bc.ca> ajoute le support de suivi de connexions et de nat pour le protocole qu'utilise l'outil de backup Amanda.

5.2 Le patch eggdrop-conntrack

Ce patch par Magnus Sandin <magnus@sandin.cx> ajoute le support de suivi de connexions pour les réseaux de eggdrop bots.

5.3 Le patch h323-conntrack-nat

Ce patch par Jozsef Kadlecsik <kadlec@blackhole.kfki.hu> ajoute le support de suivi de connexions et nat pour H.323/netmeeting.

H.323 utilise les flux de données suivants :

– port 389 -> Internet Locator Server (TCP).
– port 522 -> User Location Server (TCP).
– port 1503 -> T.120 Protocole (TCP).
– port 1720 -> H.323 (H.225 call setup, TCP)
– port 1731 -> Audio call control (TCP)
– Dynamic port -> H.245 call control (TCP)
– Dynamic port -> RTCP/RTP streaming (UDP)

Les modules de suivi de connexions et de NAT pour H.323 suivent et nat les connexions demandées sur des ports dynamiques. Le module utilise le bricolage qui est de chercher/remplacer, qu'utilisait le module ip_masq_h323.c des noyaux 2.2

Au minimum, H.323/netmeeting (vidéo/audio) est fonctionnel en laissant passer les connexions sur le port 1720, et en chargeant ce(s) module(s).

Ces modules ne supportent pas :

– H.245 tunnelling
– H.225 RAS (gatekeepers)

5.4 Le patch irc-conntrack-nat

Ce patch par Harald Welte <laforge@gnumonks.org> permet à DCC de fonctionner à travers une NAT/suivi de connexions. Par défaut, ce patch va suivre les connexions sur le port 6667. Mais vous pouvez faire en sorte que les connexions soient suivies sur un autre port avec le paramètre 'port=xx'.

5.5 Le patch mms-conntrack-nat

Ce patch par Filip Sneppe <filip.sneppe@cronos.be> ahoute le support de suivi de connexions pour le protocole Microsoft Streaming Media Services.

Ceci permet aux clients (Windows Media Player) et au serveur de négocier le protocole (UDP, TCP) et le port que le flux de données utilisera. Une analyse partielle du protocole peut être trouvée

ici `http://get.to/sdp` , avec aussi un lien vers un client Linux.

Il est recommendé d'ouvrir le port UDP 1755, puisque c'est le port qui est utilisé lors des demandes de retransmission.

Ces modules on été testés sans problème dans des mises en place utilisant SNAT et DNAT.

5.6 Le patch pptp

Ce patch par Harald Welte <laforge@gnumonks.org> permet à netfilter de suivre et de NATer les connexions pptp.

5.7 Le patch quake3-conntrack

Ce patch par Filip Sneppe <filip.sneppe@cronos.be> ajoute le support de suivi de connexions et de nat pour Quake III Arena.

5.8 rsh patch

Ce patch par Ian Larry Latter <Ian.Latter@mq.edu.au> ajoute le support pour la suivi de connexions RSH.

Une suivi des connexions RSH deviens nécessaire si la connexion dynamique d'erreur "serveur vers client" est attendue, pendant une sessions RSH normale. Ça ce passe typiquement comme ça :

```
Client 0:1023 --> Serveur 514    (flux 1 - stdin/stdout)
Client 0:1023 <-- Serveur 0:1023 (flux 2 - stderr)
```

L'auteur de ce patch vous averti que ce module peut être dangereux, et que ça n'est pas une bonne solution que d'utiliser RSH, mais surtout vous averti que vous devriez utiliser SSH à la place, autant que possible.

5.9 Le patch snmp-nat

Ce patch par James Morris <jmorris@intercode.com.au> permet à netfilter de NATter le SNMP basic. C'est la forme "basic" du protocole SNMP, comme décrite dans la

RFC 2962 http://www.faqs.org/rfcs/rfc2962.html , ça marche en modifiant l'adresse IP a l'intérieur des paquets SNMP.

