Depuis que Netfilter a fait son apparition en standard avec le noyau 2.4, un firewalling sérieux est possible sous Linux. Ce dispositif de filtrage introduit :

- La notion d'état permettant de savoir si un paquet appartient à une communication en cours ou non ;
- Le NAT : la translation d'adresse ;
- La possibilité de modifier des paquets à la volée.

Depuis, les possibilités de Netfilter n'ont cessé d'évoluer. Nous verrons successivement comment protéger un poste utilisateur, un serveur et, pour finir, un réseau.

Protéger un poste utilisateur

Qu'il s'agisse d'un poste directement connecté à Internet, équipé d'une carte WIFI ou connecté à un réseau d'entreprise, votre ordinateur est, disons-le franchement, en milieu hostile. C'est pourquoi il convient de le protéger de connexions réseau indésirables.

Les tables

Netfilter est composé de trois tables :

- filter : la table de filtrage ;
- nat : la table de translation d'adresse ;
- mangle pour altérer les paquets.

La table nat sera étudiée lors de la protection d'un réseau, la table mangle lors des applications avancées comme la gestion de bande passante ou de plusieurs FAI. Commençons donc avec la table filter responsable du filtrage de paquets. Les règles de filtrages sont organisées en trois chaînes : INPUT, OUTPUT et FORWARD. Comme le nom l'indique, ces règles régissent respectivement les filtrages des paquets reçus, émis ou en transits. Le cas du transit concerne les passerelles, cas que nous verrons dans « Protéger un réseau ».

Mon premier firewall

```
iptables -F
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT   -i lo   -j ACCEPT
iptables -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
```

La première commande fait table rase (F=Flush) des chaînes existantes de la table filter. Les trois suivantes définissent la politique (P=policy) par défaut pour les trois chaînes. La politique appliquée peut être ~~ACCEPT~~, ~~DROP~~ ou ~~REJECT~~ pour respectivement autoriser, ignorer ou rejeter un paquet. Contrairement à ~~DROP~~, ~~REJECT~~ va prévenir la machine distante que le paquet est refusé. Faisant confiance (pour le moment) au firewall, on l'autorise à émettre sans restriction. Sans la dernière ligne, la machine peut émettre des paquets, mais ignore les réponses, elle est donc incapable de dialoguer. La dernière ligne indique d'accepter les paquets liés à un état.

- ~~ESTABLISHED~~, connexion établie au sens TCP par l'échange SYN/SYN-ACK ou attendu au sens UDP/Ping ICMP ;
- ~~RELATED~~, connexion relative à une connexion déjà établie. C'est nécessaire pour gérer les protocoles de communication multicanaux.

Module FTP

Avec cette configuration, vous pouvez naviguer sur des sites web, envoyer des mails, mais rapidement vous vous rendez compte que vous ne pouvez pas faire de FTP. Pour être plus précis, cela ne fonctionne pas toujours.

```
ncftp ftp.free.fr
...
Logged in to ftp.free.fr.
ncftp / > cd pub/linux/kernel
Directory successfully changed.
ncftp /pub/linux/kernel > debug
ncftp /pub/linux/kernel > set passive
> set passive
passive                       on
ncftp /pub/linux/kernel > get README
> get README
Cmd: TYPE I
200: Switching to Binary mode.
Cmd: MLST README
500: Unknown command.
Cmd: SIZE README
213: 12056
Cmd: MDTM README
213: 19960916163600
Cmd: EPSV
229: Entering Extended Passive Mode (|||30102|)
Cmd: RETR README
150: Opening BINARY mode data connection for README (12056 bytes).
README:     ETA:   0:00    0.71/ 11.77 kB   42.25 kB/s   226: File send OK.
README:                           11.77 kB  362.78 kB/s
```

