IPChains et IPTables

BALIN Jean-Victor

Epitech IV

Introduction:

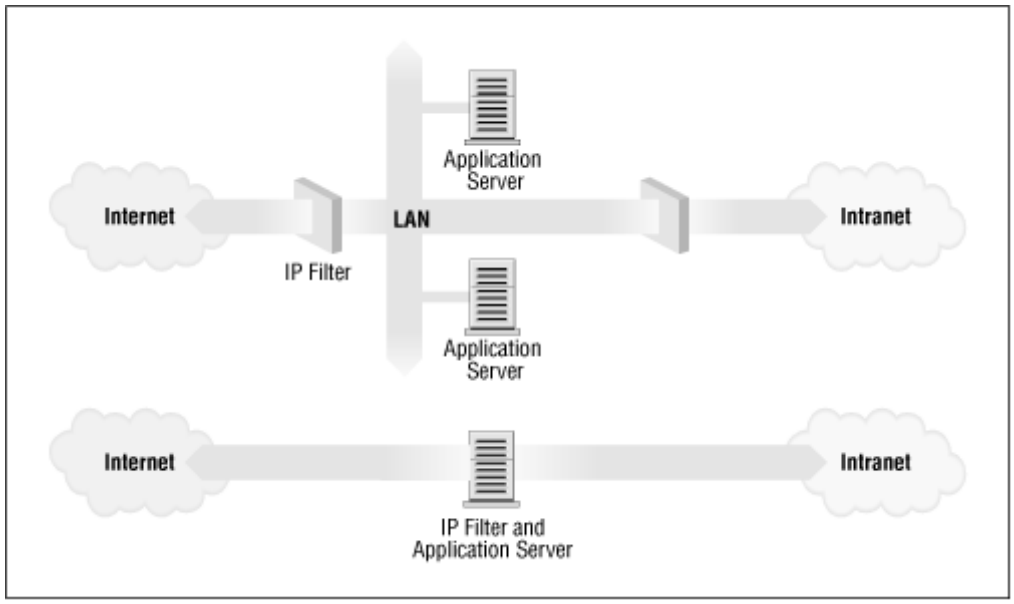
La sécurité des données informatique est devenu indispensable aussi bien dans le milieu de l'entreprise que chez soi. En effet le réseau Internet est devenu indispensable pour toute personne l'ayant accédé vu les innombrables possibilités que cet " outil " peut fournir. Le partage d'informations et la rapidité de divulgation de celles-ci en font une principale source de connaissances inépuisable. Cependant il est totalement utopique de penser que ceci ne peut présenter certains dangers. Ces derniers sont généralement assignés à la perte, la modification, le vol d'informations voir même l'endommagement du matériel sur lesquelles celles-ci sont stockées.

Il est certain qu'une personne mal intentionné ayant un accès total à vos données informatiques peut engendrer des conséquences catastrophiques pour une entreprise. Nous pouvons très bien imaginer les problèmes causés si une société de vente en ligne perdait l'intégralité de sa base de données de produits clients.

C'est pourquoi il est important d'assurer un niveau de sécurité important autour des informations critiques et/ou confidentielles d'un réseau informatique. Il est d'autant plus important de sécuriser ce réseau lorsque celui-ci est connecté au réseau des réseau, Internet.

Afin de trouver des solutions à cette volonté de sécurité de l'information nous allons exposé ici deux types de systèmes de sécurité identiques, du moins dans le principe, que l'on nomme " pare-feu " soit " Firewall " en anglais. Ceux-ci sont principalement utilisé pour filtrer, donc autoriser ou non, la réception ou l'envoie de paquets d'informations d'une machine vers une autre, d'un réseau vers un autre. Les deux utilitaires présentés ici sont " IPChains " et "IPTables ", couramment utilisés dans divers systèmes de production.

Le Firewall ou IP Filter



1 IPChains :

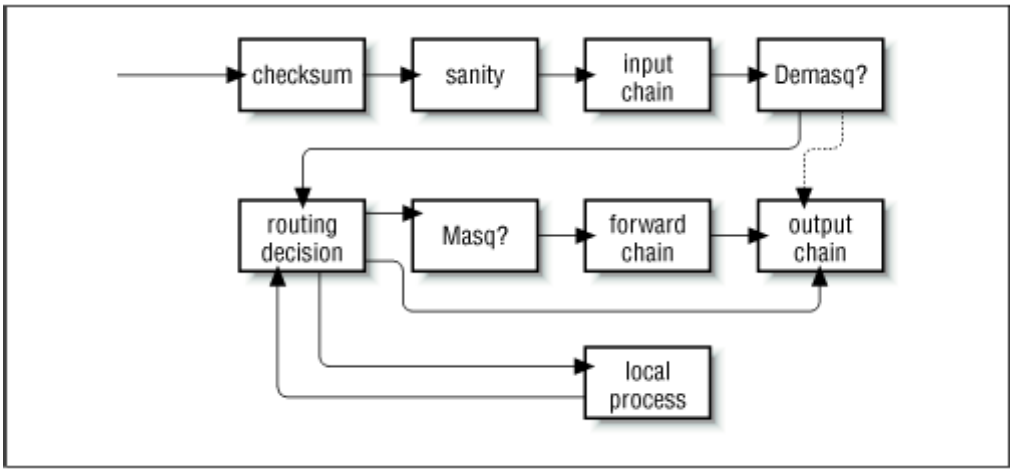
Le noyau 2.2.0 de Linux fit apparaître la troisième génération de "pare-feu IP" sous Linux appelée dès lors "pare-feu IP en chaînes" soit "IP Firewall Chains" en anglais. Ce nouveau principe de protection en chaîne utilise un programme similaire à "ipfwadm" nommé "ipchains".

Tout comme le programme "ipfwadm", "ipchains" apporte une flexibilité d'utilisation similaire à ce premier ajoutant des simplifications syntaxiques lors de la saisie de commandes et la définition de règles de protection. De plus le mécanisme dit "en chaîne" implémenté permet de configurer de multiples règles et de les lier de manière a ce qu'elles puissent interagir ensembles. Tout ceci à pour principal but de simplifier les définitions de règles de protection et de pouvoir définir celles-ci de manière beaucoup plus complexes afin de répondre aux plus grandes exigences des utilisateurs.

Réellement, l'utilitaire "ipchains" est une réécriture du code de firewalling de l'IPv4 de Linux (qui avait été principalement emprunté à BSD) et une réécriture d' "ipfwadm". Cela est principalement dut au fait que l'ancien code de firewalling de Linux ne pouvait gérer les fragments, utilisait des compteurs 32 bits (du moins sur Intel), ne permettait pas la spécification de protocoles autres que TCP, UDP ou ICMP, ne pouvait faire de grands changements atomiquement, ne permettait pas la spécification de règles inverses, le rendant propice aux erreurs des utilisateurs.

Le pare-feu en chaînes "ipchains" fut en premier lieu développé par Paul Russel et Michael Neuling. Ce premier rédigea la documentation de référence pour ce programme nommé "IPCHAINS-HOWTO".

Processus de traitement en chaîne d' "IPChains"



1.1 Mise en œuvre de l'utilitaire "ipchains" :

L'utilitaire "ipchains" insère et efface des règles dans la section de filtrage de paquets du noyau de votre système d'exploitation. C'est pour cela qu'il faut obligatoirement que le noyau de votre système d'exploitation dispose du code de chaînes de protection IP et qu'il soit configuré de façon à ce qu'il soit utilisable.

La configuration du pare-feu est sauvegardée dans le noyau, et sera, donc, perdue lors d'un redémarrage de la machine. Afin d'éviter la redéfinition des règles du pare-feu à chaque démarrage, les scripts "ipchains-save" et "ipchains-restore" sauvegarde et restaure respectivement la configuration du pare-feu afin de rendre les règles définis permanentes. Pour ce faire, configurez vos règles, puis saisissez dans une console en tant que super-utilisateur les commandes suivantes:

Pour sauvegarder la configuration du pare-feu :

```
# /sbin/ipchains-save ¿ /etc/ipchains.rules
#
```

Pour restaurer la configuration du pare-feu :

```
# /sbin/ipchains-restore ¡ /etc/ipchains.rules
#
```

L'utilitaire "ipchains" peut être mis en oeuvre de deux manières différentes. La première consiste à utiliser le script de commandes nomme "ipfwadm-wrapper" substituant principalement les fonctionnalités de l'utilitaire d'ancienne génération "ipfwadm". Cette méthode-ci ne permettra d'exécuter que les règles et commandes disponibles par l'utilitaire "ipfwadm" et n'offrira donc pas les avantages de l'ensemble des nouvelles fonctionnalités du pare-feu IP en chaînes.

La deuxième méthode d'utilisation d' "ipchains" est d'écrire le fichier de configuration du pare-feu en utilisant la nouvelle méthode syntaxique de définition de règles que propose cet utilitaire. De plus la syntaxe d' "ipchains" est beaucoup plus simple que celle utilisée par l'utilitaire "ipfwadm" ce qui facilite l'écriture du fichier de configuration.

L'utilitaire "ipfwadm" utilisait trois listes de règles pour configurer un pare-feu. Avec une protection IP en chaînes il est possible de créer un nombre arbitraire de règles, pouvant être liées, et par delà, interagir ensembles. Celles-ci inclus principalement les trois listes de règles utilisées par l'utilitaire "ipfwadm" à la différence près que celles-ci sont nommés (celle-ci sont aussi couramment appelées "chaînes de protection" ou juste "chaînes") : **input** (entrée), **output** (sortie) et **forward** (transmission).

1.2 Définition de la syntaxe des commandes d' "ipchains" :

Voici la syntaxe et la définition des commandes de l'utilitaire "ipchains":

-N chain : Créer une nouvelle chaîne "chain" spécifiée.

-X [chain] : Supprimer une chaîne "chain" spécifiée ou toutes les chaînes spécifiée si aucune chaîne n'est spécifiée.

-P chain policy : Modifie la police d'une chaîne spécifiée par celle spécifiée.

Remarque :
Les différentes polices de pare-feu sont : ACCEPT, DENY, REJECT, REDIR ou RETURN.

Définition des différentes polices utilisables :

ACCEPT : Accepter la chaîne.

DENY : Ne pas accepter la chaîne.

REJECT : Rejeter la chaîne et génère une réponse ICMP vers la source pour l'informer que la destination n'est pas accessible.

REDIR : Redirige de manière transparente le datagramme vers un port de la machine.

RETURN : La chaîne est traitée par la règle en cours mais n'est pas proposée aux autres règles qui suivent dans la chaîne de protection.

-L [chain] : Lister les règles d'une chaîne "chain" ou liste les règles de toutes les chaînes si aucune chaîne n'est spécifiée.

-F [chain] : Supprimer les règles d'une chaîne "chain" ou toutes les règles si aucune chaîne n'est spécifiée.

-Z [chain] : Mettre à zéro les compteurs de paquets et d'octets sur toutes les règles de la chaîne "chain" spécifiée ou sur toutes les règles si aucune chaîne n'est spécifiée.

-A chain : Ajouter une nouvelle règle à une chaîne "chain" spécifiée.

-I chain rulenum : Insérer une nouvelle règle à la position "rulenum" de la chaîne "chain" spécifiée.

-R chain rulenum : Remplacer une règle à la position "rulenum" de la chaîne "chain" spécifiée.