5.10 Le patch talk-conntrack-nat

Ce patch par Jozsef Kadlecsik <kadlec@blackhole.kfki.hu> permet à netfilter de suivre les connexions talk, et aussi de les NATter. Par défaut, otalk (UDP port 517) et talk (UDP port 518) sont supportés. Le support pour otalk/talk peut être sélectionné individuellement en passant un paramètre au moment du chargement du module. Les options sont :

- otalk = 0 | 1
- talk = 0 | 1

Où '0' signifie 'pas de support' alors que '1' veut dire 'supporte' le protocole voulu.

5.11 Le patch tcp-window-tracking

Ce patch par Jozsef Kadlecsik <kadlec@blackhole.kfki.hu> permet à netfilter de suivre les connexions TCP en accord avec l'article

Real Stateful TCP Packet Filtering in IP Filter http://www.iae.nl/users/guido/papers/tcp_filtering.ps.gz par Guido van Rooij. Il supporte window scaling, et peut maintenant gérer les connexions déjà établies.

5.12 Le patch tftp

Ce patch par Magnus Boden <mb@ozaba.mine.nu> permet à netfilter de suivre et de NATer les connexions tftp. Par défaut, ce patch va suivre les connexions sur le port 69. Mais vous pouvez faire en sorte que les connexions soient suivies sur un autre port avec le paramètre 'port=xx'.

6 Les nouveaux match IPv6 pour netfilter

Dans cette section, on essayera d'expliquer l'utilisation de ces nouveaux match IPv6 de netfilter. Les patches IPv6 seront abordés par ordre alphabétique. Aussi, nous n'expliquerons pas les patches qui cassent les autres patches. Mais ça pourra venir ensuite.

De manière générale, pour les matches IPv6, vous pouvez voir le petit synopsis de chacun d'eux en tapant :

```
# ip6tables -m le_match_que_vous_voulez --help
```

Ça va afficher le texte d'aide normal d'ip6tables, et le petit synopsis spécifique à "le_match_que_vous_voulez" à la fin.

6.1 Le patch agr

Ce patch par Andras Kis-Szabo <kisza@sch.bme.hu> ajoute un nouveau match :

– "eui64" : vous permet de matcher un paquet IPv6 basé sur ses paramètres d'adressage.

Ce patch peut être très utile pour ceux qui utilisent le schéma d'adressage EUI-64 et qui veulent vérifier les adresses basées sur les adresses délivrées sur le LAN.

Par exemple, si vous voulez rediriger les paquets qui ont une adresse EUI-64 correcte, faites comme suit :

```
# ip6tables -N ipv6ok
# ip6tables -A INPUT -m eui64 -j ipv6ok
# ip6tables -A INPUT -s ! 3FFE:2F00:A0::/64 -j ipv6ok
# ip6tables -A INPUT -j LOG
# ip6tables -A ipv6ok -j ACCEPT

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ipv6ok     all      anywhere             anywhere           eui64
ipv6ok     all     !3ffe:2f00:a0::/64    anywhere
LOG        all      anywhere             anywhere           LOG level warning

Chain ipv6ok (2 references)
target     prot opt source               destination
ACCEPT     all      anywhere             anywhere
```

Ce patch ne prend aucune option.

6.2 Le patch ahesp6

Ce patch par Andras Kis-Szabo <kisza@sch.bme.hu> ajoute un nouveau module qui vous permet de matcher le contenu des en-têtes ah ou esp. Les matches sont :
– "ah" : vous permet de matcher un paquet basé sur son en-tête ah.

– “esp” : vous permet de matcher un paquet basé sur son en-tête esp.
Par exemple, on va DROPper tous les paquets AH qui ont un SPI égal a 500, et vérifier le contenu des champs réservés dans l’en-tête :

```
# ip6tables -A INPUT -m ah --ahspi 500 --ahres -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ah spi:500 reserved
```

Les options supportées par le match ah sont :

–ahspi [!] spi[:spi]
-> Match le SPI indiqué (interval).

–ahlen [!] length
-> Taille de ce header.

–ahres
-> Vérifie les contenu des champs réservés dans l’en-tête.

