Guide Pratique du Filtrage de Paquets sous Linux 2.4

Rusty Russell
Titre original : '*Linux 2.4 Packet Filtering HOWTO*'
Traduction initiale : Emmanuel Roger
Dernière adaptation : Guillaume Audirac`guillaume POINT audirac CHEZ netpratique POINT fr`
Relecture : Thomas Nemeth`tnemeth CHEZ free POINT fr`
v1.26.fr.1.0, le 17 Avril 2004, traduction/adaptation

Ce document décrit comment utiliser Iptables pour filtrer les paquets indésirables, pour les noyaux Linux 2.4.

Table des matières

1 Introduction

Bienvenue, ami lecteur.

On suppose ici que vous savez ce que sont une adresse IP, une adresse réseau, le routage et le DNS. Sinon, je vous recommande de lire le Guide Pratique des Concepts du Réseau sous Linux.

Ce Guide Pratique commence par une gentille introduction (qui va vous laisser bien au chaud pour l'instant, mais sans défense dans le monde réel) avant de basculer vers une douloureuse révélation (qui va tous vous laisser confus, paranoïaques et en recherche d'armes lourdes, hormis les plus hardis).

Votre réseau **n'est pas sécurisé**. Permettre des communications faciles et rapides, en n'autorisant que les communications valides et bien intentionnées, équivaut à un autre problème insoluble, tel que permettre de parler normalement dans un théâtre bondé, tout en empêchant de hurler "Au Feu !". Bien sûr, ce cas ne sera pas résolu dans ce Guide Pratique.

Ainsi, vous seul pouvez décider où se situera le compromis. J'essayerai de vous expliquer l'utilisation de quelques outils disponibles et certaines vulnérabilités auxquelles il faut prêter attention, en espérant que vous les utiliserez avec de bonnes et non de mauvaises intentions. Un autre souci.

2 Où se Trouvent les Sites Web Officiels ? Existe-t'il une Liste de Diffusion ?

Voici les trois sites officiels :
- Merci à *Filewatcher* `http://netfilter.filewatcher.org/` .
- Merci à *l'équipe de samba et SGI* `http://netfilter.samba.org/ter` .
- Merci à *Harald Welte* `http://netfilter.gnumonks.org/` .

Vous pouvez tous les atteindre en utilisant le DNS de type Round-Robin via

http://www.netfilter.org/ et http://www.iptables.org/

Pour la liste de diffusion officielle de Netfilter :

Liste de Netfilter http://www.netfilter.org/contact.html#list .

3 Alors Qu'est-ce Qu'un Filtre à Paquets ?

Un filtre à paquets est un programme qui examine l'*en-tête* des paquets qui passent, et décide du sort du paquet entier. Il peut choisir de **DÉTRUIRE** ('DROP') le paquet (i.e. faire comme s'il n'avait jamais été reçu), d'**ACCEPTER** ('ACCEPT') le paquet (i.e. le laisser passer) ou quelque chose de plus compliqué.

Sous Linux, le filtrage de paquets s'effectue au niveau du noyau (avec un module ou directement intégré dedans), et permet des choses plus astucieuses encore avec les paquets, mais le principe de base reste toujours d'examiner les en-têtes et de décider du sort du paquet.

3.1 Pourquoi Souhaiterais-je Filtrer des Paquets ?

Contrôle. Sécurité. Vigilance.

Contrôle :

quand vous utilisez une machine sous Linux pour connecter votre propre réseau (interne) à un autre réseau (externe, disons Internet), vous avez la possibilité d'autoriser certains types de trafic et d'en interdire d'autres. Par exemple, comme l'en-tête d'un paquet contient son adresse de destination, vous pouvez l'empêcher d'aller vers une certaine partie du réseau externe. Comme autre exemple, j'utilise Netscape pour accéder aux archives de Dilbert. Il y a des publicités de doubleclick.net sur la page, et Netscape me fait perdre du temps à les télécharger. Préciser au filtre de n'autoriser aucun paquet de et vers doubleclick.net résout ce problème (même s'il y a de meilleures façons de faire ça : voir Junkbuster).

Sécurité :

quand votre machine sous Linux est le seul intermédiaire entre le chaos d'Internet et votre beau réseau bien ordonné, il est bon de savoir que vous pouvez en restreindre l'accès suivant ce qui frappe à votre porte. Par exemple, vous pouvez permettre toute connexion vers l'extérieur de votre réseau, mais vous pouvez être embêté par le célèbre 'Ping de la Mort' venant d'étrangers malicieux. Comme autre exemple, vous ne voudriez pas qu'un inconnu se connecte par 'telnet' sur votre machine, même si tous vos comptes sont protégés par un mot de passe. Peut-être voulez-vous (comme la plupart des gens) être un observateur sur Internet et non un serveur (désiré ou non). Pour cela, ne laissez tout simplement personne se connecter chez vous grâce au filtrage de paquets, en refusant les paquets entrants employés pour établir des connexions.

Vigilance :

parfois, une machine mal configurée sur le réseau local décidera d'envoyer des paquets au monde extérieur. Il est intéressant de demander au filtrage de paquets de vous avertir si quelque chose d'inhabituel se produit ; peut-être voulez-vous en tenir compte ou simplement êtes-vous curieux de nature.

3.2 Comment est-ce que je Filtre les Paquets Sous Linux ?

Les noyaux Linux ont connu le filtrage de paquets depuis la série des 1.1. La première génération, basée sur Ipfw de BSD, a été adaptée par Alan Cox fin 1994. Elle a été amélioré par Jos Vos et d'autres pour Linux 2.0 ; l'outil de configuration 'Ipfwadm' contrôlait les règles de filtrage du noyau. À la mi-1998, pour Linux

2.2, j'ai méchamment retravaillé le noyau, avec l'aide de Michael Neuling, et j'ai commencé l'outil appelé 'Ipchains'. Finalement, l'outil de 4ème génération 'Iptables' et une autre réécriture du noyau sont arrivés à la mi-1999 pour Linux 2.4. C'est sur Iptables que se concentre ce Guide Pratique.

Vous avez besoin d'un noyau qui inclut l'infrastructure de Netfilter : Netfilter est un architecture générale à l'intérieur du noyau sur laquelle d'autres choses (comme le module Iptables) viennent se greffer. Cela signifie que vous avez besoin d'un noyau 2.3.15 ou plus, et de répondre 'Y' à CONFIG_NETFILTER lors de la compilation du noyau.

L'outil `Iptables` communique avec le noyau et lui indique les paquets à filtrer. A moins que vous ne soyez un programmeur ou bien excessivement curieux, c'est ainsi que vous contrôlerez le filtrage des paquets.

3.2.1 Iptables

L'outil `Iptables` insère et retire des règles de la table de filtrage des paquets du noyau. Donc peu importe ce que vous configurez, tout sera perdu au redémarrage ; voyez 3.2.2 (Construire des Règles Permanentes) pour vous assurer que vos règles seront rétablies au prochain amorçage.