-D chain rulenum : Supprimer une règle à la position "rulenum" de la chaîne "chain" spécifiée.

-D chain : Supprimer une ou plusieurs règles de la chaîne "chain" spécifiée concordant avec la spécification de la règle.

-C chain : Teste le datagramme identifiée par la spécification de la règle. Cette commande renvoie un message décrivant comment le datagramme fut traité dans la chaîne de protection. Ceci est utile pour tester la configuration du pare-feu.

1.2.1 Spécification des règles :

Chaque règle spécifie un ensemble de conditions que le paquet doit suivre, et ce qu'il faut faire s'il les suit (envoie vers une "destination").

Exemple :

Vous pouvez vouloir refuser tous les paquets ICMP venant de l'adresse IP 127.0.0.1. Donc, dans ce cas les conditions sont que le protocole doit être ICMP et que l'adresse source doit être 127.0.0.1. Notre destination est "DENY" (rejet du paquet).

127.0.0.1 est l'interface "loopback", que vous avez même si vous n'avez de connexion réseau réelle. Vous pouvez utiliser le programme "ping" pour générer de tels paquets (il envoie simplement un paquet ICMP de type 8 (requête d'écho) à qui tous les hôtes coopératifs doivent obligeamment répondre avec un paquet ICMP de type 0 (réponse à un écho)). Ceci le rend utile pour les tests.

```
# 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
# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
# 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 voir ici que le premier ping réussit (le "-c 1" spécifie à l'utilitaire "ping" de n'envoyer qu'un seul paquet).

Puis nous ajoutons (-A) à la chaîne d'entrée ("input"), une règle spécifiant que tous les paquets provenant de 127.0.0.1 ("-s 127.0.0.1") avec le protocole ICMP ("-p ICMP") doivent être refusés ("-j DENY").

Soit :

```
# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
#
```

Testons, dès lors, cette règle, en utilisant un second "ping" identique au premier. Il y aura une pause avant que le programme ne se termine, attendant une réponse qui ne viendra jamais.

Pour supprimer cette règle précédemment définis il y a deux manière de procéder. La première est que, puisque nous savons que c'est la seule règle de la chaîne d'entrée, nous pouvons utiliser une suppression numérotée :

```
# ipchains -D input 1
#
```

Ceci supprime la règle numéro 1 de la chaîne d'entrée.

La deuxième possibilité est de copier la commande –A (ajouter une chaîne), mais en remplaçant le -A par –D (supprimer une chaîne). C'est utile lorsque vous avez une chaîne complexe de règles et que vous ne voulez pas avoir à les compter. Dans ce cas, nous pouvons utiliser :

```
# ipchains -D input -s 127.0.0.1 -p icmp -j DENY
#
```

La syntaxe de -D doit avoir exactement les mêmes options que la commande -A. S'il y a des règles multiples identiques dans la même chaîne, seule la première sera supprimée.

1.2.2 Spécifications exhaustive de filtrage :

Spécification les adresses IP source et destination :

Les adresses IP source (-s) et destination (-d) peuvent être spécifiées de quatre façons différentes. La façon la plus classique est d'utiliser le nom complet de la machine, comme "localhost" ou "www.monhost.com". La deuxième méthode est de spécifier l'adresse IP de la machine, comme par exemple "127.0.0.1".

Les deux autres méthodes permettent la spécification d'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 IP de 192.95.207.0 à 192.95.207.255, incluse ; les chiffres suivant le "/" indiquent quelles parties de l'adresse IP sont significatives. "/32" ou "/255.255.255.255" sont les défauts (vérifient toutes les adresses IP). Pour ne spécifier aucune adresse IP, "/0" peut être utilisé, par exemple :