Le match esp marche exactement de la même manière qu’en IPv4 :

```
# ip6tables -A INPUT -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           esp spi:500
```

Les options supportées pour le match esp sont :

–espspi [!] spi[:spi]
-> match spi (range)

En IPv6 ces matches peuvent être concaténés :

```
# ip6tables -A INPUT -m ah --ahspi 500 --ahres --ahlen ! 40 -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ah spi:500 length:!40 reserved esp spi:500
```

6.3 Le patch frag6

Ce patch by Andras Kis-Szabo <kisza@sch.bme.hu> ajoute un nouveau module ’frag’ qui vous permet de matcher un paquet basé sur le contenu de son en-tête de fragmentation.

Par exemple, on va DROPper tous les paquets qui ont un ID entre 100 et 200, et qui sont le premier fragment d’un flot :

```
# ip6tables -A INPUT -m frag --fragid 100:200 --fragfirst -j DROP

# ip6tables --list
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           frag ids:100:200 first
```

Les options supportées pour le match frag sont :

–fragid [!] id[:id]
-> Match l'ID (interval) de la fragmentation.

–fraglen [!] length
-> Match la longueur totale de l'en-tête.

–fragres
-> Vérifie le contenu des champs réservés.

–fragfirst
-> C'est le premier fragment.

–fragmore
-> Il y a plus de fragments qui arrivent.

–fraglast
-> C'est le dernier fragment.

6.4 Le patch ipv6header

Ce patch par Andras Kis-Szabo <kisza@sch.bme.hu> ajoute un nouveau module qui permet de matcher un paquet par rapport à son extension d'en-tête.

Par exemple, nous allons DROPper les paquets qui ont une en-tête 'hop-by-hop', 'ipv6-route' et 'protocol' :

```
# ip6tables -A INPUT -m ipv6header --header hop-by-hop,ipv6-route,protocol -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ipv6header flags:hop-by-hop,ipv6-route,protocol
```

Et maintenant, nous allons DROPper les paquets qui ont une en-tête 'ipv6-route' :

```
# ip6tables -A INPUT -m ipv6header --header ipv6-route --soft -j DROP

# ip6ptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ipv6header flags:ipv6-route soft
```

Les options supportées pour le match ipv6header sont :

[!] –header headers
-> Sépcifiez l'en-tête à matcher parmi :
– hop,dst,route,frag,auth,esp,none,proto
– hop-by-hop,ipv6-opts,ipv6-route,ipv6-frag,ah,esp,ipv6-nonxt,protocol
– 0,60,43,44,51,50,59

–soft
-> soft-mode : le match véfiriera seulement l'existance de l'en-tête.

6.5 Le patch ipv6-ports

Ce patch par Jan Rekorajski <baggins@pld.org.pl> ajoute 4 nouveaux matches :

– “limit” : vous permet de restreindre le nombre de connexions parallèles faites à partir d’une seule machine ou d’un même réseau.
– “mac” : vous permet de matcher un paquet basé sur son adresse MAC.
– “multiport” : vous permet de spécifier les ports avec un mixe d’intervalles de port ainsi que des ports tout seul pour UDP et TCP.
– “owner” : vous permet de matcher un paquet basé sur le numéro ID du processus qui a généré le paquet.

Ces matches sont un portage de leur équivalent IPv4. Voyez la documentation plus haut ou la page du manuel pour plus de détails !

6.6 Le patch length

Ce patch par Imran Patel <ipatel@crosswinds.net> ajoute un nouveau match qui vous permet de matcher un paquet basé sur sa taille (ce patch est une adaptation sans aucune honte du match length IPv4 écrit par James Morris <jmorris@intercode.com.au>)

Par exemple, si vous voulez DROPper tous les pings qui ont une taille de paquet plus grande que 85 octets :

```
# ip6tables -A INPUT -p ipv6-icmp --icmpv6-type echo-request -m length --length 85:0xffff -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       ipv6-icmp --  anywhere             anywhere           ipv6-icmp echo-request length 85:65535
```

Les options supportées par le match length sont :

[!] –length length[:length]
-> Match la taille du paquet

Tout comme son équivalent IPv4, les valeurs non présentes seront arbitrairement mises à leur valeur par défaut.

6.7 Le patch route6

Ce patch par Andras Kis-Szabo <kisza@sch.bme.hu> ajoute un nouveau match ‘rt’ qui vous permet de matches un paquet basé sur le contenu de son en-tête de routage.