`Iptables` est un successeur d'`Ipfwadm` et d'`Ipchains` : voyez 8 (Utiliser Ipchains et Ipfwadm) pour éviter de peiner sur Iptables si vous utilisez déjà un des ces deux outils.

3.2.2 Construire des Règles Permanentes

Votre configuration actuelle de pare-feu est stockée dans le noyau, et donc sera perdue au redémarrage. Vous pouvez essayer les scripts 'iptables-save' et 'iptables-restore' pour la sauvegarder et la restituer à partir d'un fichier.

L'autre méthode consiste à mettre les commandes de configuration requises dans un script d'initialisation. Soyez sûr de faire quelque chose d'intelligent si l'une des commandes venait à échouer (habituellement 'exec /sbin/sulogin').

4 Bon Sang, Qui Êtes-Vous et Pourquoi Jouez-Vous avec mon Noyau ?

Je m'appelle Rusty Russel ; je suis le mainteneur du Pare-feu IP pour Linux et juste un autre codeur actif qui s'est trouvé à la bonne place au bon moment. J'ai écrit Ipchains (voir plus haut 3.2 (Comment est-ce que je Filtre les Paquets Sous Linux ?) pour les remerciements mérités aux personnes qui ont fait le vrai travail), et j'en ai appris assez pour effectuer le filtrage de paquets convenablement cette fois. J'espère.

WatchGuard `http://www.watchguard.com` , une excellente entreprise de pare-feux, qui vend l'interface Firebox, m'a gentiment rémunéré à ne rien faire, excepté rédiger ce document et maintenir mes précédents travaux. J'avais prévu 6 mois et ça en a pris 12, mais je pense que cette fois, ça a été bien fait. Ne nombreuses réécritures, un crash-disque, un portable volé, quelques systèmes de fichiers corrompus et un écran détruit plus tard, voilà ce que ça donne.

Pendant que j'y suis, je voudrais éclaircir quelques fausses idées récurrentes : je ne suis pas un gourou du noyau. Je le sais parce que mon travail sur le noyau m'a mis en contact avec certains d'entre-eux : David S. Miller, Alexey Kuznetsov, Andi Kleen, Alan Cox. De toute façon, ils sont tous occupés à faire de la magie en profondeur, me laissant barboter en surface, là où on est en sécurité.

5 Le Guide Rapide de Rusty sur le Filtrage de Paquets

La plupart des gens ont juste une simple connexion PPP à Internet, et ne veulent pas que quelqu'un pénètre sur leur réseau ou leur pare-feu :

```
## Insérer les modules de suivi de connexion (non nécessaire si compilé dans le noyau).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Créer une chaîne qui bloque les nouvelles connexions, sauf celles qui viennent de l'intérieur.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Sauter sur cette chaîne à partir des chaînes INPUT et FORWARD.
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

6 Comment les Paquets Traversent les Filtres

Le noyau contient à la base trois listes de règles dans la table 'filter' ; ces listes sont appelées **chaînes de pare-feu** ou simplement **chaînes**. Les trois chaînes sont nommées **INPUT**, **OUTPUT** et **FORWARD**.

Pour les fans de l'art ASCII, les chaînes sont arrangées comme suit : **(Note : c'est un agencement très différent des noyaux 2.0 et 2.2 !)**

```
                              _____
Entrée                       /     \       Sortie
      --> [Décision] ---->|FORWARD|----->
          [de routage]     \_____/      ^
               |                        |
               v                       ____
              ___                     /    \
             /   \                   |OUTPUT|
            |INPUT|                   \____/
             \___/                      ^
               |                        |
                ---> Processus local ---
```

Les trois cercles représentent les trois chaînes mentionnées ci-dessus. Lorsqu'un paquet atteint un cercle dans le diagramme, cette chaîne est examinée pour décider du sort du paquet. Si la chaîne stipule de DÉTRUIRE ('DROP') le paquet, il est supprimé ici, mais si la chaîne stipule d'ACCEPTER ('ACCEPT') le paquet, il continue sa route dans le diagramme.

Une chaîne est un liste de vérification de **règles**. Chaque règle énonce 'si l'en-tête du paquet est comme ceci, voilà ce qu'il convient de faire du paquet'. Si une règle ne concorde pas avec le paquet, alors la règle suivante est examinée. Finalement, s'il ne reste plus de chaînes à examiner, le noyau consulte **la règle par défaut** de la chaîne pour prendre une décision. Dans un système sécuritaire et consciencieux, cette politique par défaut devrait DÉTRUIRE le paquet.

1. Quand un paquet arrive (supposons par la carte Ethernet), le noyau regarde en premier sa destination : on appelle ceci le 'routage'.

2. S'il est destiné à cette machine, le paquet continue vers le bas du diagramme, vers la chaîne INPUT. S'il la traverse, les processus en attente du paquet le recevront.
3. Autrement, si le noyau n'a pas autorisé la réexpédition ('forwarding') ou s'il ne sait pas comment réexpédier le paquet, celui-ci est détruit. Si la réexpédition est autorisée et que le paquet est destiné à une autre interface réseau (si vous en possédez une autre), il passe directement à la chaîne FORWARD dans le diagramme. S'il est accepté ('ACCEPT'), il sera envoyé.
4. Finalement, un programme en exécution sur la machine peut aussi envoyer des paquets sur le réseau. Ces paquets iront immédiatement vers la chaîne OUTPUT : si elle l'accepte ('ACCEPT'), alors le paquet continuera vers l'interface à laquelle il est destiné.

7 Utiliser Iptables

Iptables bénéficie d'une page de manuel bien détaillée (`man iptables`) si vous avez besoin de détails particuliers. Pour ceux qui sont familiers d'Ipchains, vous devriez simplement aller voir 10 (Différences entre Iptables et Ipchains) car ils sont vraiment similaires.

Différentes choses peuvent être réalisées avec `Iptables`. Pour commencer, travaillons avec les trois chaînes pré-définies `INPUT`, `OUTPUT` et `FORWARD` que vous ne pouvez pas effacer. Et analysons les opérations pour administrer les chaînes :

1. Créer une nouvelle chaîne (-N).
2. Effacer une chaîne vide (-X).
3. Changer la règle par défaut pour une chaîne pré-définie (-P).
4. Lister les règles d'une chaîne (-L).
5. Retirer les règles d'une chaîne (-F).
6. Mettre à zéro les compteurs de bits et de paquets d'une chaîne (-Z).

Il y a plusieurs manières de manipuler une règle à l'intérieur d'une chaîne :

1. Ajouter une nouvelle règle à la chaîne (-A).
2. Insérer une nouvelle règle à une position précise dans la chaîne (-I).
3. Remplacer une règle à une position précise dans la chaîne (-R).
4. Supprimer une règle à une position précise dans la chaîne ou la première qui concorde (-D).

7.1 Ce que Vous Verrez Quand Votre Ordinateur Démarrera

Iptables peut être utilisé sous forme d'un module, appelé ('iptable_filter.o'), qui devrait être automatiquement chargé lorsque vous exécutez la commande `iptables`. Il peut également être compilé de façon permanente dans le noyau.

Avant qu'aucune commande n'ait été exécutée (attention : certaines distributions lanceront Iptables dans leurs scripts d'initialisation), il n'y a de règles dans aucune des chaînes pré-définies ('INPUT', 'FORWARD' et 'OUTPUT') ; toutes les chaînes ont une règle par défaut de type ACCEPT. Vous pouvez modifier la règle par défaut de la chaîne FORWARD en spécifiant l'option 'forward=0' au module iptable_filter.

7.2 Opérations sur une Seule Chaîne

C'est le gagne-pain du filtrage de paquets ; manipuler des règles. Dans la plupart des cas, vous utiliserez les commandes d'ajout (-A) et d'effacement (-D). Les autres (-I pour insérer et -R pour remplacer) sont de simples extensions de ces concepts.

Chaque règle spécifie un ensemble de conditions que le paquet doit remplir et ce qui sera fait si le paquet les remplit (une 'cible'). Par exemple, vous voudriez détruire tous les paquets ICMP qui proviennent de l'adresse IP 127.0.0.1. Dans ce cas, les conditions sont que le protocole est ICMP et que l'adresse source est 127.0.0.1. Notre cible est de type 'DROP'.

127.0.0.1 est l'interface de bouclage ('loopback'), que vous posséderez même si vous n'avez pas de réelle connexion réseau. Vous pouvez utiliser la commande 'ping' pour générer de tels paquets. Elle envoie simplement un ICMP de type 8 (echo request) auquel tous les hôtes coopératifs devraient répondre aimablement avec un ICMP de type 0 (echo reply). En cela, il est très utile pour le test.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Vous pouvez constater que le 1er ping a fonctionné (le '-c 1' signale à ping de n'envoyer qu'un seul paquet).

Ensuite, nous ajoutons (-A) à la chaîne 'INPUT', une règle spécifiant que pour les paquets issus de 127.0.0.1 ('-s 127.0.0.1') avec le protocole ICMP ('-p icmp'), nous devons sauter vers la cible DROP ('-j DROP').

Ensuite, nous testons notre règle en utilisant le second ping. Il y aura une pause avant que le programme ne stoppe parce qu'il attendra une réponse qui ne viendra jamais.

Nous pouvons effacer une règle de deux façons. Premièrement, comme nous savons que c'est la seule règle dans la chaîne INPUT, nous pouvons utiliser un effacement par numérotation :