En mode passif, le fichier a été récupéré sans problème.

```
ncftp ftp.free.fr
...
Logged in to ftp.free.fr.
ncftp / > cd pub/linux/kernel
Directory successfully changed.
ncftp /pub/linux/kernel > debug
ncftp /pub/linux/kernel > set passive off
> set passive off
> get README
Cmd: TYPE I
200: Switching to Binary mode.
Cmd: MLST README
500: Unknown command.
Cmd: SIZE README
213: 12056
Cmd: MDTM README
213: 19960916163600
Cmd: PORT 172,18,100,33,176,86
200: PORT command successful. Consider using PASV.
Cmd: RETR README
425: Failed to establish connection.
get README: server said: Failed to establish connection.
ncftp /pub/linux/kernel >
Cmd: QUIT
```

```
221: Goodbye.
```

Voila, le problème est clairement identifié. Le mode actif (mode par défaut du FTP de Windows) ne fonctionne pas ! FTP utilise un protocole bi-canal, l'ouverture du second canal pour la transmission des données est effectuée par les commandes circulant dans le premier canal. La connexion au second canal s'effectue du client vers le serveur en mode passif et du serveur vers le client en mode actif. Pour avoir connaissance des ports utilisés, le firewall doit donc étudier la communication transitant sur le port TCP 21. Le module ~~ftp_conntrack_ftp~~ est chargé de cette opération. Pour le charger, il suffit d'ajouter en début de script ~~modprobe ip_conntrack_ftp~~. Par défaut, seules les connexions FTP sur le port classique (tcp/21) sont analysées. Si vous avez un proxy FTP sur le port 2121, vous devez charger le module ~~ip_conntrack_ftp~~ de la façon suivante modprobe ~~ip_conntrack_ftp ports=21,2121~~.

```
Gestion des états
```

Pour Netfilter, un paquet peut être dans l'un des états suivants :

- ~~INVALID~~ : le paquet est en erreur, il contient un en-tête invalide par exemple ou bien le paquet ne peut pas être identifié.
- ~~ESTABLISHED~~ : le paquet appartient à une connexion établie.
- ~~RELATED~~ : le paquet débute une nouvelle connexion relative à une connexion établie (exemple FTP).
- ~~NEW~~ : le paquet débute une nouvelle connexion.

Donc si l'état associé avec le paquet est ~~NEW~~ ou ~~RELATED~~, le paquet commence une connexion. Toute communication TCP doit commencer par un paquet comportant le flag SYN. Si ce n'est pas le cas, on peut rejeter la connexion ou l'ignorer. Soyons encore plus stricts, n'acceptons que les paquets TCP avec le flag SYN parmi SYN, ACK, FIN, RST, URG et PSH. Cela bloquera les scans de ports dits « furtifs ».

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state INVALID -j DROP
iptables -A INPUT -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -j DROP
```

Délai à l'établissement de certaines connexions

Si certaines connexions FTP ou SMTP sortantes ont du mal à démarrer, la cause peut être toute simple. Le serveur sur lequel vous vous connectez peut chercher à identifier l'utilisateur à l'origine de la connexion (RFC 1413 – Identification Protocol) en se connectant sur le port auth/ident (TCP 113). La politique par défaut mise en place sur les connexions entrantes est ~~DROP~~. L'émetteur va donc tenter plusieurs connexions jusqu'à ce qu'il abandonne après un certain temps. Pour ne pas avoir à subir ce délai, le plus simple est de rejeter la connexion, car vous n'avez aucune raison de répondre à cette requête, obsolète par ailleurs.

```
iptables -A INPUT -p tcp --destination-port auth -j REJECT --reject-with tcp-reset
```

La connexion sera rejetée avec un Reset TCP.

Consignation des anomalies réseau

Afin de connaître les connexions bloquées par le firewall, il faut mettre en place un système d'enregistrement. Le système d'enregistrement classique est la cible LOG, les logs sont traités par le démon syslogd.

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state INVALID -m limit --limit 4/s -j LOG --log-prefix «INPUT INVALID «
iptables -A INPUT -m state --state INVALID -j DROP
iptables -A INPUT -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -m limit --limit 4/s -j LOG --log-prefix «TCP INPUT without SYN «
iptables -A INPUT -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --destination-port auth -j REJECT --reject-with tcp-reset
iptables -A INPUT -m limit --limit 4/s -j LOG --log-prefix «INPUT bad «
iptables -A INPUT -j DROP
```