Par exemple, nous allons DROPper tous les paquets qui ont un type de routage 0, qui sont proches du dernier hop (max 2 hops de distance), le chemin de routage contient : :1 et : :2 (mais pas strictement) :

```
# ip6tables -A INPUT -m rt --rt-type 0 --rt-segsleft :2 --rt-0-addrs ::1,::2 --rt-0-not-strict -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           rt type:0 segslefts:0:2 0-addrs ::1,::2 0-not-strict
```

Les options supportées par le match rt sont :

–rt-type [!] type
-> matche le type de routage.

–rt-segsleft [!] num[:num]
 -> Match le champs Segments Left (segments restants, interval).

–rt-len [!] length
 -> longueur totale de l'en-tête.

–rt-0-res
 -> Vérifie le contenu des champs réservés.

–rt-0-addrs ADDR[,ADDR...]
 -> adresses Type=0 (liste, max : 16)

–rt-0-not-strict
 -> Liste d'adresses Type=0, une liste pas stricte.

7 Les nouvelles targets IPv6 pour netfilter.

Dans cette section, on essayera d'expliquer l'utilisation de ces nouvelles targets de netfilter. Les patches seront abordés par ordre alphabétique. Aussi, nous n'expliquerons pas les patches qui cassent les autres patches. Mais ça pourra venir ensuite.

De manière générale, pour les targets, vous pouvez voir le petit synopsis de chacune d'elles en tapant :

```
# ip6tables -j LA_TARGET_QUE_VOUS_VOULEZ --help
```

Ça va afficher le texte d'aide normal d'ip6tables, et le petit synopsis spécifique de "LA_TARGET_QUE_VOUS_VOULEZ" à la fin.

7.1 Le patch LOG

Ce patch par Jan Rekorajski <baggins@pld.org.pl> ajoute une nouvelle target qui vous permet de LOGguer le paquet tout comme son équivalent IPv4.

L'utilisation de cette target est exactement la même que dans iptables, allez donc voir la man page pour plus de détails !

7.2 Le patch REJECT

Ce patch par Harald Welte <laforge@gnumonks.org> ajoute une nouvelle target qui vous permet de rejeter (REJECT) les paquets comme son équivalent IPv4 REJECT de iptables.

Les exemples sont les mêmes que ceux d'iptables IPv4, alors allez voir la man page pour plus de détails !

8 Nouveaux patches de suivi de connexions IPv6 pour netfilter

Le suivi de connexion IPv6 n'est pas supportée pour le moment.

9 Contribuer

9.1 Contribuer une nouvelle extension

L'équipe principale de netfilter est toujours heureuse d'accepter des nouvelles extensions/bug-fixes. Dans cette section nous n'allons pas expliquer comment packager ces extensions pour faciliter leur inclusion dans p-o-m pour le moment. Mais ça pourrait venir dans une version ultérieure de ce HOWTO.

Tout d'abord, vous devriez être familier avec le

Netfilter Hacking HOWTO `http://www.netfilter.org/documentation/HOWTO/fr/netfilter-hacking-HOWTO.html`
.

Rusty a déjà écrit un petit document qui explique dans les grandes lignes comment préparer un patch pour netfilter, c'est dans :

```
/cheming/vers/netfiltercvs/netfilter/patch-o-matic/NEWPATCHES
```

Ou allez lire la dernière version de ce document on-line à l'adresse :

NEWPATCHES `http://cvs.netfilter.org/cgi-bin/cvsweb/netfilter/patch-o-matic/NEWPATCHES` .

Finalement, c'est une bonne idée de souscrire à la liste de diffusion (en anglais) netfilter-devel. Pour plus d'infos sur la démarche à suivre pour s'abonner, allez voir sur la homepage de netfilter.

9.2 Contribuer à ce HOWTO

Vous êtes le bienvenu si vous souhaitez améliorer ce HOWTO. Pour ce faire, le meilleur moyen est d'envoyer un patch du document maître SGML à la liste netfilter-devel. Merci d'avance ! Merci aussi aux développeurs qui ont contribué les parties de ce HOWTO qui étaient en relation avec leurs patches.