```
# iptables -D INPUT 1
#
```

pour effacer la règle numéro 1 dans la chaîne INPUT.

La seconde façon correspond au reflet de la commande -A, mais en substituant -A à -D. C'est pratique quand votre chaîne de règles devient complexe et que vous ne souhaitez pas les compter pour découvrir que c'est la règle 37 que vous voulez supprimer. Dans ce cas, on utiliserait :

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

La syntaxe de la commande -D doit avoir exactement les mêmes options que celles de -A (ou -I, ou -R). S'il y a plusieurs règles identiques dans la même chaîne, la première seulement sera effacée.

7.3 Spécifications de Filtrage

Nous avons vu l'utilisation de '-p' pour spécifier le protocole et de '-s' pour spécifier l'adresse source, mais il y a d'autres options que vous pouvez utiliser pour spécifier les caractéristiques d'un paquet. Ce qui suit en est un inventaire exhaustif.

7.3.1 Spécifier les Adresses IP Source et Destination

Les adresses IP source ('-s', '–source' ou '–src') et destination ('-d', '–destination' ou '–dst') peuvent être spécifiées de 4 façons. La plus courante est d'utiliser le nom complet, comme 'localhost' ou 'www.linuxhq.com'. La seconde façon est de spécifier l'adresse IP comme '127.0.0.1'.

Les troisième et quatrième façons permettent de spécifier un groupe d'adresses IP, comme '199.95.207.0/24' ou '199.95.207.0/255.255.255.0'. Toutes deux spécifient toutes les adresses de 199.95.207.0 à 199.95.207.255 inclus ; les chiffres après le '/' précisent la partie signifiante des adresses IP. '/32' ou '/255.255.255.255' correspondent au cas par défaut (soit à toutes les adresses IP). Pour spécifier absolument toutes les adresses IP, '/0' peut être utilisé, comme dans :

```
[ NOTE: '-s 0/0' est ici redondant. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Ceci est rarement utilisé, puisque l'effet de la commande ci-dessus est le même que de ne pas spécifier du tout l'option '-s'.

7.3.2 Spécifier une Inversion

De nombreuses options, dont '-s' (ou '–source') et '-d' (ou '–destination'), peuvent voir leurs arguments précédés de ' !' (indiquant la négation) pour coïncider avec les adresses DIFFÉRENTES de celles indiquées. Par exemple '-s ! localhost' correspond à tout paquet qui **ne provient pas** de localhost.

7.3.3 Spécifier un Protocole

Le protocole peut être désigné avec l'option '-p' (ou '–protocol'). Un protocole est soit un nombre (si vous connaissez les valeurs numériques des protocoles IP), soit un nom pour les cas particuliers comme 'TCP', 'UDP' ou 'ICMP'. La casse n'a pas d'importance, donc 'tcp' marche aussi bien que 'TCP'.

Le nom du protocole peut être préfixé d'un ' !' pour l'inverser, comme '-p ! TCP' pour spécifier les paquets qui **ne sont pas** issus de TCP.

7.3.4 Spécifier une Interface

Les options '-i' (ou '–in-interface') et '-o' (ou '–out-interface') précisent le nom d'une **interface** à laquelle le paquet doit correspondre. Une interface représente le dispositif matériel par lequel le paquet est entré ('-i') ou sortira ('-o'). Grâce à la commande `ifconfig`, vous pouvez lister les interfaces actives (ou 'up').

Les paquets traversant la chaîne `INPUT` n'ont pas encore d'interface de sortie, donc toute règle utilisant '-o' dans cette chaîne ne coïncidera jamais. De la même manière, les paquets traversant la chaîne `OUTPUT` n'ont pas d'interface d'entrée, donc toute règle utilisant '-i' dans cette chaîne ne coïncidera jamais.

Seuls les paquets traversant la chaîne `FORWARD` ont à la fois une interface d'entrée et de sortie.

Il est parfaitement autorisé de spécifier une interface qui n'existe pas à cet instant ; la règle ne coïncidera pas jusqu'à ce que l'interface soit active (ou 'up'). C'est particulièrement utile pour les accès par ligne commutée avec PPP (habituellement l'interface `ppp0`) et ses semblables.

Comme cas particulier, un nom d'interface se terminant par un '+' coïncidera avec toutes les interfaces qui commencent par ce nom (qu'elles existent déjà ou non). Par exemple, pour spécifier une règle qui concorde avec toutes les interfaces PPP, on utilisera l'option `-i ppp+`.