```
# ipchains -A input -s 0/0 -j DENY
#
```

Ceci est rarement utilisé, car l'effet produit par cette ligne de commande est le même que celui obtenu en ne spécifiant pas l'option "-s".

Spécifier l'inversion :

De nombreuses options, incluant les options "-s" et "-d" peuvent avoir leurs propres arguments précédés par "!" (prononcé "non") pour ne vérifier que les adresses n'étant pas équivalentes à celles données. Par exemple, "-s ! localhost" vérifiera tout paquets ne provenant pas de localhost.

Spécifier le protocole :

Le protocole peut être spécifié en utilisant l'option "-p".

Le protocole peut être :
soit un nombre (si vous connaissez les valeurs numériques des protocoles pour IP).
soit le nom des cas spéciaux parmis "TCP", "UDP" ou "ICMP". La casse n'est pas prise en compte, donc "tcp" fonctionne aussi bien que "TCP".

Le nom du protocole peut être préfixé par un "!", pour l'inverser, comme dans "-p ! TCP".

Spécifier les ports UDP et TCP :

Pour les cas spéciaux où un protocole TCP ou UDP est spécifié, il peut y avoir un argument supplémentaire indiquant le port TCP ou UDP à filtrer, ou un intervalle (inclusif) de ports. Un intervalle est représenté en utilisant le caractère ":", par exemple "6000:6010", qui couvre 11 ports, du 6000 au 6010, de manière inclusive. Si la borne inférieure est omise, alors elle se met par défaut à 0. Si la borne supérieure est omise, elle est considérée par défaut comme étant 65535. Ainsi, pour spécifier les connexions TCP venant des ports inférieurs à 1024, la syntaxe pourrait être "-p TCP -s 0.0.0.0/0 :1023". Les numéros de ports peuvent être spécifiés par leur nom, par exemple "www".

Notez que la spécification du port peut être précédée par un "!", qui l'inverse. Ainsi, pour spécifier tous les paquets TCP, sauf un paquet WWW, vous pourriez spécifier

-p TCP -d 0.0.0.0/0 ! www

Il est important de réaliser que spécifier

-p TCP -d ! 192.168.1.1 www

est très différent de

-p TCP -d 192.168.1.1 ! www

La première ligne spécifie tout paquet TCP dirigé vers le port WWW de n'importe quelle machine sauf 192.168.1.1. La seconde spécifie toute connexion TCP vers tout port de 192.168.1.1 sauf le port WWW.

Enfin, le cas suivant spécifie tout, sauf le port WWW et la machine 192.168.1.1 :

-p TCP -d ! 192.168.1.1 ! www

Spécifier les types et codes ICMP :

ICMP accepte aussi un argument optionnel, mais comme l'ICMP ne dispose pas de ports (ICMP a un type et un code), ils ont une signification différente.

Vous pouvez les spécifier par les noms ICMP (utilisez "ipchains -h icmp" pour lister les noms disponibles) après l'option "-s", ou en tant que type et code ICMP numérique, où le type suit l'option "-s" et le code suit l'option "-d".

Les noms ICMP sont relativement longs : vous avez uniquement besoin de suffisamment de lettres pour rendre chaque nom distinct des autres.

Voici un petit résumé de quelques paquets ICMP les plus communs :

Numéro Nom Utilisé par

0 echo-reply ping

3 destination-unreachable Tout trafic TCP/UDP

5 redirect Routage, si pas de démon de routage

8 echo-request ping

11 time-exceeded traceroute Notez que les noms ICMP ne peuvent être précédés de "!" pour le moment.

Spécifier une interface :

L'option "-i" spécifie le nom d'une **interface** à vérifier. Une interface est le périphérique physique (carte réseau par exemple) d'où vient le paquet, ou bien par où sort ce paquet. Vous pouvez utiliser la commande "ifconfig" pour lister les interfaces qui sont "up", autrement dit qui sont actuellement en fonctionnement.

L'interface pour les paquets entrants, ceux traversant la chaîne d'entrée, est considérée comme étant l'interface d'où les paquets proviennent. Logiquement, l'interface des paquets sortants, ceux traversant la chaîne de sortie, est l'interface vers laquelle les paquets se dirigent. L'interface pour les paquets traversant la chaîne de retransmission est également l'interface par laquelle ils sortiront.

Il est parfaitement autorisé de spécifier une interface qui n'existe pas au moment de la spécification ; la règle ne vérifiera rien jusqu'à ce que l'interface soit mise en place. Ceci est extrêmement utile pour les connexions ppp intermittentes (habituellement les interfaces du type ppp0) et autres.

Cas spécial :

Un nom d'interface se finissant par un "+" vérifiera toutes les interfaces (qu'elles existent à ce moment ou non) qui commencent par cette chaîne.

Par exemple, pour spécifier une règle qui vérifiera toutes les interfaces PPP, l'option -i ppp+ pourrait être utilisée.

Le nom de l'interface peut être précédé par un "!" pour vérifier un paquet qui ne vérifie pas l'(les) interface(s) spécifiée(s).

Spécifier uniquement des paquets TCP SYN :

Il est parfois utile d'autoriser des connexions TCP dans une direction, mais pas dans l'autre. Par exemple, vous pouvez vouloir autoriser les connexions vers un serveur WWW externe, mais pas les connexions venant de ce serveur.

L'approche naïve serait de bloquer les paquets TCP venant du serveur. Malheureusement, les connexions TCP utilisent des paquets circulant dans les deux sens pour fonctionner correctement.

La solution est alors de bloquer uniquement les paquets utilisés pour la demande d'une connexion. Ces paquets sont nommés paquets SYN (techniquement ce sont des paquets avec le drapeau SYN mis, et les drapeaux FIN et ACK supprimés). En interdisant seulement ces paquets, nous pouvons stopper toute demande de connexion entrante.

L'option "-y" est utilisée pour cela : elle est valide seulement pour les règles spécifiant le protocole TCP. Par exemple, pour spécifier une demande de connexion TCP venant de 192.168.1.1 la syntaxe sera :
-p TCP -s 192.168.1.1 -y

Une fois de plus, ce drapeau peut être inversé en le faisant précéder par un "!", qui vérifie tout paquet autre que ceux d'initialisation de connexion (paquets SYN).

Utiliser les fragments :

Si un paquet est trop important en taille alors celui-ci est divisé en fragments, et envoyé en plusieurs paquets. Le receveur réassemble les fragments afin de reconstruire le paquet en entier.

Le problème avec les fragments se situe dans certaines des spécifications listées ci-dessus (en particulier, le port source, le port de destination, le type et le code ICMP, ou le drapeau TCP SYN), qui demandent au noyau de jeter un regard sur le début du paquet, qui est contenu seulement dans le premier fragment.

Si votre machine est la seule connexion vers un réseau extérieur, vous pouvez spécifier au noyau du système d'exploitation de réassembler tous les fragments passant par cette machine, en compilant le noyau avec l'option IP: "always defragment" mise à "Y". Ceci évite proprement la plupart des problèmes.

D'autre part, il est important de comprendre comment les fragments sont traités par les règles de filtrage. Toute règle de filtrage demandant des informations dont nous ne disposons pas ne vérifiera *rien*. Ceci signifie que le premier fragment est traité comme tout autre paquet. Le deuxième fragment et les suivants ne le seront pas.

Ainsi, une règle -p TCP -s 192.168.1.1 www (spécifiant un port source de "www" ne vérifiera jamais un fragment (autre que le premier fragment). La règle opposée -p TCP -s 192.168.1.1 ! www ne fonctionnera pas non plus).

Cependant, vous pouvez spécifier une règle spéciale pour le deuxième fragment et les suivants, en utilisant l'option "-f". Évidemment, il est illégal de spécifier un port TCP ou UDP, un type ou un code ICMP, ou un drapeau TCP SYN dans une règle de fragment puisque ceux-ci ne seront pas présent dans le fragment reçut.

Il est également autorisé de spécifier une règle qui ne s'applique *pas* au deuxième fragment et aux suivants, en plaçant un "!" avant le "-f".

Des attaques connus, permettent de crasher des machines en envoyant de simples fragments. Il est alors préconisé soit de réassembler tout les fragments d'un paquet sur le pare-feu lui-même afin de le soumettre comme les autres paquets aux règles de la chaîne de protection, soit de laisser passer le deuxième fragment et les suivants, puisque le filtrage s'effectuera sur le premier fragment ; Cette deuxième méthode est plus dangereuse car votre réseau est dès lors plus vulnérable puisque vous laissez des paquets non voulus

entrer dans votre réseau.

À noter que les paquets mal formés (TCP, UDP et paquets ICMP trop courts pour que le code pare-feu puisse lire les ports ou les types et codes ICMP) sont également traités comme des fragments. Seuls les fragments TCP débutant en position 8 sont supprimés explicitement par le code pare-feu (un message doit apparaître dans le syslog si cela arrive).

Par exemple:
La règle suivante supprimera tout fragment allant sur 192.168.1.1 :
```
# ipchains -A output -f -d 192.168.1.1 -j DENY
#
```

Spécifier une destination :

Une destination dit au noyau ce qu'il doit faire d'un paquet qui vérifie une règle. "ipchains" utilise "-j", signifiant "jump-to", pour la spécification de la destination. Le nom de la cible doit comporter moins de 8 caractères, et la casse est à respecter : "RETOUR" et "retour" sont totalement différents.

Le cas le plus simple est lorsqu'il n'y a pas de destination spécifiée. Ce type de règle, souvent appelé règle de "comptage", est utile pour compter un certain type de paquet.

Que cette règle soit vérifiée ou non, le noyau examine simplement la règle suivante dans la chaîne. Par exemple, pour compter le nombre de paquets venant de 192.168.1.1, nous pouvons faire ceci :
```
# ipchains -A input -s 192.168.1.1
#
```
(En utilisant "ipchains -L -v" nous pouvons voir les compteurs de paquets et d'octets associés à chaque règle).

Il y a six destinations spéciales. Les trois premières, ACCEPT, REJECT et DENY sont relativement simples. ACCEPT autorise le passage du paquet. DENY supprime le paquet comme s'il n'avait jamais été reçu. REJECT supprime le paquet, mais (si ce n'est pas un paquet ICMP) génère une réponse ICMP vers la source pour lui dire que la destination n'est pas accessible.

La suivante, MASQ dit au noyau de camoufler le paquet. Pour que ceci fonctionne, votre noyau doit être compilé avec le camouflage IP intégré. Cette destination est valide uniquement pour les paquets qui traversent la chaîne forward.

L'autre destination majeure spéciale est REDIRECT demandant au noyau d'envoyer un paquet vers un port local au lieu de celui prévu au départ. Ceci peut être spécifié uniquement pour les règles spécifiant le protocole TCP ou UDP. Optionnellement, un port (nom ou numéro du port) peut être spécifié après "-j REDIRECT" qui redirigera la paquet vers ce port particulier, même si celui-ci était dirigé vers un autre port. Cette destination est valide uniquement pour les paquets traversant la chaîne input.

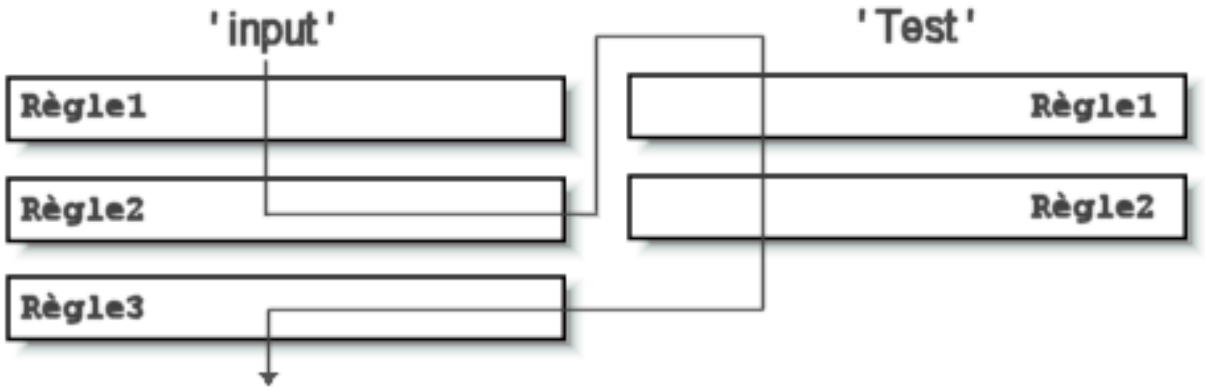
La dernière destination spéciale est la RETURN qui est identique à une terminaison immédiate de la chaîne.

Toute autre destination indique une chaîne définie par l'utilisateur. Le paquet traversera tout d'abord les règles de cette chaîne. Si cette chaîne ne décide pas du destin du paquet, lorsque la traversée de cette chaîne sera achevée, la traversée reprendra sur la règle suivante de la chaîne courante.

Considérons deux chaînes : input (la chaîne intégrée) et Test (une chaîne définie par l'utilisateur).

Considérons un paquet TCP venant de 192.168.1.1, allant vers 1.2.3.4. Il pénètre dans la chaîne input, et est testé par la Règle1 - pas de correspondance. La Règle2 correspond, et sa destination est Test, donc la règle suivante à examiner est le début de Test. La Règle1 de Test correspond, mais ne spécifie pas de destination, donc la règle suivante est examinée soit la Règle2. Elle ne correspond pas, nous avons donc atteint la fin des règles de la chaîne Test. Nous retournons alors à la chaîne input, dont nous avons juste examiné la Règle2, et nous examinons alors la Règle3, qui ne correspond pas non plus.

Le chemin suivi par le paquet est donc le suivant :



Enregistrement des paquets :

C'est un effet de bord que la vérification d'une règle peut avoir ; vous pouvez enregistrer les paquets vérifiés en utilisant l'option "-l". Vous n'aurez généralement pas besoin de ceci pour les paquets habituels, mais ce peut être une option très utile si vous désirez être tenu au courant des évènements exceptionnels.

Le noyau enregistre cette information de la manière suivante :
Packet log: input DENY eth0 PROTO=17 192.168.2.1:53 192.168.1.1:1025
L=34 S=0x00 I=18 F=0x0000 T=254

Ce message d'information est prévu pour être concis, et contient des informations techniques. Il se décompose de la façon suivante :

'input' est la chaîne qui contenait la règle correspondant au paquet, et qui a causé l'apparition du message.

'DENY' est ce que la règle a dit au paquet de faire. Si ceci est un '-' alors la règle n'a pas du tout affecté le paquet (une règle de comptage).

'eth0' est le nom de l'interface. Puisque ceci était la chaîne d'entrée, cela signifie que le paquet vient de 'eth0'.

'PROTO=17' signifie que le paquet était de protocole 17. Une liste des numéros de protocoles est donnée dans '/etc/protocols'. Les protocoles les plus communs sont 1 (ICMP), 6 (TCP) et 17 (UDP).

'192.168.2.1' signifie que l'adresse IP source du paquet était 192.168.2.1.

':53' signifie que le port source était le port 53. En regardant dans '/etc/services' on s'aperçoit que ceci est le port 'domain', autrement dit c'est probablement une réponse DNS. Pour UDP et TCP, ce numéro est le port source. Pour ICMP, c'est le type ICMP. Pour les autres, ce sera 65535.

'192.168.1.1' est l'adresse IP de destination.

':1025' signifie que le port de destination était 1025. Pour UDP et TCP, ce numéro est le port de destination. Pour ICMP, il s'agit du code ICMP. Pour les autres, ce sera 65535.

'L=34' signifie que le paquet avait une longueur totale de 34 octets.

'S=0x00' est le champ "Type Of Service" (divisez par 4 pour obtenir le Type of Service utilisé par "ipchains").

'I=18' est l'identificateur de l'IP.

'F=0x0000' est l'offset du fragment 16 bits, avec les options. Une valeur débutant par '0x4' ou '0x5' signifie que le bit "Don't Fragment" (ne pas fragmenter) est mis. '0x2' ou '0x3' signifie que le bit "More Fragments" (des fragments suivent) est mis ; attendez vous à recevoir plus de fragments après. Le reste du nombre est le décalage de ce fragment, divisé par 8.

'T=254' est la durée de vie du paquet. On soustrait 1 à cette valeur à chaque passage d'une passerelle, et on débute généralement à 15 ou 255.

'(#5)' il peut y avoir un numéro entre parenthèses sur les noyaux Linux les plus récents. Il s'agit du numéro de la règle qui a causé l'enregistrement du paquet.

Manipuler le type de service :

Il y a quatre bits utilisés par eux-mêmes dans l'entête IP, appelés les bits de "Type of Service" (TOS). Ceux-ci ont pour effet de modifier la manière dont les paquets sont traités : les quatres bits sont "Minimum Delay", "Maximum Throughput", "Maximum Reliability" et "Minimum Cost". Un seul de ces bits est autorisé à être placé.

Note : bien entendu, vous n'avez aucun contrôle sur les paquets arrivants ; vous pouvez seulement contrôler la priorité des paquets qui quittent votre machine. Pour négocier les priorités de chaque côté, un protocole comme RSVP doit être utilisé.

L'utilisation la plus commune est de placer les connexions telnet et contrôle du ftp en "Minimum Delay" et les données FTP en "Maximum Throughput". Ceci peut être fait de la manière suivante:

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

L'option "-t" prend deux paramètres supplémentaires, tous les deux en hexadécimal. Ceci permet un contrôle complexe des bits du TOS : le premier masque est ANDé avec le TOS actuel du paquet, et ensuite le deuxième masque est XORé avec lui. Si cela est trop confus, utilisez simplement le tableau suivant :

Nom du TOS	***Valeur***	***Utilisations typiques***
Minimum Delay	0x01 0x10	ftp, telnet
Maximum Throughput	0x01 0x08	ftp-data
Maximum Reliability	0x01 0x04	snmp
Minimum Cost	0x01 0x02	nntp

Ainsi, pour voir les bénéfices maximum des changements de TOS pour les liaisons modem PPP, ajoutez un 'ifconfig $1 txqueuelen' dans votre script /etc/ppp/ip-up. Le nombre à utiliser dépend de la vitesse du modem et de

la taille du tampon du modem ; voici à nouveau les idées d'Andi :

La meilleure valeur pour une configuration donnée nécessite de l'expérimentation. Si les queues sont trop courtes sur le routeur, alors les paquets sauteront. Bien sûr, on peut toujours gagner même sans changer le TOS, c'est juste que le changement du TOS aide à gagner les bénéfices sur les programmes non coopératifs (mais tous les programmes Linux standards sont coopératifs).

1.3 Opérations sur une chaîne entière :

Une des options les plus utiles d' "ipchains" est la possibilité de regrouper des règles dans des chaînes. Vous pouvez appeler les chaînes de la manière qui vous plaît, tant que les noms ne sont pas ceux des chaînes intégrées (input, output et forward) ou des destinations ((MASQ, REDIRECT, ACCEPT, DENY, REJECT ou RETURN). Il est souhaitable d'éviter les noms en majuscules, de plus le nom de la chaîne ne doit pas dépasser 8 caractères.

1.3.1 Créer une nouvelle chaîne :

Créons une nouvelle chaîne. Étant un type imaginatif, je l'appellerai test.