Chaque ligne de log est préfixée grâce à ~~--log-prefix~~ par un petit commentaire allant jusqu'à 29 caractères. Le module ~~limit~~ est utilisé pour limiter le nombre d'enregistrement dans le temps. Dans cette configuration, le nombre moyen d'enregistrement est de 4 par seconde. Cela permet de limiter les risques de saturation de la partition hébergeant les logs. Si vous travaillez en console, vous allez rapidement vous rendre compte qu'elle devient difficilement utilisable car des lignes de logs Netfilter apparaissent sans cesse (enfin, surtout si vous êtes connecté à Internet). Le démon klogd est chargé de gérer les logs émis par le noyau, comme les alertes sont envoyées avec le niveau de gravité 4 (~~KERN_WARNING~~), il faut lui dire de ne répercuter sur la console que les messages de gravité 3 (~~KERN_ERR~~) ou plus critique (~~KERN_CRIT=2, KERN_ALERT=1, KERN_EMERG=0~~). Sur une RedHat ou une Fedora, mettre dans ~~/etc/sysconfig/syslog~~ :

```
KLOGD_OPTIONS=»-x -c 3»
```

Par défaut, ~~klogd/syslogd~~ écrit les informations de manière synchrone : chaque écriture est suivie d'un ~~flush~~ qui force l'écriture sur disque des informations. Le délai introduit par cette synchronisation pénalise fortement les performances de votre machine. Il est préférable de modifier le fichier de configuration ~~/etc/syslog.conf~~ pour passer en asynchrone (exemple d'une Fedora) :

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                -/var/log/messages
```

Remarque : Une alternative possible au système classique de log est d'utiliser la cible ULOG. Dans ce cas, le démon ulogd, http://www.netfilter.org/projects/ulogd/, va gérer les enregistrements, le stockage peut s'effectuer dans une base de données.

Filtrage des adresses privées

Sur Internet ne doit être routé que du trafic portant sur des adresses publiques. Il se peut (worms, erreur de configuration) que des paquets arrivent depuis Internet jusqu'à votre machine avec comme adresse source une adresse IP privée. Il est inutile d'accepter ce genre de paquet, ne nous gênons pas pour le faire. Dans cet exemple, eth0 est l'interface internet.

```
iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 127.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
```

De même, il ne sert à rien de tenter une connexion vers une adresse IP privée en allant sur Internet. Un ~~REJECT~~ va mettre fin immédiatement à cette tentative.

```
iptables -A OUTPUT -o eth0 -d 10.0.0.0/8 -j REJECT
iptables -A OUTPUT -o eth0 -s 127.0.0.0/8 -j REJECT
iptables -A OUTPUT -o eth0 -d 172.16.0.0/12 -j REJECT
iptables -A OUTPUT -o eth0 -d 192.168.0.0/16 -j REJECT
```

Filtrage des broadcasts

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

Autoriser un accès à votre serveur

Si vous décidez d'héberger un petit site web sur ce serveur, il va falloir autoriser le trafic web. Rien de bien sorcier :

```
   iptables -A INPUT -p tcp --destination-port http -j ACCEPT
```

Admettons, pour l'exemple, que pour télécharger la dernière Fedora, je décide d'utiliser du Peer-To-Peer. Le principe de ces réseaux est de mettre à disposition les séquences ou morceaux de fichier récupérés, les peers sur

lesquels on se connecte nécessitant de pouvoir se connecter eux-mêmes sur la machine à l'origine de la connexion. Selon le logiciel de Torrent utilisé, il est possible de spécifier la plage de port source qui va être utilisée pour partager les données déjà reçues. Il s'agit souvent des ports TCP 6881-6999.