Le nom de l'interface peut être précédé par un ' !' (avec des espaces autour), pour concorder avec un paquet qui **n'appartient pas** à l'interface spécifiée, par exemple `-i ! ppp+`.

7.3.5 Spécifier des Fragments

Parfois, un paquet est trop volumineux pour être transmis en une fois sur la ligne. Quand ça arrive, il est divisé en **fragments** et envoyé en plusieurs paquets. L'autre extrémité réassemble ces fragments pour reconstruire le paquet entier.

Le problème avec les fragments, c'est que le fragment initial possède l'en-tête complet (IP + TCP, UDP et ICMP) à examiner, mais que les paquets suivants ont seulement un morceau de l'en-tête (IP sans les champs additionnels de protocole). Donc il est impossible de rechercher à l'intérieur des fragments suivants les en-têtes de protocoles (comme c'est fait pour les extensions TCP, UDP et ICMP).

Si vous faîtes du traçage de connexion (ou NAT), alors tous les fragments seront fusionnés avant qu'ils n'atteignent le code de filtrage de paquets, par conséquent vous n'aurez pas à vous soucier des fragments.

Attention, notez également que la chaîne INPUT de la table 'filter' (ou toute autre table du 'hook' NF_IP_LOCAL_IN) n'est traversée qu'après défragmentation du coeur de la pile IP.

Autrement, il est important de comprendre comment les fragments sont traités par les règles de filtrage. Toute règle de filtrage qui demande des informations indisponibles *ne concordera pas*. Cela signifie que le premier fragment est traité comme tout autre paquet, mais que le second et les suivants ne le seront pas. Donc une règle `-p TCP -sport www` (qui spécifie le port source 'www') ne coïncidera jamais avec un fragment (autre que le premier fragment). Il en est de même de la règle inverse `-p TCP -sport ! www`.

Cependant, vous pouvez spécifier une règle pour les fragments au-delà du premier, à l'aide de l'option '-f' (ou '–fragment'). Il est aussi valide de spécifier une règle qui *ne s'applique pas* aux fragments autres que le premier, en faisant précéder '-f' de ' !'.

Habituellement, accepter les fragments suivant le premier est considéré comme sûr, parce que le filtrage étant effectif sur le premier fragment, on empêche donc le réassemblage sur la machine cible ; mais des bogues ont été trouvés qui permettent de planter des machines simplement en leur envoyant des fragments. Vous êtes prévenus.

Note pour les experts en réseaux : les paquets malformés (les paquets TCP, UDP et ICMP trop courts pour que le code du pare-feu ne puisse lire les ports, le code ICMP ou son type) sont détruits quand de telles combinaisons sont tentées, comme les fragments TCP qui commencent en position 8.

À titre d'exemple, la règle suivante va détruire tous les fragments à destination de 192.168.1.1 :

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

7.3.6 Extensions à Iptables : Nouvelles Correspondances

`Iptables` est **extensible**, ce qui veut dire que le noyau et le programme Iptables peuvent être étendus pour avoir de nouvelles fonctionnalités.

Quelques-unes de ces extensions sont normalisées, et d'autres sont plus exotiques. Elles peuvent être réalisées par quiconque et distribuées séparément à la demande d'utilisateurs.

Les extensions du noyau se situent normalement dans le sous-répertoire des modules du noyau comme /lib/modules/2.4.0-test10/kernel/net/ipv4/netfilter. Elles sont chargées à la demande si votre noyau a été compilé avec CONFIG_KMOD, donc vous ne devriez pas avoir à les insérer à la main.

Les extensions au programme Iptables sont des bibliothèques partagées qui se situent généralement dans /usr/local/lib/iptables/, bien qu'une distribution puisse les mettre dans /lib/iptables ou /usr/lib/iptables.

Les extensions sont de deux types : nouvelles correspondances et nouvelles cibles (nous aborderons les nouvelles cibles plus tard). Quelques protocoles offrent aussi de nouveaux tests : pour le moment TCP, UDP et ICMP.

Pour ceux-ci, vous pourrez spécifier de nouveaux tests en ligne de commande après l'option '-p', qui chargera implicitement l'extension. Pour spécifier explicitement de nouveaux tests, utilisez l'option '-m' pour charger l'extension, après quoi, ses options seront disponibles.

Pour obtenir de l'aide sur une extension, utilisez l'option pour la charger ('-p', '-j' ou '-m') suivie de '-h' ou '–help', par exemple :

```
# iptables -p tcp --help
#
```

Extensions de TCP Les extensions de TCP sont automatiquement chargées si '-p tcp' est spécifié. Elles offrent les options suivantes (aucune d'entre-elles ne convient aux fragments).

–tcp-flags
: suivi d'un ' !' optionnel, puis 2 chaînes de caractères pour les fanions, vous permet de filtrer sur des fanions TCP spécifiques. La première chaîne correspond au masque : la liste de fanions que vous désirez examiner. La deuxième chaîne précisent lesquels doivent être présents. Par exemple :

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DROP
```

Ceci indique que tous les fanions doivent être examinés ('ALL' est synonyme de 'SYN,ACK,FIN,RST,URG,PSH'), mais seulement SYN et ACK doivent être présents. Il y a aussi l'argument 'NONE' signifiant aucun fanion.

–syn
: précédé optionnellement d'un ' !', c'est un raccourci pour '–tcp-flags SYN,RST,ACK SYN'.

–source-port
: suivi d'un ' !' optionnel, puis soit un port TCP seul ou une plage de ports. Les ports peuvent être des noms, tels qu'ils sont listés dans '/etc/services', ou des nombres. Les plages sont soit 2 noms de ports séparés par ' :' (pour exprimer un intervalle), un port suivi d'un ' :' (pour exprimer supérieur ou égal à), ou un port précédé de ' :' (pour exprimer inférieur ou égal à).

–sport
: est synonyme de '–source-port'.

–destination-port
: et

–dport
: correspondent aux mêmes options qu'au-dessus, si ce n'est qu'elles spécifient le port de destination au lieu du port de source.

–tcp-option
: suivi d'un ' !' optionnel et d'un nombre, correspond à un paquet avec une option TCP égale à ce nombre. Un paquet qui n'a pas un en-tête TCP complet est automatiquement détruit lors d'une tentative pour examiner ses options TCP.

Une Explication des Fanions TCP Il est parfois utile d'autoriser les connexions TCP dans un sens mais pas dans l'autre. Par exemple, vous pourriez vouloir autoriser les connexions vers un serveur WWW externe, mais pas les connexions à partir de ce serveur.

Une approche naïve serait de bloquer les paquets TCP venant du serveur. Malheureusement, les connexions TCP nécessitent des paquets évoluant dans les deux sens pour fonctionner.

La solution est de bloquer seulement les paquets utilisés pour demander une connexion. Ces paquets sont appelés des paquets **SYN** (d'accord, techniquement, ce sont des paquets avec le fanion SYN et pas de fanions FIN et ACK, mais nous les appelons des paquets SYN pour simplifier). En disqualifiant uniquement ces paquets, nous pouvons stopper les tentatives de connexions.

Le fanion '–syn' est utilisé pour cela : il est valide seulement pour les règles qui spécifient TCP comme protocole. Par exemple, pour spécifier une tentative de connexion à partir de l'adresse 192.168.1.1 :