```
# ipchains -N test
#
```

Maintenant vous pouvez y rajouter des règles, comme détaillé ci-dessus.

1.3.2 Vider une chaîne :

Il y a un moyen simple de vider toutes les règles d'une chaîne, en utilisant la commande "-F".

```
# ipchains -F forward
#
```

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

1.3.3 Supprimer une chaîne :

La suppression d'une chaîne est tout aussi simple.

```
# ipchains -X test
#
```

Il y a quelques restrictions à la suppression des chaînes : elles doivent être vides et elles ne doivent pas être la destination d'une quelconque règle. Vous ne pouvez pas supprimer les chaînes intégrées.

1.3.4 Afficher une chaîne :

Vous pouvez afficher toutes les règles d'une chaîne en utilisant la commande "-L".

```
# ipchains -L input
Chain input (refcnt = 1): (policy ACCEPT)
Target prot opt source destination ports
ACCEPT icmp —— anywhere anywhere any
# ipchains -L test
Chain test (refcnt = 0):
Target prot opt source destination ports
DENY icmp —— localnet/24 anywhere any
#
```

La "refcnt" listée pour test est le nombre de règles qui ont test comme destination. Ceci doit être égal à zéro (et la chaîne doit être vide) avant que cette chaîne ne puisse être supprimée.

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

Il y a trois options qui peuvent accompagner "-L". L'option "-n" (numérique) est très utile et empêche ipchains d'essayer de vérifier les adresses IP, ce qui (si vous utilisez un DNS) causera de longs délais si votre DNS n'est pas configuré proprement, ou si vous filtrez les requêtes DNS. Ceci affichera également les

ports par leur numéro plutôt que par leur nom.

L'option "-v" vous montrera tous les détails des règles, comme les compteurs de paquets et d'octets, les masques de TOS, l'interface, et les marques de paquets. Autrement, ces valeurs seront omises. Par exemple :

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
Pkts bytes target prot opt tosa tosx ifname mark source destination ports
10 840 ACCEPT icmp ---- 0xFF 0x00 lo anywhere anywhere any
```

Notez que les compteurs de paquets et d'octets sont affichés en utilisant les suffixes "K", "M" ou "G" pour 1000, 1.000.000 et 1.000.000.000, respectivement. En utilisant également l'option "-x" (développe les nombres), ipchains affichera les nombres en entier, quelques soit leur taille.

1.3.5 Remise à zéro des compteurs

Il est parfois utile de pouvoir remettre à zéro les compteurs. Ceci peut être fait par l'option "-Z" (compteurs mis à Zéro). Par exemple :

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
pkts bytes target prot opt tosa tosx ifname mark source destination ports
10 840 ACCEPT icmp ---- 0xFF 0x00 lo anywhere anywhere any
# ipchains -Z input
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
pkts bytes target prot opt tosa tosx ifname mark source destination ports
0 0 ACCEPT icmp ---- 0xFF 0x00 lo anywhere anywhere any
#
```

Le problème de cette approche est que parfois vous voudrez connaître les valeurs du compteur tout de suite après la remise à zéro. Dans l'exemple ci-dessus, quelques paquets peuvent passer entre les commandes "-L" et "-Z". Pour cette raison, vous pouvez utiliser le "-L" et le "-Z" ensemble, pour remettre à zéro les compteurs tout en les lisant. Malheureusement, si vous

faîtes ceci, vous ne pouvez opérer sur une seule chaîne : vous devrez lister et remettre à zéro toutes les chaînes en une seule fois.

```
# ipchains -L -v -Z
Chain input (policy ACCEPT):
pkts bytes target prot opt tosa tosx ifname mark source destination ports
10 840 ACCEPT icmp ---- 0xFF 0x00 lo anywhere anywhere any
Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
0 0 DENY icmp ---- 0xFF 0x00 ppp0 localnet/24 anywhere any
# ipchains -L -v
Chain input (policy ACCEPT):
pkts bytes target prot opt tosa tosx ifname mark source destination ports
10 840 ACCEPT icmp ---- 0xFF 0x00 lo anywhere anywhere any
Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
0 0 DENY icmp ---- 0xFF 0x00 ppp0 localnet/24 anywhere any
#
```

1.3.6 Choisir une police :

La police de la chaîne détermine la destination du paquet. Seules les chaînes intégrées (input, output et forward) ont des polices, car si un paquet atteint la fin d'une chaîne définie par l'utilisateur, le paquet revient sur la chaîne précédente.

La police peut être une des quatre premières destinations spéciales : ACCEPT, DENY, REJECT ou MASQ. MASQ est la seule destination valide pour la chaîne "forward".

Il est également important de noter qu'une destination RETURN dans une règle de l'une des chaînes intégrées est utile pour expliciter la destination de la chaîne lorsqu'un paquet correspond à la règle.

1.3.7 Opérations sur le camouflage :

Il y a plusieurs paramètres que vous pouvez modifier pour le camouflage IP. Ils sont intégrés avec "ipchains" car il n'est pas évident d'écrire un outil séparé pour eux.

La commande du camouflage IP est "-M", et peut être combinée avec "-L" pour lister les connexions actuellement camouflées, ou avec "-S" pour configurer les paramètres du camouflage.

La commande "-L" peut être accompagnée par "-n" (montre des nombres à la place des noms de machines et de ports) ou "-v" (affiche les deltas dans les séquences de nombres pour la connexion camouflée).

La commande "-S" doit être suivie par trois valeurs de fin d'attente, toutes en secondes : pour les sessions TCP, pour les sessions TCP précédées d'un paquet FIN, et pour les paquets UDP. Si vous ne voulez pas changer l'une de ces valeurs, donnez-lui simplement une valeur de "0".

1.3.8 Vérifier un paquet :

Parfois, vous voulez savoir ce qui arrive lorsqu'un certain paquet entre dans votre machine, par exemple pour déboguer votre chaîne pare-feu. L'utilitaire "ipchains" dispose de la commande "-C" pour autoriser cela, en utilisant exactement les mêmes routines que celles que le noyau utilise pour vérifier les vrais paquets.

Pour spécifier sur quelle chaîne le paquet doit être testé mettez son nom après l'argument "-C". Même si le noyau commence toujours par traverser les chaînes input, output ou forward, vous êtes autorisé à commencer en traversant n'importe quelle chaîne pour tester.

Les détails du "paquet" sont spécifiés en utilisant la même syntaxe que celle utilisée pour spécifier les règles pare-feu. En particulier, un protocole ("-p"), une adresse source ("-s"), une adresse de destination ("-d") et une interface ("-i") sont nécessaires. Si le protocole est TCP ou UDP, alors une

source unique et une destination unique doivent être spécifiées, et un type et un code ICMP doivent être spécifiés pour le protocole ICMP (à moins que l'option "-f" soit spécifiée pour indiquer une règle de fragment, auquel cas ces options sont illégales).

Si le protocole est TCP (et que l'option "-f" n'est pas spécifiée), l'option "-y' doit être explicitée, afin d'indiquer que le paquet test doit être configuré avec le bit SYN.

Voici un exemple de test d'un paquet TCP SYN venant de 192.168.1.1, port 60000, et allant sur 192.168.1.2, port www, arrivant de l'interface eth0 et entrant dans la chaîne "input" (il s'agit d'une initialisation classique d'une connexion WWW) :

```
# ipchains -C input -p tcp -y -i eth0 -s 192.168.1.1 60000 -d 192.168.1.2
www
packet accepted
#
```

Voir ce qui arrive avec des règles multiples précisées en une seule fois :

Parfois, une simple ligne de commande peut affecter de multiples règles. Ceci se fait par deux méthodes.

Premièrement, si vous spécifiez un nom de machine qui correspond (en utilisant DNS) à de multiples adresses IP, ipchains agira comme si vous aviez tapé de multiples commandes avec chaque combinaison d'adresses.

Ainsi, si le nom de machine "www.foo.com" correspond à trois adresses IP, et si le nom de machine "www.bar.com" correspond à deux adresses IP, alors la commande "ipchains -A input -j reject -s www.bar.com -d www.foo.com" ajoutera six règles à la chaîne input.

L'autre méthode pour avoir "ipchains" réalisant de multiples actions est d'utiliser l'option bidirectionnelle ("-b"). Cette option fait agir "ipchains" comme si vous aviez tapé la commande deux fois, la deuxième fois avec les arguments "-s" et "-d" inversés. Ainsi, pour éviter la transmission soit de, soit vers 192.168.1.1, vous pourriez utiliser la commande :

```
# ipchains -b -A forward -j reject -s 192.168.1.1
#
```

L'option -b peut être utilisée avec les commandes insérer ("-I"), supprimer ("-D") (mais pas avec la variation qui prend un numéro de règle), ajouter ("-A") et vérifier ("-C").

Une autre option utile est "-v" qui imprime exactement ce que "ipchains" fait avec vos commandes. Ceci est utile si vous traitez avec des commandes qui peuvent affecter de multiples règles.

Par exemple, nous devons ici vérifier le comportement de fragments entre 192.168.1.1 et 192.168.1.2.

```
# ipchains -v -b -C input -p tcp -f -s 192.168.1.1 -d 192.168.1.2 -i lo
tcp opt —f- tos 0xFF 0x00 via lo 192.168.1.1 -¿ 192.168.1.2 * -¿ *
packet accepted
tcp opt —f- tos 0xFF 0x00 via lo 192.168.1.2 -¿ 192.168.1.1 * -¿ *
packet accepted
#
```

1.4 Exemples utiles :

J'ai une connexion intermittente en PPP (-i ppp0). Je récupère les news (-p TCP -s news.virtual.net.au nntp) et le courrier (-p TCP -s mail.virtual.net.au pop-3) à chaque fois que je me connecte. J'utilise la méthode ftp de Debian pour mettre ma machine à jour régulièrement (-p TCP -y -s ftp.debian.org.au ftp-data). Je visionne le web au travers du proxy de mon FAI (Fournisseur d'Accès Internet) lorsque je suis en ligne (-p TCP -d proxy.virtual.net.au 8080), mais je déteste les publicités de doubleclick.net des Archives de Dilbert (-p TCP -y -d 199.95.207.0/24 et -p TCP -y -d 199.95.208.0/24).

J'autorise les gens à essayer le ftp sur ma machine lorsque je suis en ligne (-p TCP -d $LOCALIP ftp), mais je n'autorise personne de l'extérieur à prétendre avoir une adresse IP sur mon réseau interne (-s 192.168.1.0/24). Ceci est communément appelé IP spoofing.

Cette configuration est relativement simple, car il n'y a pour l'instant aucune autre machine sur mon réseau interne.

Je ne veux pas que des processus locaux (ex : Netscape, lynx, etc.) se connectent à doubleclick.net :

```
# ipchains -A output -d 199.95.207.0/24 -j REJECT
# ipchains -A output -d 199.95.208.0/24 -j REJECT
#
```

Maintenant je désire changer les priorités des divers paquets sortants (il n'y a pas vraiment d'intérêt à le faire pour les paquets entrants). Puisqu'il y a un certain nombre de ces règles, il y a intérêt à les mettre ensemble dans une seule chaîne, nommée ppp-out.