```
iptables -A INPUT -p TCP --destination-port 6881:6900 -j ACCEPT
```

Donner un accès à votre réseau interne

Chaîne utilisateurs

Dans cet exemple, notre firewall a deux interfaces :

- ~~eth0~~, l'interface externe connectée à Internet ;
- ~~eth1~~ reliant le firewall au réseau interne.

~~good~~ va désigner le réseau interne, ~~bad~~ Internet et me le firewall. Définissons les chaînes utilisateurs correspondant aux différentes communications possibles. Par exemple, ~~good-bad~~ désigne les nouvelles communications provenant du réseau interne à destination d'Internet. Le début du script devient :

```
iptables -F
iptables -F -t nat
iptables -X
iptables -N bad-me
iptables -N good-bad
iptables -N good-me
```

Après avoir supprimé les chaînes systèmes ~~-F~~ (~~--flush~~) des tables filter et nat, on efface les chaînes utilisateurs ~~-X~~ (~~--delete-chain~~) déjà existantes, s'il y en a, avant de définir ses propres chaînes ~~-N~~ (~~--new-chain~~).

Dispatch vers les règles utilisateurs

Reprenons la règle ~~INPUT~~ pour orienter selon l'interface d'entrée (~~-i in-interface~~) vers la chaîne utilisateur.

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -s 10.0.0.0/8       -j DROP
iptables -A INPUT -i eth0 -s 127.0.0.0/8      -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12    -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16   -j DROP
iptables -A INPUT -m state --state INVALID -m limit --limit 4/s -j LOG\ --log-prefix «INPUT INVALID «
iptables -A INPUT -m state --state INVALID    -j DROP
iptables -A INPUT -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN\ -m limit --limit 4/s -j LOG --log-prefix «TCP INPUT without SYN «
iptables -A INPUT -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN\ -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --destination-port auth -j REJECT --reject-with\ tcp-reset
iptables -A INPUT   -i eth0 -j bad-me
iptables -A INPUT   -i eth1 -j good-me
iptables -A INPUT -m limit --limit 4/s -j LOG --log-prefix «INPUT bad «
iptables -A INPUT -j DROP
```

Faisons de même pour la chaîne ~~FORWARD~~ traitant les connexions passant via le firewall.

```
iptables -A FORWARD -i ! eth0 -o eth0 -p tcp -d 10.0.0.0/8    -j REJECT --reject-with tcp-reset
iptables -A FORWARD -i ! eth0 -o eth0 -p tcp -d 127.0.0.0/8    -j REJECT --reject-with tcp-reset
iptables -A FORWARD -i ! eth0 -o eth0 -p tcp -d 172.16.0.0/12         -j REJECT --reject-with tcp-reset
iptables -A FORWARD -i ! eth0 -o eth0 -p tcp -d 192.168.0.0/16        -j REJECT --reject-with tcp-reset
iptables -A FORWARD -i ! eth0 -o eth0 -d 10.0.0.0/8    -j REJECT
iptables -A FORWARD -i ! eth0 -o eth0 -d 127.0.0.0/8   -j REJECT
iptables -A FORWARD -i ! eth0 -o eth0 -d 172.16.0.0/12         -j REJECT
iptables -A FORWARD -i ! eth0 -o eth0 -d 192.168.0.0/16        -j REJECT
iptables -A FORWARD   -m state --state INVALID -m limit --limit 4/s -j LOG --log-prefix «FORWARD INVALID «
iptables -A FORWARD   -m state --state INVALID -j DROP
iptables -A FORWARD   -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -m limit --limit 4/s -j LOG\ --log-prefix «TCP FORWARD without SYN «
iptables -A FORWARD -i ! eth0 -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -j REJECT
iptables -A FORWARD    -m state --state NEW,RELATED -p tcp --tcp-flags ! ALL SYN -j DROP
iptables -A FORWARD    -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp --destination-port auth -j REJECT --reject-with tcp-reset
iptables -A FORWARD -i eth1 -o eth0 -j good-bad
iptables -A FORWARD -j LOG -m limit --limit 3/s --log-prefix «bad FORWARD «
iptables -A FORWARD -j DROP
```