```
-p TCP -s 192.168.1.1 --syn
```

Ce fanion peut être inversé en le faisant précéder de ' !', qui signifie alors tous les paquets sauf ceux d'initiation d'une connexion.

Extensions d'UDP Ces extensions sont automatiquement chargées si '-p udp' est spécifié. Elles procurent les options '–source-port', '–sport', '–destination-port' et '–dport' comme explicité ci-dessus pour le protocole TCP.

Extensions d'ICMP Cette extension est automatiquement chargée si '-p icmp' est spécifié. Elle ne fournit qu'une seule nouvelle option :

–icmp-type

suivi d'un ' !' optionnel, puis un nom de type ICMP (par exemple 'host-unreachable'), ou un type sous forme numérique (par exemple '3'), ou encore un type et un code numériques séparés par un '/' (par exemple '3/3'). Une liste des types ICMP disponibles est fournie avec '-p icmp –help'.

Autres Extensions de Correspondance Les autres extensions du paquetage Netfilter sont des extensions de démonstration qui, une fois installées, peuvent être invoquées avec l'option '-m'.

mac

Ce module doit être spécifié explicitement avec '-m mac' ou '–match mac'. Il est utilisé pour correspondre avec des adresses Ethernet (MAC) de paquets entrants, et il est donc seulement utile pour des paquets traversant les chaînes INPUT et PREROUTING. Il ne propose qu'une seule option :

–mac-source

suivi d'un ' !' optionnel, puis d'une adresse Ethernet en octets (notation hexadécimale) séparés par des ' :', par exemple '–mac-source 00 :60 :08 :91 :CC :B7'.

limit

Ce module doit être spécifié explicitement avec '-m limit' ou '–match limit'. Il est utilisé pour limiter le taux de correspondances, typiquement pour réduire des messages de journalisation (ou 'log'). Il prendra en compte les correspondances seulement un certain nombre de fois pas seconde (par défaut, 3 correspondances par heure, avec une réserve de 5). Il possède deux arguments optionnels :

–limit

suivi d'un nombre, pour spécifier (en moyenne) le nombre maximum de correspondances acceptées par seconde. On peut spécifier son unité explicitement, en utilisant '/second', '/minute', '/hour' ou '/day', ou en abrégé (ainsi '5/second' est identique à '5/s').

–limit-burst
: suivi d'un nombre, indique la réserve maximale (ou la salve) avant que la limite ci-dessus n'entre en jeu.

Cette correspondance peut souvent être employée avec la cible LOG pour limiter les occurences de journalisation. Pour comprendre comment cela fonctionne, examinons la règle suivante, qui journalise les paquets avec les paramètres de limite par défaut :

```
# iptables -A FORWARD -m limit -j LOG
```

La première fois que cette règle est satisfaite, le paquet est journalisé. En fait, avec une réserve de 5, seuls les 5 premiers paquets seront journalisés. Ensuite, 20 minutes doivent passer avant qu'un nouveau paquet ne soit journalisé par cette règle, sans tenir compte du nombre de paquets qui correspondent. Aussi, chaque fois que 20 minutes s'écoulent sans constater de correspondance, un paquet de la réserve est récupéré (sa valeur est incrémentée). Si aucun paquet ne satisfait la règle pendant 100 minutes, la réserve est complètement rechargée et on est revenu au point de départ.

Note : vous ne pouvez actuellement créer de règle avec un temps de recharge de plus de 59 heures, donc si vous configurez un débit moyen de 1 par jour, alors le débit de réserve doit être inférieur à 3.

Vous pouvez aussi utiliser ce module pour éviter les diverses attaques engendrant un déni de service (DoS) et avec un débit supérieur pour augmenter la vitesse de réaction.

Protection contre les inondations de requêtes de connexions ('syn-flood') :

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

Balayage de ports furtif :

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

Ping de la mort :

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Ce module fonctionne comme une porte à hystérésis, comme le montre l'illustration ci-dessous.

```
             Débit (pqts/s)
                   ^        .---.
                   |       / DoS \
                   |      /       \
Limite de DoS -|.....:.........\.......................
 = (limit *    |    /:          \
limit-burst)   |   / :           \         .-.
               |  /  :            \       /   \
               | /   :             \     /     \
Fin de DoS    -|/....:..............:.../.......\..../.
 = limit       |     :              :`-'         `--'
---------------+-----+--------------+------------------> Temps (s)
  LOGIQUE => Concord.| Non concord. |  Concordance
```

On mentionne un paquet par seconde avec une réserve de 5 paquets, mais ici, les paquets commencent à arriver avec un débit de 4/s pendant 3 secondes puis recommencent après 3 autres secondes.

```
        <-Inond. 1 ->           <- Inond. 2  ->
Nb total
  de    ^                Maximum __--      YNNN
paquets|             Débit   __--      YNNN
       |           de    __--      YNNN
    10 |      Ligne __--         Y
       |          __--          Y
       |      __--             Y
```