```
# ipchains -N ppp-out
# ipchains -A output -i ppp0 -j ppp-out
#
```

Délai minimal pour le trafic web et telnet :

```
# ipchains -A ppp-out -p TCP -d proxy.virtual.net.au 8080 -t 0x01 0x10
# ipchains -A ppp-out -p TCP -d 0.0.0.0 telnet -t 0x01 0x10
#
```

Coût faible pour les données ftp, nntp et pop-3 :

```
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 ftp-data -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 nntp -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 pop-3 -t 0x01 0x02
#
```

Il y a quelques restrictions sur les paquets venant de l'interface ppp0 ; créons une chaîne nommée "ppp-in" :

```
# ipchains -N ppp-in
# ipchains -A input -i ppp0 -j ppp-in
#
```

Maintenant, aucun des paquets venant de ppp0 ne doit prétendre avoir une adresse source de la forme 192.168.1.*, donc nous les enregistrons et les interdisons :

```
# ipchains -A ppp-in -s 192.168.1.0/24 -l -j DENY
#
```

J'autorise l'entrée des paquets UDP pour le DNS (je fais tourner un serveur de nom cache qui renvoie toutes les demandes sur 203.29.16.1, donc je m'attend à des réponses DNS venant uniquement de là), l'entrée du ftp, et le retour des données ftp (ftp-data) uniquement (ce qui doit être uniquement entre un port strictement supérieur à 1023, et pas sur les ports X11 autour de 6000).

```
# ipchains -A ppp-in -p UDP -s 203.29.16.1 -d $LOCALIP dns -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 1024:5999
-j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 6010: -j
ACCEPT
# ipchains -A ppp-in -p TCP -d $LOCALIP ftp -j ACCEPT
#
```

Enfin, les paquets local vers local sont acceptés:

```
# ipchains -A input -i lo -j ACCEPT
#
```

Maintenant, ma police par défaut sur la chaîne input est DENY, ce qui fait que tout le reste disparaît :

```
# ipchains -P input DENY
#
```

NOTE : Il est préférable de ne pas configurer ces chaînes dans cet ordre, car des paquets pourraient passer durant la configuration. Le moyen le plus sûr est généralement de configurer la police à DENY tout d'abord, et ensuite d'insérer les règles. Bien sûr, si vos règles ont besoin d'effectuer des demandes DNS pour résoudre des noms de machines, vous pourriez avoir des problèmes.

1.5 Utiliser le script de commande

"**ipchains-save**" **:** Comme toute la configuration du pare-feu est envoyée dans le noyau de votre système d'exploitation, celle-ci sera effacée lors d'un redémarrage de la machine.

Donc, le script de commande "ipchains-save" permet de lire votre configuration actuelle des chaînes et de la sauvegarder dans un fichier.

Le script de commande "ipchains-save" peut sauvegarder une chaîne seule, ou toutes les chaînes (si aucun nom de chaîne n'a été spécifié). La seule option actuellement autorisée est le "-v" qui affiche les règles (vers stderr) lorsqu'elles sont sauvées. La police de la chaîne est aussi sauvée pour les chaînes input, output et forward.

```
# ipchains-save ¿ my_firewall
Saving 'input'.
Saving 'output'.
Saving 'forward'.
Saving 'ppp-in'.
Saving 'ppp-out'.
#
```

1.6 Utiliser le script de commande

"**ipchains-restore**" : Le script de commande "ipchains-restore" restaure les chaînes que vous avez sauvegardées avec "ipchains-save". Il peut prendre deux options : "-v" qui décrit chaque règle lorsqu'elle est ajoutée, et "-f" qui force le nettoyage des chaînes définies par l'utilisateur si elles existent, comme décrit plus bas.

Si une chaîne définie par l'utilisateur est trouvée à l'entrée, "ipchains-restore" vérifie que cette chaîne existe déjà. Si elle existe, alors vous serez interrogé pour savoir si la chaîne doit être nettoyée (suppression de toutes les règles) ou si la restauration de la chaîne doit être ignorée. Si vous spécifiez "-f" sur la ligne de commande, vous ne serez pas interrogé ; la chaîne sera écrasée.

Par exemple :

```
# ipchains-restore ¡ my_firewall
Restoring 'input'.
Restoring 'output'.
Restoring 'forward'.
Restoring 'ppp-in'.
Chain 'ppp-in' already exists. Skip or flush? [S/f]? s
Skipping 'ppp-in'.
Restoring 'ppp-out'.
Chain 'ppp-out' already exists. Skip or flush? [S/f]? f
Flushing 'ppp-out'.
#
```

Comment organiser vos règles pare-feu :

Vous pouvez tenter de les organiser pour optimiser la vitesse (minimiser le nombre de vérifications de règles pour la plupart des paquets) ou pour augmenter la maniabilité.

Si vous avez une connexion intermittente, disons une connexion PPP, vous pouvez configurer la première règle de la chaîne d'entrée pour être '-i ppp0 -j DENY' au lancement, puis avoir quelque chose comme ceci dans votre script de commandes "ip-up" :

```
# Re-crée la chaîne "ppp-in"
```

```
ipchains-restore -f ¡ ppp-in.firewall

# Remplace la règle DENY par un saut vers la chaîne se chargeant du ppp
ipchains -R input 1 -i ppp0 -j ppp-in
```

Votre script ip-down pourrait ressembler à ça :
ipchains -R input 1 -i ppp0 -j DENY

1.7 Ce qu'il ne faut pas filtrer :

Il y a un certain nombre de choses auxquelles vous devez faire attention avant de commencer à filtrer quelque chose que vous n'auriez pas voulu filtrer.

1.7.1 Les paquets ICMP :

Les paquets ICMP sont utilisés (entre autres choses) pour indiquer des problèmes aux autres protocoles (comme TCP et UDP). Les paquets "destination-unreachable" (destination non accessible) en particulier. Le blocage de ces paquets signifie que vous n'obtiendrez jamais les erreurs "Host unreachable" ou "No route to host" ; toute connexion attendra une réponse qui ne viendra jamais.

Un problème plus inquiétant est le rôle des paquets ICMP dans la découverte MTU. Toutes les bonnes implémentations de TCP (y compris celle de Linux) utilisent la recherche MTU pour tenter de trouver quel est le plus grand paquet qui peut atteindre une destination sans être fragmenté (la fragmentation diminue les performances, principalement lorsque des fragments occasionnels sont perdus). La recherche MTU fonctionne en envoyant des paquets avec le bit "Don't Fragment" activé, et en envoyant ensuite des paquets plus petits s'il reçoit un paquet ICMP indiquant "Fragmentation needed but DF set" ('fragmentation-needed'). C'est un paquet de type "destination-unreachable", et s'il n'est jamais reçu, l'hôte local ne réduira pas le MTU, et les performances seront abyssales ou inexistantes.

Notez qu'il est commun de bloquer tous les messages ICMP de redirection, soit ceux de type 5; ils peuvent être utilisés pour manipuler le routage (bien que les piles IP bien conçues disposent de gardes-fou), et sont donc souvent considérés comme quelques peu risqués.

1.7.2 Connexions TCP au DNS (serveur de nom) :

Si vous tentez de bloquer toutes les connexions TCP sortantes, rappelez-vous que le DNS n'utilise pas toujours UDP ; si la réponse du serveur dépasse les 512 octets, le client utilise une connexion TCP (allant toujours sur le port numéro 53) pour obtenir les données.

Ceci peut être un piège car le DNS fonctionnera partiellement si vous interdisez de tels transferts TCP ; vous pouvez expérimenter des délais longs et étranges, et d'autres problèmes liés au DNS si vous le faites.

Si les demandes de votre DNS sont toujours dirigées vers les mêmes sources externes (soit directement en utilisant la ligne "nameserver" dans "/etc/resolv.conf" ou en utilisant un serveur de noms cache en mode de redirection), alors vous n'aurez besoin d'autoriser que les connexions du port "domain" sur ce serveur de nom à partir du port local "domain" (si vous utilisez un serveur de nom cache) ou d'un port élevé (¿ 1023) si vous utilisez "/etc/resolv.conf".

1.7.3 Cauchemars du FTP :

L'autre problème classique du filtrage de paquets est celui posé par le transfert de fichiers par le protocole FTP.

Le FTP a deux modes : le mode traditionnel est appelé mode actif et le plus récent est appelé mode passif. Les navigateurs web utilisent souvent le mode passif par défaut, mais les programmes de FTP en ligne de commande utilisent en général par défaut le mode actif.

En mode actif, lorsque l'hôte distant désire envoyer un fichier (ou même les résultats d'une commande tel que "ls" ou "dir"), il essaye d'ouvrir une connexion TCP sur la machine locale. Cela signifie que vous ne pouvez filtrez ces connexions TCP sans supprimer le FTP actif.

Si vous avez comme option l'utilisation du mode passif, alors tout va bien ; le mode passif fait passer les connexions de données du client au serveur, même pour les données arrivantes. Autrement, il est recommendé de n'autoriser que les connexions TCP vers les ports supérieurs à 1024 et de les interdire entre 6000 et 6010 (6000 est utilisé par X-Windows).

1.7.4 Filtrer le ping de la mort (Ping of Death) :

Les machines Linux sont maintenant immunisées du fameux "Ping of Death", qui implique l'envoi de paquets ICMP illégalement grands faisant déborder les espaces mémoire de la pile TCP du récepteur causant de gros dégâts tel le crash de la machine.

Si vous voulez protéger des machines qui peuvent être vulnérables, vos pouvez simplement bloquer les fragments ICMP. Les paquets ICMP normaux ne sont pas assez gros pour nécessiter la fragmentation, et vous ne casserez rien à part les gros pings.

Même si tous les programmes exploitant cette erreur utilisent l'ICMP, il n'y a pas de raisons qu'un fragment TCP ou UDP (ou d'un protocole inconnu) ne puisse être utilisé pour cette attaque, donc le blocage des fragments ICMP est seulement une solution temporaire.

1.7.5 Filtrer Teardrop et Bonk :

Teardrop et Bonk sont deux attaques (principalement contre les machines sous Microsoft Windows NT) qui reposent sur des fragments superposés. Afin de prévenir de telles attaques il est recommandé d'avoir un routeur défragmentant les paquets, ou alors d'interdire tous les fragments vers vos

machines vulnérables.

1.7.6 Filtrer les bombes à fragments :

Quelques piles TCP moins fiables sont connues pour avoir des problèmes à gérer de larges ensembles de fragments de paquets lorsqu'elles ne reçoivent pas tous les fragments. Linux n'a pas ce problème. Vous pouvez filtrer les fragments (ce qui peut casser les utilisations légitimes) ou compiler votre noyau avec l'option "IP: always defragment" mise sur "Y" (seulement si votre machine Linux est la seule route possible pour ces paquets).

1.7.7 Changer les règles pare-feu :

Il y a quelques problèmes de temps qui sont impliqués dans la modification des règles pare-feu. Si vous n'y faites pas attention, vous pouvez laisser entrer des paquets lorsque vous avez fait la moitié de vos changements. Une approche simpliste serait de faire la suite de commandes:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY
... Mise en place des changements ...
# ipchains -D input 1
# ipchains -D output 1
# ipchains -D forward 1
#
```

Ceci supprime tous les paquets pour la durée des changements.