Création des règles utilisateurs

Depuis Internet, les connexions sur le serveur web sont autorisées ainsi que les connexions ~~Torrent~~, mais seulement cela.

```
iptables -A bad-me -p tcp --destination-port http -j ACCEPT
iptables -A bad-me -p TCP --destination-port 6881:6900 -j ACCEPT
iptables -A bad-me -m limit --limit 4/s -j LOG --log-prefix «bad-me bad «
iptables -A bad-me -j DROP
```

Le réseau interne est autorisé à se connecter sur le serveur web et afin de permettre l'administration du firewall/serveur web depuis le réseau interne, on autorise l'accès ssh. Les connexions du réseau interne vers d'autres ports sont rejetées. Par défaut, un paquet ICMP port-unreachable est renvoyé lors d'un ~~REJECT~~ mais dans le cas des connexions TCP, il est préférable de renvoyer un TCP Reset car sinon la réponse peut être interprétée comme une indisponibilité temporaire et le client à l'origine de la connexion va retenter la connexion.

```
iptables -A good-me -p tcp --destination-port http -j ACCEPT
iptables -A good-me -p tcp --destination-port ssh  -j ACCEPT
iptables -A good-me -m limit --limit 4/s -j LOG --log-prefix «good-me bad «
iptables -A good-me -p tcp -j REJECT --reject-with tcp-reset
iptables -A good-me -j REJECT
```

Le réseau interne a le droit de faire du ftp, du http et du https. Le module ~~multiport~~ permet de spécifier jusqu'à 15 ports en une seule ligne.

```
iptables -A good-bad -p tcp -m multiport --destination-port ftp,http,https -j ACCEPT
iptables -A good-bad -m limit --limit 4/s -j LOG --log-prefix «good-bad bad «
iptables -A good-bad -p tcp -j REJECT --reject-with tcp-reset
iptables -A good-bad -j REJECT
```

Activation du routage

Dans le fichier de configuration ~~/etc/sysctl.conf~~, indiquez ~~net.ipv4.ip_forward = 1~~ pour activer le routage. La commande ~~sysctl -p~~ va paramétrer le noyau à partir des valeurs de ce fichier. Remarque : Ce même paramétrage est aussi accessible par l'arborescence ~~/proc echo 1 > /proc/net/ipv4/ip_forward~~.

Protection antispoofing

Certaines attaques de type déni de service ou idle scan vont usurper l'adresse IP source. Le noyau Linux offre la possibilité d'ignorer les paquets venant d'une interface alors que la table de routage indique une autre interface pour la réponse. Dans ~~/etc/sysctl.conf~~, indiquez ~~net.ipv4.conf.default.rp_filter = 1~~.

Masquerading

Votre réseau interne ne peut communiquer tel quel avec Internet. Les serveurs sur lesquels vous vous connectez ne pourront vous répondre que si la source des paquets est une adresse publique. C'est là que le masquerading intervient, chaque paquet émis vers Internet va utiliser l'adresse source de l'interface de sortie.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Architecture réseau, la DMZ

Imaginons qu'un service est accessible depuis Internet sur un serveur se trouvant dans le réseau utilisateur. Si une faille est présente sur ce serveur, le réseau interne risque d'être compromis. Pour réduire ce risque, il est possible d'ajouter une zone tampon, la DMZ, pour héberger les serveurs devant être accessibles depuis Internet. Notre firewall a désormais trois interfaces :

- ~~eth0~~, l'interface externe connectée à Internet ;
- ~~eth1~~ reliant le firewall au réseau interne ;
- ~~eth2~~, l'interface de la DMZ, la zone où se trouvent les serveurs accessibles depuis Internet.

Si le service se révèle être vulnérable, le pirate va se retrouver coincé dans la DMZ, le firewall ne lui permettra pas d'exploiter les failles présentes sur les postes clients. Il nous faut ajouter les chaînes utilisateurs correspondantes.

```
iptables -N dmz-me
iptables -N bad-dmz
iptables -N good-dmz
iptables -N dmz-bad
iptables -N dmz-good
```

ainsi que les redirections vers les chaînes utilisateurs

```
iptables -A INPUT   -i eth2 -j dmz-me
iptables -A FORWARD -i eth0 -o eth2 -j bad-dmz
iptables -A FORWARD -i eth1 -o eth2 -j good-dmz
iptables -A FORWARD -i eth2 -o eth0 -j dmz-bad
iptables -A FORWARD -i eth2 -o eth1 -j dmz-good
```