```
  | __--    YNNN
  |-    YNNN
5 |    Y
  |   Y                               Key:  Y -> La règle concorde
  |  Y                                      N -> La règle ne concorde pas
  | Y
  |Y
0 +-------------------------------------------------->  Temps (secondes)
   0   1   2   3   4   5   6   7   8   9  10  11  12
```

Vous constatez que les 5 premiers paquets sont autorisés à dépasser la limite d'un paquet par seconde, puis la limitation entre en jeu. Si une pause intervient, une autre rafale est autorisée, mais pas au-delà du débit maximum fixée par la règle (1 paquet par seconde après épuisement de la réserve).

owner

Ce module essaie de faire correspondre les diverses caractéristiques du créateur du paquet, pour les paquets générés localement. Il est uniquement valide dans la chaîne OUTPUT, et même dans ce cas, certains paquets sans propriétaire (comme les réponses ICMP d'un ping) ne corresponderont jamais.

–uid-owner id_utilisateur

Concorde si le paquet a été créé par un processus avec l'identifiant d'utilisateur (numérique) indiqué.

–uid-owner id_de_groupe

Concorde si le paquet a été créé par un processus avec l'identifiant de groupe (numérique) indiqué.

–pid-owner id_de_processus

Concorde si le paquet a été créé par un processus avec le numéro de processus indiqué.

–sid-owner id_de_session

Concorde si le paquet a été créé par un processus appartenant au groupe de session indiqué.

unclean

Ce module expérimental doit être spécifié explicitement avec '-m unclean ou '–match unclean'. Il effectue des vérifications diverses et variées sur la bonne constitution des paquets. Ce module n'a pas été vérifié et ne devrait pas être utilisé comme dispositif de sécurité (il pourrait rendre les choses plus dangereuses s'il contient lui-même des bogues). Il ne dispose d'aucune option.

La Correspondance d'État Le critère de correspondance le plus utile est fourni par l'extension 'state', qui interprète l'analyse de traçage de connexion du module 'ip_conntrack'. Il est fortement recommandé.

Spécifier '-m state' permet d'accéder à l'option supplémentaire '–state', qui évalue la correspondance avec une liste d'états séparés par des virgules (les ' !' indiquent les états qui **ne correspondent pas**). Ces états sont :

NEW

Le paquet démarre une nouvelle connexion.

ESTABLISHED

Le paquet est lié à une connexion existante (c'est-à-dire un paquet de réponse ou un paquet envoyé vers l'extérieur appartenant à une connexion qui a déjà répondu).

RELATED

Le paquet est associé à une connexion existante sans faire partie de cette connexion, comme une erreur ICMP ou (avec le module FTP chargé) un paquet établissant une connexion de données FTP.

INVALID

Le paquet ne peut pas être identifié pour une raison quelconque : ceci comprend un manque de mémoire et des erreurs ICMP décorrélées d'une connexion connue. Généralement, ces paquets devraient être détruits.

Un exemple de cette puissante extension de correspondance pourrait être :

```
# iptables -A FORWARD -i ppp0 -m state ! --state NEW -j DROP
```

7.4 Spécifications de Cible

Maintenant que nous connaissons les tests réalisables sur un paquet, nous avons besoin de déterminer ce qu'il convient de faire des paquets sélectionnés par nos tests. C'est ce qu'on appelle la **cible** d'une règle.

Les deux cibles pré-définies sont simples : DROP et ACCEPT. Nous les avons déjà abordées. Si une règle correspond à un paquet et que la cible est une de ces deux-là, aucune autre règle ne sera consultée : le sort du paquet est déjà décidé.

Il existe deux types de cibles différentes des cibles pré-définies : les extensions et les chaînes créées par l'utilisateur.

7.4.1 Les Chaînes Créées Par l'Utilisateur

Une propriété puissante d'`Iptables` (héritée d'`Ipchains`) est la possibilité pour l'utilisateur de créer de nouvelles chaînes, en plus de celles existantes (INPUT, FORWARD et OUTPUT). Par convention, les chaînes utilisateur sont en minuscule pour les différencier (nous décrirons comment créer de nouvelles chaînes utilisateur dans 7.5 (Opérations sur une Chaîne Entière) ci-dessous).

Quand un paquet correspond à une règle dont la cible est une chaîne utilisateur, le paquet commence à traverser les règles de cette chaîne. Si cette chaîne utilisateur ne décide pas du sort du paquet, alors une fois la traversée terminée, le test reprend sur la règle suivante de la chaîne courante.

Le temps est venu pour un peu plus d'art ASCII. Considérons deux chaînes rudimentaires : `INPUT` (la chaîne pré-définie) et `test` (une chaîne définie par l'utilisateur).

```
 `INPUT'                         `test'
----------------------------    ----------------------------
| Règle1: -p ICMP -j DROP  |    | Règle1: -s 192.168.1.1   |
|--------------------------|    |--------------------------|
| Règle2: -p TCP -j test   |    | Règle2: -d 192.168.1.1   |
|--------------------------|    ----------------------------
| Règle3: -p UDP -j DROP   |
----------------------------
```

Considérons un paquet TCP venant de 192.168.1.1 et à destination de 1.2.3.4. Il pénètre dans la chaîne `INPUT`, et est évalué par la Règle1 - il n'y a pas de correspondance. Par contre, la Règle2 concorde et sa cible s'avère être la chaîne `test`, par conséquent, la règle suivante examinée est le début de cette nouvelle chaîne. La Règle1 de `test` concorde mais ne spécifie pas de cible, donc la règle suivante est examinée, soit la Règle2. Elle ne concorde pas et nous avons atteint la fin de la chaîne `test`. Nous retournons donc à la chaîne `INPUT`, dont nous venons d'examiner la Règle2. Nous passons alors à la Règle3, qui ne concorde pas davantage.

On peut schématiser ainsi l'itinéraire du paquet :

```
                        v    __________________________
 `INPUT'                |   /    `test'                v
------------------------|--/    -----------------------|----
```

```
| Règle1                 | /|     | Règle1                |   |
|-----------------------|/-|     |----------------------|---|
| Règle2                /  |     | Règle2                |   |
|--------------------------|     -----------------------v----
| Règle3                /--+___________________________/
------------------------|---
                        v
```

Les chaînes utilisateur peuvent sauter dans d'autres chaînes utilisateur (mais ne faîtes pas de boucle : vos paquets seraient irrémédiablement détruits).

7.4.2 Extensions d'Iptables : Nouvelles Cibles

L'autre type d'extension est une cible. Une extension de cible est constituée d'un module du noyau et d'une extension optionnelle d'`Iptables` pour permettre de nouvelles options en ligne de commande. À l'origine, plusieurs extensions existent déjà dans la distribution de Netfilter :

LOG

Ce module permet de journaliser les paquets qui correspondent. Il propose des options supplémentaires :

–log-level

Suivi d'un nombre de niveaux ou d'un nom. Les noms valides (insensibles à la casse) sont 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' et 'emerg'. Ils correspondent respectivement aux nombres 7 à 0. Lisez la page de manuel de syslog.conf pour connaître leur signification. Le niveau par défaut est 'warning'.

–log-prefix

Suivi d'une chaîne de 29 caractères maximum ; ce message est envoyé au début du message de journalisation, pour rendre son identification unique.

Ce module est surtout utile lors du dépassement d'une limite de correspondance, pour ne pas submerger les journaux.

REJECT

Ce module se comporte comme 'DROP', excepté que l'expéditeur du paquet reçoit en plus un message d'erreur ICMP 'port unreachable'. Notez que le message d'erreur ICMP n'est pas envoyé si (voir RFC 1122) :

- Le paquet filtré est un message d'erreur ICMP ou un type ICMP inconnu.
- Le paquet filtré est un fragment sans en-tête.
- Trop de messages d'erreur ICMP ont été envoyés à cette destination récemment (voir /proc/sys/net/ipv4/icmp_ratelimit).

REJECT peut aussi recevoir une option '–reject-with' qui change le type du paquet de réponse utilisé : consultez la page de manuel.

7.4.3 Cibles Pré-Définies Spéciales

Il y a deux cibles spéciales pré-définies : `RETURN` et `QUEUE`.

`RETURN` a le même effet que d'atteindre la fin d'une chaîne : pour une règle dans une chaîne pré-définie, le comportement par défaut de la chaîne est exécuté ; pour une règle dans une chaîne utilisateur, l'analyse se poursuit dans la chaîne précédente, immédiatement après la règle qui a bifurqué sur cette chaîne utilisateur.

`QUEUE` est une cible spéciale, qui met les paquets en file d'attente pour les traiter dans l'environnement de l'utilisateur. Pour que ça fonctionne, deux éléments supplémentaires sont requis :

– un "arbitre de queue" (ou gestionnaire de file d'attente), qui compose avec les mécanismes de transfert des paquets entre le noyau et l'espace utilisateur ; et
– une application utilisateur, pour recevoir, éventuellement manipuler et établir un verdict sur le sort des paquets.

L'arbitre de queue standard pour Iptables (version IPv4) est le module ip_queue. Il est distribué avec le noyau et signalé comme expérimental.

Voici un exemple rapide de l'utilisation d'Iptables pour mettre des paquets en file d'attente afin de les traiter dans un environnement utilisateur :

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

Avec cette règle, les paquets ICMP sortants générés en local (par exemple créés avec ping) sont transmis au module ip_queue, qui tente ensuite de les délivrer à une application utilisateur. Si aucune application n'est prête à recevoir ces paquets, ils sont détruits.

Pour écrire une application utilisateur, utilisez l'API libipq. Elle est distribuée avec Iptables. Des exemples de code peuvent être trouvés dans les outils "testsuite" (par exemple redirect.c) accessibles via CVS.

Le statut de ip_queue peut être vérifié via :