Si vos changements sont restreints à une chaîne simple, vous pouvez créer une nouvelle chaîne avec les nouvelles règles, et ensuite remplacer ("-R") la règle qui pointait sur la vieille chaîne par une qui pointe sur la nouvelle chaîne ; ensuite, vous pouvez supprimer la vieille chaîne. Le remplacement se fera à vitesse atomique.

1.7.8 Comment mettre en place la protection contre l'IP spoof ?

L'IP spoofing est une technique dans laquelle un hôte envoie des paquets prétendant venir d'un autre hôte. Puisque le filtrage des paquets prend ses décisions sur la base de cette adresse source, l'IP spoofing est utilisé pour abuser les filtres de paquets. Elle est également utilisée pour cacher l'identité d'un attaquant utilisant les techniques SYN, Teardrop, Ping of Death et autres dérivés.

Le meilleur moyen de se protéger de l'IP spoofing est la vérification de l'adresse source, et il est réalisé par le code de routage, et non par le pare-feu et autres.
Cherchez un fichier nommé "/proc/sys/net/ipv4/conf/all/rp_filter". S'il existe, alors l'activation de la vérification de l'adresse source à chaque lancement est la bonne solution pour vous. Pour se faire, insérez les lignes suivantes quelque part dans vos scripts d'initialisation, avant l'initialisation des interfaces réseau :

```
# This is the best method: turn on Source Address Verification and get
# spoof protection on all current and future interfaces.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
echo -n "Setting up IP spoofing protection..."
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
echo 1 ¿ $f
done
echo "done."
else
echo PROBLEMS SETTING UP IP SPOOFING PROTECTION. BE WORRIED.
echo "CONTROL-D will exit from this shell and continue system startup."
echo
# Start a single user shell on the console
/sbin/sulogin $CONSOLE
fi
```

Si vous ne pouvez faire ceci, vous pouvez insérer manuellement les règles pour protéger chaque interface. Ceci nécessite la connaissance de chaque interface. La série des noyaux 2.1 de Linux et supérieurs rejette automatiquement les paquets qui prétendent venir des adresses 127.* (réservées pour

l'interface de loopback, lo).

Par exemple, disons que vous avez trois interfaces, eth0, eth1 et ppp0. Nous pouvons utiliser l'utilitaire "ifconfig" pour nous donner les adresses et les masques de réseau de chaque interface. Disons que eth0 est rattachée à un réseau 192.168.1.0 avec le masque de sous-réseau 255.255.255.0, eth1 est raccordée à un réseau 10.0.0.0 avec le masque de sous-réseau 255.0.0.0, et ppp0 connectée à l'Internet (où toutes les adresses sauf les adresses IP privées réservées sont autorisées), nous insérerions les règles suivantes :

```
# ipchains -A input -i eth0 -s ! 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i ! eth0 -s 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i eth1 -s ! 10.0.0.0/255.0.0.0 -j DENY
# ipchains -A input -i ! eth1 -s 10.0.0.0/255.0.0.0 -j DENY
#
```

Cette approche n'est pas aussi bonne que l'approche par vérification de l'adresse source, parce que si votre réseau change, vous devrez changer vos règles pare-feu pour être à jour.

Si vous utilisez un noyau de série 2.0, vous pouvez également vouloir protéger l'interface loopback, en utilisant une règle telle que :

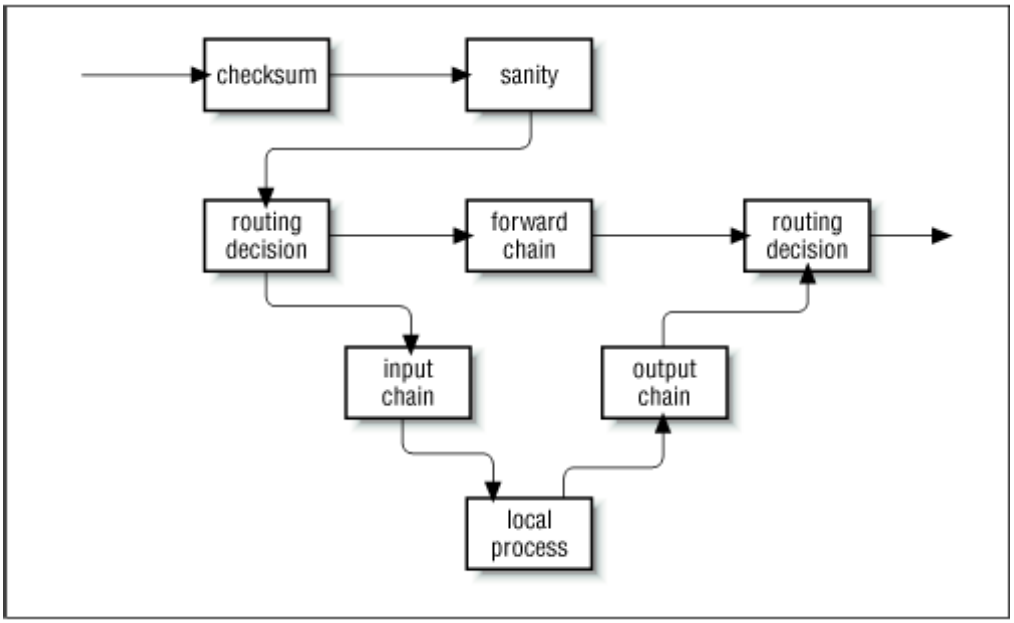
```
# ipchains -A input -i ! lo -s 127.0.0.0/255.0.0.0 -j DENY
#
```

2 IPTables :

Le noyau 2.4.x de Linux fit apparaître une nouvelle génération de "pare-feu IP" étant un descendant direct de l'utilitaire "ipchains". Ce nouvel utilitaire dispose de commandes syntaxiques similaires à l'utilitaire "ipchains" tout en y ajoutant quelques commandes supplémentaires. De plus il présente de nouvelles fonctionnalités totalement opérationnelles comme la translation d'adressage IP (NAT).

Les commandes et règles syntaxiques de filtrages sont identiques à celles de l'utilitaire "ipchains" vu au chapitre précédent. C'est pourquoi seules les commandes n'étant pas présentent dans l'utilitaire "ipchains" seront ici indiqués. Ceci dit afin d'assurer une bonne compréhension de celles-ci les définitions seront répétée. De plus la partie de translation d'adresses (NAT) ne sera pas exposée ici car celle-ci nécessiterai un chapitre entier pour sa définition.

Processus de traitement en chaîne d' "IPTables"



2.1 Liste exhaustive des spécifications de filtrage :

2.1.1 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 méthodes différentes. La méthode la plus commune est d'utiliser le nom complet, comme 'localhost' ou 'www.monurl.com'. La seconde méthode est de spécifier l'adresse IP comme '127.0.0.1'.

Les troisièmes et quatrièmes méthodes permettent de spécifier un groupe d'adresse IP, comme '199.95.207.0/24' ou '199.95.207.0/255.255.255.0'. Celles-ci spécifient toutes deux les adresses de 199.95.207.0 a 199.95.207.255 incluses; les nombres suivant le '/' indiquent quelle partie des adresses IP a de la signification. '/32' ou '/255.255.255.255' est le défaut (correspond a toutes les adresses IP). Pour spécifier toutes les adresses IP '/0' peut être utilisé, exemple:

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

Ceci est rarement utilisé, car ceci envoie le même résultat que si l'option '-s' n'était pas spécifiée.

2.1.2 Spécifier une inversion :

De nombreuses options comme '-s' (ou '–source') et '-d' ('–destination') peuvent avoir leurs arguments précédés de '!' (prononcé 'NOT') pour correspondre aux adresses non égales à celles spécifiées. Par exemple '-s ! localhost' correspond a tout paquet ne venant pas de localhost.

2.1.3 Spécifier un protocole :

Le protocole peut être spécifié avec l'option '-p' (ou '–protocol'). Le protocole peut être un nombre représentant la valeur numérique du protocole ou un nom tels 'TCP', 'UDP' ou encore 'ICMP'. La casse n'a pas d'importance, donc 'tcp' marche aussi bien que 'TCP'.

Le nom du protocole peut être préfixé d'un '!', afin de l'inverser, comme '-p ! TCP' pour spécifier les paquets n'étant pas du TCP.

2.1.4 Spécifier une interface :

Les options '-i' (ou '–in-interface') et '-o' (ou '–out-interface') spécifient le nom d'une interface à laquelle le paquet doit correspondre. Une interface est la connexion physique par lequel le paquet arrive ('-i') ou sort ('-o').

Remarque :
L'utilitaire "ifconfig" peut vous informer des différentes interfaces présentes actuellement sur votre machine.

Les paquets traversant la chaîne INPUT n'ont pas encore d'interface de sortie donc, une règle utilisant '-o' -o dans cette chaîne ne conviendra pas, les paquets traversant la chaîne OUTPUT n'ont pas d'interface d'entrée, donc toute règle utilisant '-i' dans cette chaîne ne correspondra jamais.

Seuls les paquets traversant la chaîne FORWARD on une interface d'entrée et de sortie.

Il est parfaitement légal de spécifier une interface qui n'existe pas; la règle ne sera pas prise en compte tant que l'interface n'est pas connectée (dit aussi 'up').

Un cas spécial est un nom d'interface se terminant par '+' car celle-ci conviendra à toutes les interface (existante ou non) commençant par ces lettres. Par exemple, pour spécifier une règle convenant à toutes les interfaces PPP, on utilisera l'option "-i ppp+".

Le nom de l'interface peut être précédé par un '!' pour convenir à un paquet n'appartenant pas à l'interface spécifiée.

2.1.5 Spécifier des fragments :

Lorsqu'un paquet à une taille trop importante, celui-ci est divisé en fragments, puis transmis en de multiples paquets plus petits. Par la suite le receveur réassemble ces fragments pour reconstruire le paquet entier.

Le problème avec ceci c'est que le fragment initial à son en-tête complète (IP + TCP, UDP et ICMP) pouvant être filtrée mais que les paquets suiv-

ants, donc les autres fragments du paquet, n'ont seulement que quelques morceaux de l'en-tète (IP sans les champs de protocole). Donc filtrer ces fragments n'est pas possible.

Ceci dit en utilisant le suivi de connexion ou le NAT, alors tout les fragments seront recollés avant qu'ils n'arrivent au code de filtrage de paquets, le problème est donc écarté.

Sinon il est important de comprendre comment les fragments sont traités par les règles de filtrage. Toute règle de filtrage demandant des informations que l'on ne dispose pas ne conviendra pas. Ceci veut dire 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 (spécifiant le port source 'www') ne conviendra jamais à un fragment autre que le premier fragment. L'opposé est aussi valable -p TCP –sport ! www et ne conviendra pas non plus.

Ceci étant dit, il est possible de spécifier une règle pour le second fragment et les suivants , en utilisant l'option '-f' (ou '–fragment'). Il est aussi légal de spécifier une règle qui ne s'applique pas au second fragment et aux suivants, en précédant '-f' avec '!'.