```
/proc/net/ip_queue
```

La longueur maximale de la file d'attente (c'est-à-dire le nombre maximal de paquets sans verdict délivrés à l'espace utilisateur) peut être contrôlée via :

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

La valeur par défaut de cette longueur maximale est 1024. Une fois cette limite atteinte, les nouveaux paquets seront détruits jusqu'à ce que la longueur de la file redescende sous la limite. Les protocoles bien conçus comme TCP interprètent une destruction de paquets comme une congestion, et se limiteront lorsque la file se remplit. Cependant, des expériences sont nécessaires pour déterminer idéalement la longueur maximale de file d'attente pour une situation donnée, si la valeur par défaut est trop faible.

7.5 Opérations sur une Chaîne Entière

Une propriété utile d'`Iptables` est la possibilité de rassembler des règles apparentées dans des chaînes. Vous pouvez appeler les chaînes comme vous le souhaitez, mais je vous recommande les minuscules pour éviter la confusion avec les chaînes pré-définies et les cibles. Les noms de chaînes peuvent aller jusqu'à 31 caractères maximum.

7.5.1 Créer une Nouvelle Chaîne

Créons une nouvelle chaîne. Comme je suis quelqu'un d'imaginatif, je l'appelle `test`. Nous lui appliquons l'option '-N' ou '–new-chain' :

```
# iptables -N test
#
```

C'est aussi simple que ça. Maintenant, vous pouvez lui ajouter des règles comme expliqué précédemment.

7.5.2 Supprimer une Chaîne

Effacer une chaîne est également très simple, grâce à l'option '-X' ou '–delete-chain'. Pourquoi '-X' ? Et bien, toutes les lettres satisfaisantes étaient déjà prises.

```
# iptables -X test
#
```

Il y a des restrictions à la suppression d'une chaîne : elle doit être vide (voir 7.5.3 (Vider une Chaîne) ci-dessous) et elle ne doit pas être la cible d'une autre règle. Et vous ne pouvez pas supprimer une des trois chaînes pré-définies.

Si vous ne spécifiez pas de chaîne, alors *toutes* les chaînes définies par l'utilisateur seront effacées, si possible.

7.5.3 Vider une Chaîne

Il y a une façon simple de vider une chaîne de toutes ses règles, en utilisant la commande '-F' (ou '–flush').

```
# iptables -F FORWARD
#
```

Si vous ne spécifiez pas de chaîne, alors *toutes* les chaînes seront vidées.

7.5.4 Lister une Chaîne

Vous pouvez lister toutes les règles d'une chaîne en utilisant la commande '-L' (ou '–list').

Le paramètre 'refcnt' listé pour chaque chaîne utilisateur correspond au nombre de règles ayant cette chaîne pour cible. Il doit être à zéro (et la chaîne doit être vide) avant qu'elle ne soit effacée.

Si le nom de la chaîne est omis, toutes les chaînes sont listées, même les chaînes vides.

Trois options peuvent accompagner '-L'. L'option '-n' (numérique) est vraiment utile car elle évite à `Iptables` d'essayer de résoudre les adresses IP, ce qui (si vous utilisez des DNS comme la plupart des gens) génèrera des temps d'attente si votre DNS n'est pas configuré convenablement, ou si vous avez filtré les requêtes DNS. Cette option force aussi l'affichage des ports TCP et UDP avec leur numéro plutôt qu'avec leur nom.

L'option '-v' vous montre tous les détails des règles, tels que les compteurs de paquets et d'octets, les comparaisons de types de service (TOS) et les interfaces. Sinon ces paramètres sont omis.

Notez que les compteurs de paquets et d'octets sont écrits avec les suffixes 'K', 'M' ou 'G', respectivement pour 1.000, 1.000.000 et 1.000.000.000. L'option '-x' (pour 'expand') écrit les nombres intégralement, quelle que soit leur taille.

7.5.5 Initialiser les Compteurs

Il est pratique de pouvoir remettre les compteurs à zéro. Ceci peut être accompli avec l'option '-Z' (ou '–zero').

Considérons les commandes suivantes :

```
# iptables -L FORWARD
# iptables -Z FORWARD
#
```

Dans l'exemple ci-dessus, quelques paquets pourraient traverser, juste entre les commandes '-L' et '-Z'. Pour cette raison, vous pouvez utiliser les commandes '-L' et '-Z' *ensemble*, pour simultanément lister les chaînes et réinitialiser les compteurs.

7.5.6 Configurer le Comportement Par Défaut

Auparavant, quand nous avons abordé le cheminement des paquets à travers les chaînes, nous avons considéré le sort d'un paquet atteignant la fin d'une chaîne pré-définie. Dans un telle situation, le **comportement par défaut** de la chaîne détermine le sort du paquet. Seules les chaînes pré-définies (`INPUT`, `OUTPUT` et `FORWARD`) possèdent un comportement par défaut. En effet, lorsqu'un paquet atteint la fin d'une chaîne utilisateur, la progression reprend à la chaîne précédente.

Le comportement par défaut peut être soit `ACCEPT`, soit `DROP`. Par exemple :