Note: les paquets mal formés (TCP, UDP et ICMP trop courts pour que le code de firewalling lise les ports ou le code ICMP et le type) sont rejetés (DROP) quand de telles combinaisons sont tentées, comme les fragments TCP commençant à la position 8.

Exemple :
La règle suivante va laisser tomber tout les fragments qui vont vers 192.168.1.1:

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

2.2 Extensions d "IPTables" :

2.2.1 Nouvelles Correspondances :

L'utilitaire "iptables" est extensible, en d'autres termes le noyau et le programme "iptables" peuvent être étendus afin d'avoir de nouvelles fonction-

nalités.

Quelques-unes de ces extensions sont standard, et d'autres non. Elles peuvent être créées par d'autres personnes et distribuées séparément aux utilisateurs.

Les extensions au noyau sont normalement situées dans le répertoire des modules du noyau (kernel) tel : "/lib/modules/2.3.15/net". Elles sont chargées à la demande si le noyau du système d'exploitation a été compilé avec l'option "CONFIG_KMOD".

Les extensions du programme "iptables" sont réellement des librairies partagées étant généralement situées dans le répertoire "/usr/local/lib/iptables/" sous Linux, ceci dit cela peut être modifié selon la distribution utilisé exemple : "/lib/iptables" ou encore "/usr/lib/iptables".

Les extensions sont de deux types différents: nouvelles correspondances et nouvelles cibles; nous parlerons des nouvelles cibles ci-dessous. Quelques protocoles offrent aussi de nouveaux tests : pour le moment TCP, UDP et ICMP.

Pour utiliser ces extensions il faut spécifier l'option '-m' pour charger l'extension, après quoi l'extension sera disponible.

Pour obtenir de l'aide sur une extension, il faut faire suivre l'option '-h' ou '–help' après l'une des options courantes telles que ('-p', '-j' ou '-m').

Exemple :

```
# iptables -p tcp –help
#
```

2.2.2 Extensions TCP :

Les extensions TCP sont automatiquement chargées si '-p tcp' est spécifié sur la ligne de commandes.

Celles-ci permettent les options suivantes (aucune d'entre elles ne conviennentt aux fragments) :

–tcp-flags

Suivi d'un '!' (NOT ou NON) optionnel, puis 2 chaînes de caractères de drapeaux (FLAGS) suivent, permettant de filtrer des drapeaux TCP spécifiques.

La première de ces chaînes de drapeaux est le masque.

La seconde chaîne de drapeaux indique celui devant être présent.

Exemple:

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

Ceci indique que tout les drapeaux doivent être examinés ('ALL' est synonyme de 'SYN, ACK, FIN, RST, URG, PSH'), mais seulement les drapeaux SYN et ACK doivent être présents. Il existe aussi l'argument 'NONE' aucun drapeau.

–syn

Précédé optionnellement d'un '!', est en fait un raccourci de '–tcp-flags SYN,RST,ACK, SYN'.

–source-port

Suivi d'un '!' (NOT ou NON) optionnel, puis suit soit un port TCP seul, soit un 'bloc' de ports. Les ports peuvent être des noms de ports, ceux-ci sont listés dans le fichier "/etc/services", ou des nombres. Les 'blocs' sont soit 2 noms de ports séparés par le caractère ':', ou (pour spécifier plus grand que ou égal à) un port avec un caractère ':' ajouté, ou bien encore (pour spécifier plus petit que ou égal à) un port précédé de ':'.

–sport

Est synonyme de '–source-port'.

–destination-port

et

–dport

Ceux-ci spécifient simplement la destination plutôt que la source qui convient.

–tcp-option
suivi d'un '!' (NOT ou NON) optionnel ou d'un nombre, convient à un paquet qui a une option TCP qui équivalent à ce nombre. Un paquet qui n'a pas une en-tête TCP complète est rejetée automatiquement si un essai est fait pour examiner ses options TCP.

2.2.3 Une Explication des Drapeaux TCP :

Il est parfois utile d'autoriser des connexions TCP dans une direction, mais pas dans l'autre. Par exemple, vous pouvez vouloir autoriser les connexions vers un serveur WWW externe, mais pas les connexions venant de ce serveur.

L'approche naïve serait de bloquer les paquets TCP venant du serveur. Malheureusement, les connexions TCP utilisent des paquets circulant dans les deux sens pour fonctionner correctement.

La solution est alors de bloquer uniquement les paquets utilisés pour la demande d'une connexion. Ces paquets sont nommés paquets SYN. En interdisant seulement ces paquets, nous pouvons stopper toute demande de connexion entrante.

Le drapeau '–syn' est utilisé pour cela : il est seulement valide pour les règles spécifiant TCP comme protocole. Par exemple, pour spécifier un essai de connexion de la part de 192.168.1.1 la syntaxe serait :
-p TCP -s 192.168.1.1 –syn

Ce drapeau peut être spécifié en le précédant de '!', qui veut dire tout les paquets sauf ceux d'initiation de connexion.

2.2.4 Extensions UDP :

Ces extensions sont automatiquement chargées si '-p udp' est spécifié. Elles donnent les options '–source-port', '–sport', '–destination-port' et '–dport' comme détaillé pour le tcp ci-dessus.

2.2.5 Extensions ICMP :

Ces extensions sont automatiquement chargées si '-p icmp' est spécifié. Elles donnent seulement une seule nouvelle option:

–icmp-type
Suivi d'un '!' (NOT ou NON) optionnel, puis suit un nom de type icmp (ex : 'host-unreachable'), ou un type numérique (ex : '3'), ou encore un type et un code numérique séparés par un '/' (ex : '3/3'). Une liste des types icmp disponibles est donnée en utilisant '-p icmp –help'.

2.2.6 Autres Extensions de Concordance :

Les autres extensions dans le package netfilter sont des extensions de démonstration, qui (si 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 concorder avec des adresses Ethernet (MAC) de paquets arrivant, et est seulement utile pour convenir avec des paquets traversant les chaînes INPUT et PREROUTING. Il donne une seule option:

–mac-source
Suivi d'un '!' (NOT ou NON) optionnel, puis suit une adresse Ethernet en notation hexadécimale séparée par la caractère ':', ex : '–mac-source 00:60:08:91:CC:B7'.

limit
Ce module doit être spécifié explicitement avec '-m limit' ou '–match limit'. Il est utilisé afin de limiter la vitesse de concordance, comme pour spécifier des messages de log. Il conviendra seulement à un certain nombre de fois pas seconde (par défaut 3 concordances par heure, avec une réserve de 5). Il prend deux arguments optionnels:

–limit
Suivi d'un nombre : spécifie le nombre maximum de concordances allouées par seconde. Le nombre peut spécifier les unités explicitement, en utilisant '/second', '/minute', '/hour' ou '/day', ou une partie (donc '5/second' est identique que '5/s').

–limit-burst
Suivi d'un nombre, il indique la réserve maximale avant d'atteindre la limite ci-dessus.

Cette concordance peut souvent être utilisée avec la cible LOG pour effectuer du logging limité en débit. Pour comprendre comment cela fonctionne on va regarder à la règle suivante, qui logge les paquets avec les paramètres de limite par défaut:

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

La première fois que cette règle est atteinte, le paquet sera loggé; en fait comme la réserve est de 5, les 5 premiers paquets seront loggés. Après cela, 20 minutes passeront avant qu'un paquet ne soit loggé par cette règle, sans tenir compte du nombre de paquets qui l'atteigne. Aussi, chaque 20 minutes qui passe sans concorder avec un paquet, un paquet de la réserve sera regagné; si aucun paquet n'atteint la règle pendant 100 minutes, la réserve sera complètement rechargée; on est revenu au point de départ.

Note : Pour ne pas créer de règle avec un temps de recharge de plus de 59 heures, donc avoir un débit moyen de 1 par jour, alors le débit de réserve doit être inférieur à 3.

On peut aussi utiliser ce module afin d'éviter les dénuement de service (DoS) avec un débit supérieur ce qui augmentera la vitesse de réaction.

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

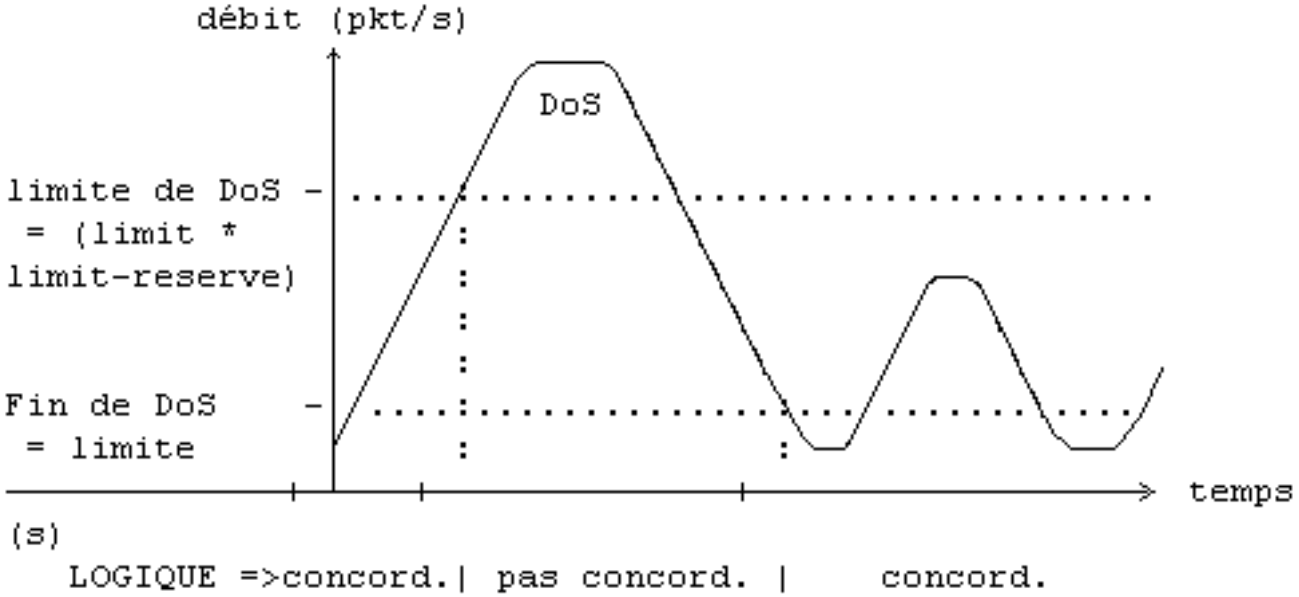
Test de ports furtif :
```
# iptables -A FORWARD -p tcp –tcp-flags SYN,ACK,FIN,RST RST -m limit
```

–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 montré dans le graphe qui suit.



On dit un paquet par seconde avec une réserve de 5 paquets mais, les paquets commences a arriver a 4/s pendant 3 secondes puis recommencent après 3 autres secondes

owner Ce module essaye de concorder les caractéristiques variées du créateur du paquet, pour les paquets générés localement. Il est seulement valide pour la chaîne OUTPUT, et même comme cela, certains paquets (comme les réponses ICMP) n'auront pas de possesseur, et donc ne concorderont jamais.

–uid-owner userid
Concorde si le paquet a été créé par un processus qui a le user id effectif (numérique) donné.

–uid-owner groupid
Concorde si le paquet a été créé par un processus qui a le group id effectif (numérique) donné.

–pid-owner processid
Concorde si le paquet a été créé par un processus qui a l'id du processus donné.

–sid-owner processid
Concorde si le paquet a été créé par un processus qui a le groupe de session donné.

unclean
Ce module expérimental doit être spécifié explicitement avec '-m unclean ou '–match unclean'. Il effectue des vérifications sanitaires aléatoires et variées sur les paquets. Ce module n'a pas été testé, et ne devrait pas être utilisé comme une fonction de sécurité (il rend peut-être les choses plus mauvaises si il a des erreurs de développement). Il n'a pas d'options.

2.2.7 La Concordance d'Etat :

Le critère le plus utile est fourni par l'extension 'state' interprétant l'analyse de suivi de connexion du module 'ip_conntrack'. Il est fortement recommandé de l'utiliser.

Spécifier '-m state' accepte une option additionnelle '–state', qui est une liste, séparée par des virgules, d'états convenant (les '!' indiquent les états qui ne conviennent pas. Ces états sont :

NEW
Un paquet engendrant une nouvelle connexion.

ESTABLISHED
Un paquet appartenant à une connexion existante (ex : une qui a eu des paquets de réponse).

RELATED
Un paquet qui est associée à, mais pas ne fait pas partie d'une connexion existante, comme une erreur ICMP, ou (avec le module ftp chargé), un paquet établissant une connexion de données par le protocole FTP.

INVALID
Un paquet ne pouvant être identifié pour n'importe quelle raison ; ceci inclus le manque de mémoire et les erreurs ICMP qui ne correspondent à aucune connexion connue. Généralement, ces paquets doivent être rejetés.

2.3 Spécifications de Cibles :

Après avoir vu les différents examens pouvant être réalisés sur un paquet, il faut maintenant définir les actions à établir sur ceux-ci. Ceci est appelé la cible d'une règle.

Il y a deux cibles simples compilées: DROP et ACCEPT. Nous les avons déjà rencontrées. Si une règle correspond à un paquet et que la cible est une de celles ci, les règles suivantes ne sont pas consultées : le destin du paquet est alors décidé.

Il y a deux autres types de cibles que celles compilées: les extensions et les chaînes créées par l'utilisateur.

2.3.1 Les chaînes créées par l'utilisateur :

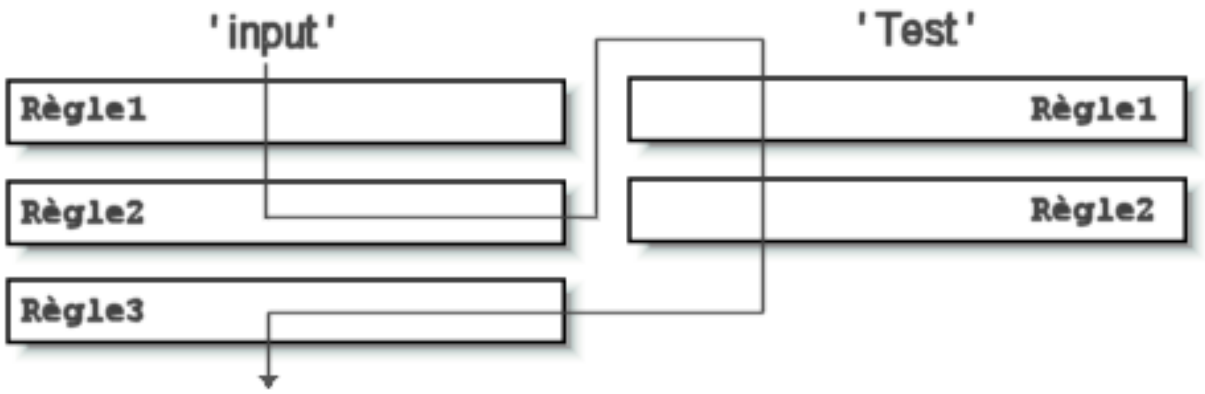
Une propriété puissante que l'utilitaire "iptables" hérite d' "ipchains" est la possibilité pour l'utilisateur de créer de nouvelles chaînes, en addition a celles déjà existantes (INPUT, FORWARD et OUTPUT). Par convention les chaînes utilisateurs sont en minuscules afin de les distinguer.

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

Considérons deux chaines: INPUT (la chaine par defaut) et test (une chaine utilisateur).

Considérons un paquet TCP venant de 192.168.1.1 et allant vers 1.2.3.4. Il entre dans la chaîne INPUT, et est testé par la règle1 - pas de concordance. La règle2 concorde et sa cible est test, donc la règle suivante examinée est le début de la chaîne test. La règle1 de test concorde mais ne spécifie pas de cible, donc la règle suivante est examinée, la règle2. Ca ne concorde pas, nous avons atteint la fin de la chaîne test. Nous retournons à la chaîne INPUT ou nous venons d'examiner la règle2, donc nous examinons la règle3, qui ne concorde pas d'avantage.

Donc la traversée du paquet est :



Les chaînes utilisateurs peuvent aller dans d'autres chaînes utilisateurs (mais ne peut faire de boucle : les paquet seront rejetés si ils sont dans une boucle).

2.4 Extensions d "IPTables" :

2.4.1 Nouvelles Cibles :

L'autre type de cible est une extension. Une extension de cible consiste en un module du noyau, et une extension optionnelle à "iptables" permettant de nouvelles options sur la ligne de commande. Il y a plusieurs extensions dans la distribution "netfilter" :

LOG

Ce module permet de logger les paquets qui concordent. Il donne plusieurs options additionelles:

–log-level

Suivi d'un nombre de niveau ou d'un nom. Les noms valides sont (non sensible à la casse) 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' and 'emerg', et correspondent aux nombres allant de 7 à 0. Voir la page de manuel de "syslog.conf" pour une explication de ces niveaux.

–log-prefix

Suivi d'une chaîne de caractères de 30 caractères maximum, ce message est envoyé au début du message de log, pour uniquement lui permettre d'être identifié.

Ce module est le plus utile après une cible "limit", pour ne pas saturer les logs.

REJECT

Ce module est identique à la cible 'DROP', excepté que l'envoyeur reçoit un message d'erreur ICMP 'port unreachable'. Il est à noter 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.

Nous avons envoyé trop de messages d'erreur ICMP à cette destination récemment.

REJECT peut aussi accepter une option '–reject-with' altèrant le type du paquet de réponse utilisé.

2.4.2 Cibles spéciales compilées :

Il y a 2 cibles spéciales compilées: RETURN et QUEUE.

RETURN va directement la fin d'une chaîne: pour une règle dans une chaîne compilée, la police de la chaîne est exécutée. Pour une règle dans une chaîne utilisateur, la traversée continue dans la chaîne précédente, juste après la règle qui est allée sur cette chaîne.

QUEUE est une cible spéciale, elle met les paquets en queue afin de les traiter en espace utilisateur. Pour que cela soit utile, deux composants supplémentaires sont requis:

un "arbitre de queue", qui utilise les mécanismes du noyau pour passer les paquets entre le noyau et l'espace utilisateur.

une application utilisateur pour recevoir, voir manipuler, et donner un verdict sur les paquets.

L'arbitre de queue standard IPV4 pour l'utilitaire "iptables" est le module "ip_queue", qui est distribué avec le noyau et marqué comme expérimental.

Voici un exemple rapide d'utiliser "iptables" afin de mettre des paquets en queue pour les traiter en espace utilisateur:

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

Avec cette règle, les paquets ICMP sortants générés localement (comme créés par l'exemple avec l'utilitaire ping) sont passés au module "ip_queue", qui essaye par la suite de délivrer les paquets à une application utilisateur. Si aucune application n'est présente pour prendre les paquets, ils sont alors rejetés.

Pour écrire une application utilisateur, l'API "libipq" devra être utilisée. Elle est distribuée avec l'utilitaire "iptables".

Le statut de "ip_queue" peut être vérifié par:

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

La longueur maximale de la queue (le nombre de paquets délivrés à l'espace utilisateur sans verdict retourné) peut être contrôlé par:

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

La valeur par défaut de la longueur maximale de la queue est égale à 1024. Une fois que cette limite est atteinte, les nouveaux paquets seront rejetés jusqu'a ce que la longueur de la queue soit inférieure à cette limite. Les

protocoles comme le TCP interprètent les paquets rejetés comme une congestion, et réagiront positivement lorsque la queue se remplit. Des expériences doivent être faites pour pouvoir déterminer la longueur de queue maximale pour une situation donnée si la valeur par défaut est trop petite.

Conclusion :

Les outils concernant le filtrage de paquets d'informations "IPChains" et "IPTables" on tous deux fait leurs preuves dans divers milieux de production. Ceci étant dit "IPTables" est une évolution majeure d' "IPChains" sans être pour autant une copie littérale du code source de ce dernier. De plus "IPTables" offre les même possibilités que l'utilitaire "IPChains" ajoutant à ce dernier une grande flexibilité d'utilisation, via des modules dynamiques externes pouvant lui être attachés, ainsi que quelques simplifications syntaxiques.

Remarquons plus généralement l'évolution de ce type de logiciel dans le domaine du masquage et du routage de paquets. Il y a fortement à penser que les évolutions ainsi que les besoins futures de ce type d'utilitaires vont tendre vers une flexibilité accrue des définition même des règles de filtrage et s'élargir sur des protocoles biens plus divers que ceux supportés actuellement.

En effet nous pouvons constater qu'entre les trois dernières générations de pare-feu IP sous Linux les définitions syntaxiques se structurent de plus en plus de manière algorithmique laissant probablement entendre qu'un langage de " programmation " propre à ces utilitaires pourrait voir le jour.

Il est aussi à noter que si les évolutions de ces outils sont aussi techniquement avancés c'est peut-être dût au fait que de plus en plus de personnes sont malveillantes et que ne pas se protéger contre celles-ci devient de plus en plus irresponsable.

Glossaire:

Firewall : Système permettant de protéger un réseau local d'intrusions de personnes en provenance d'Internet. On l'appelle parfois aussi:

pare-feu garde-barrière antéserveur écluse gate-kepper coupe-feu

Le système firewall est un système à la fois matériel et logiciel. Ce système constitue un intermédiaire entre le réseau local et le "monde extérieur". Il permet de bloquer des port de communication, c'est-à-dire en interdire l'accès aux personnes provenant de l'extérieur. Il fonctionne sur le principe du filtrage de paquets, c'est-à-dire qu'il s'agit d'un système analysant les en-têtes des paquets IP (aussi appelés datagrammes) échangés entre deux machines.

Port : est un numéro associé à un service ou une application réseau.

Chaîne : Vérification de règles.

Références:

Linux Network Administrator's Guide, 2nd Edition par Olaf Kirch & Terry Dawson

Edition O'Reilly URL : http://www.oreilly.com/

Linux IPCHAINS-HOWTO par Rusty Russell

URL : http://netfilter.samba.org/ipchains/HOWTO.html

Linux IPCHAINS-HOWTO de Paul Russell en version française par Arnaud Launay

URL : http://www.freenix.fr/unix/linux/HOWTO/IPCHAINS-HOWTO.html

IPTables Tutorial 1.1.6 de Oskar Andreasson.

Linux 2.4 Packet Filtering HOWTO.

URL : http://www.commentcamarche.net

URL : http://www.iptables.org