```
# iptables -P FORWARD DROP
#
```

8 Utiliser Ipchains et Ipfwadm

Dans la distribution de Netfilter, il y a des modules appelés ipchains.o et ipfwadm.o. Insérez l'un d'eux dans votre noyau (ATTENTION : ils sont incompatibles avec ip_tables.o !). Ensuite, vous pouvez utiliser Ipchains et Ipfwadm comme jadis.

Ceci sera maintenu pendant encore quelque temps. Je pense qu'une formule raisonnable est 2 * [notification de remplacement - sortie de la version stable initiale], après la date de disponibilité d'une version stable de remplacement. Cela signifie que le support sera probablement abandonné dans Linux 2.6 ou 2.8.

9 Mélanger le NAT et le Filtrage de Paquets

Il est courant de vouloir faire de la Traduction d'Adresses Réseaux ou NAT (voir le Guide Pratique du NAT) et du filtrage de paquets. La bonne nouvelle est qu'ils se mélangent extrêmement bien.

Configurez complètement votre filtrage de paquets en ignorant le NAT que vous réalisez. Les adresses sources et destinations vues par le filtre à paquets seront les adresses 'réelles'. Par exemple, si vous effectuez du DNAT pour envoyer toutes les connexions en direction du port 80 de l'adresse 1.2.3.4, vers le port 8080 de l'adresse 10.1.1.1, le filtre verra les paquets se dirigeant sur le port 8080 de 10.1.1.1 (la destination réelle), et non le port 80 de 1.2.3.4. De la même manière, vous pouvez ignorer le camouflage d'adresses ('masquerading') : les paquets sembleront venir de leurs adresses IP internes réelles (disons 10.1.1.1) et les réponses sembleront retourner là-bas.

Vous pouvez utiliser l'extension de correspondance d'état ('state') sans imposer de travail supplémentaire au filtre à paquets, puisque le NAT requiert de toute façon le traçage de connexions. Pour enrichir l'exemple élémentaire sur le camouflage d'adresses dans le Guide Pratique du NAT (rejetant toute nouvelle connexion issue de l'interface ppp0), vous pourriez faire ceci :

```
# Camoufler ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Rejeter les paquets d'états NEW et INVALID venant de ou réexpédiés par ppp0
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

# Activer la réexpedition d'adresse IP ('forwarding')
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10 Différences Entre Iptables et Ipchains

- Premièrement, les noms des chaînes pré-définies ont changé, passant de minuscules en MAJUSCULES, parce que les chaînes INPUT et OUTPUT récupèrent maintenant seulement les paquets à destination locale et générés en local. Auparavant, elles voyaient respectivement tous les paquets entrants et sortants.
- Le fanion '-i' identifie maintenant l'interface d'entrée, et fonctionne seulement dans les chaînes INPUT et FORWARD. Les règles dans les chaînes FORWARD ou OUTPUT qui utilisaient '-i' doivent être changées en '-o'.
- Les ports TCP et UDP sont maintenant explicités avec les options –source-port ou –sport (ou –destination-port/–dport), et doivent être placés après les options '-p tcp' ou '-p udp', puisqu'ils concernent respectivement les extensions TCP ou UDP.
- Le fanion TCP '-y' est maintenant devenu '–syn', et doit être placé après '-p tcp'.
- La cible 'DENY' est dorénavant appelée 'DROP', finalement.
- Réinitialiser une chaîne en la listant fonctionne.
- Réinitialiser les chaînes pré-définies réinitialise aussi les compteurs du comportement par défaut.
- Lister les chaînes vous renvoie une vue instantanée des compteurs.
- REJECT et LOG sont maintenant des cibles étendues, autrement dit, ce sont des modules du noyau distincts.
- Les noms de chaînes peuvent aller jusqu'à 31 caractères.
- MASQ est à présent devenu MASQUERADE et utilise une syntaxe différente. REDIRECT, tout en gardant le même nom, a aussi subi un changement de syntaxe. Voir le Guide Pratique du NAT pour plus d'informations sur leur configuration.
- L'option '-o' n'est plus utilisée pour diriger les paquets vers l'espace utilisateur (voir '-i' ci-dessus). Les paquets sont maintenant envoyés dans l'espace utilisateur via la cible QUEUE.
- Probablement un tas d'autres choses que j'ai oublié.

11 Quelques Conseils sur la Conception d'un Filtre à Paquets

Dans l'univers de la sécurité informatique, la sagesse élémentaire suggère de tout fermer puis de créer des ouvertures quand c'est nécessaire. On l'exprime habituellement ainsi : 'tout ce qui n'est pas explicitement autorisé est interdit'. Je vous recommande cette approche si la sécurité est votre souci majeur.

Ne faites pas tourner de services dont vous n'avez pas besoin, même si vous pensez avoir bloqué l'accès vers ceux-ci.

Si vous créez un pare-feu dédié, commencez par ne rien faire tourner et bloquer tous les paquets. Ensuite, ajoutez les services et laissez passer les paquets quand c'est nécessaire.

Je suis partisan de la sécurité en profondeur : associez les enveloppeurs de paquets ou 'tcp-wrappers' (pour les connexions au filtre à paquets lui-même), les mandataires ou 'proxies' (pour les connexions traversant le filtre à paquets), la vérification de route et le filtrage de paquets. La vérification de route intervient quand un paquet issu d'une interface inattendue est détruit : par exemple, si votre réseau interne contient des adresses du genre 10.1.1.0/24, et qu'un paquet avec cette adresse source vient sur votre interface externe, il sera détruit. On peut activer ce mode pour une interface (comme ppp0) avec :

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

Ou pour toutes les interfaces présentes et futures avec :

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
```

```
#
```

La distribution Debian fait cela par défaut quand c'est possible. Si vous utilisez un routage asymétrique (c'est-à-dire que vous attendez des paquets venant de directions étranges), vous souhaiterez sûrement désactiver ce filtrage sur ces interfaces.

La journalisation est utile quand vous réalisez un pare-feu, surtout si quelque-chose ne marche pas. Mais sur un pare-feu de production, associez-le toujours avec la correspondance de type 'limit', pour empêcher que quelqu'un ne sature vos journaux.

Je recommande fortement le traçage de connexions sur les systèmes sécurisés : il introduit un peu plus de charge, comme toutes les connexions sont suivies, mais il est très utile pour contrôler l'accès à vos réseaux. Vous devrez charger le module 'ip_conntrack.o' si votre noyau ne charge pas automatiquement les modules et s'il n'est pas déjà compilé dans le noyau. Si vous voulez tracer précisément des protocoles complexes, vous devrez charger le module d'assistance approprié (par exemple 'ip_conntrack_ftp.o').

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet from ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet not from ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Construire un bon pare-feu est au-delà du sujet de ce Guide Pratique, mais suivez mon conseil, **soyez toujours minimaliste**. Consultez le Guide Pratique de la Sécurité pour avoir plus d'informations sur la manière de tester et sonder votre machine.

12 Commentaires et Corrections

Merci de faire parvenir en anglais à l'auteur vos questions et commentaires relatifs à la version originale de ce document à l'adresse *netfilter@lists.samba.org* .

N'hésitez pas à faire parvenir tout commentaire relatif à la version française de ce document à *commentaires CHEZ traduc POINT org* en précisant le titre et la version de ce document.