Pour l'exemple, je vais considérer que la DMZ contient un relais de messagerie en 192.168.1.1, un cache Internet 192.168.1.2 et un serveur web 192.168.1.3. Le serveur de messagerie interne est en 10.0.0.1. Définissons dans le script les adresses IP sous forme de variable, cela permettra de gérer facilement ces serveurs par des noms.

```
RELAIS_SMTP=192.168.1.1
SRV_PROXY=192.168.1.2
SRV_WEB=192.168.1.3
SRV_SMTP=10.0.0.1
```

Règles de filtrage

Depuis Internet, seuls les serveurs de messagerie et le serveur web sont joignables.

```
iptables -A bad-dmz -p tcp -d $RELAIS_SMTP  --dport smtp -j ACCEPT
iptables -A bad-dmz -p tcp -d $SRV_WEB      --dport http -j ACCEPT
iptables -A bad-dmz -m limit --limit 4/s -j LOG --log-prefix «bad-dmz bad «
iptables -A bad-dmz -j DROP
```

Limitons aussi les connexions du réseau interne vers la DMZ mais tolérons le ping.

```
iptables -A good-dmz -p tcp -d $RELAIS_SMTP         --dport smtp -j ACCEPT
iptables -A good-dmz -p tcp -d $SRV_WEB     --dport http -j ACCEPT
iptables -A good-dmz -p tcp -d $SRV_PROXY   --dport squid -j ACCEPT
iptables -A good-dmz -p icmp --icmp-type echo-request -j ACCEPT
iptables -A good-dmz -m limit --limit 4/s -j LOG --log-prefix «bad-dmz bad «
iptables -A good-dmz -p tcp -j REJECT --reject-with tcp-reset
iptables -A good-dmz -j REJECT
```

L'analyse des logs permet de découvrir les tentatives de rebond d'un pirate si celui-ci a piraté un serveur de la DMZ.

NAT, Network Address Translation

Brusquement, vous revenez à la raison ! Vous vous rappelez que votre connexion Internet ne vous donne qu'une seule adresse IP publique et deux serveurs doivent être accessibles depuis Internet : le serveur web et la passerelle de messagerie. Une sueur froide vous parcourt l'échine, cela fait plusieurs heures que vous galérez sur ce projet de firewall. Pas de panique ! La solution existe. La translation d'adresse va venir à votre rescousse, on va rediriger les ports concernés vers les bonnes destinations.

```
iptables -t nat -A PREROUTING -j DNAT -i eth0 -p tcp --dport smtp --to-destination\ $RELAIS_SMTP
iptables -t nat -A PREROUTING -j DNAT -i eth0 -p tcp --dport http --to-destination\ $SRV_WEB
```

Ben voila, ce n'était pas si compliqué. Si jamais vous disposez de plusieurs adresses IP, il suffit de définir les adresses IP publiques, puis de réaliser les translations d'adresse. Admettons que votre provider vous ait accordé la grâce d'un adressage publique /29, soit 8 adresses publiques (2 adresses de broadcast, soit 6 utilisables) A.B.C.0 à A.B.C.7.

- A.B.C.0 broadcast (/29 et /30) ;
- A.B.C.1 le routeur ;
- A.B.C.2 votre firewall;
- A.B.C.3 brodcast(/29);
- A.B.C.4 votre relais SMTP ;
- A.B.C.5 votre serveur web ;
- A.B.C.6 libre ;
- A.B.C.7 broadcast (/30).

Au niveau du routeur, l'interface réseau doit être en A.B.C.1/30 avec une route pour le réseau A.B.C.0/29 utilisant la passerelle A.B.C.2. Au niveau du firewall l'interface eth0 doit être configurée en A.B.C.2/30 avec comme passerelle par défaut A.B.C.1.

```
RELAIS_SMTP_PUB=A.B.C.4
SRV_WEB_PUB=A.B.C.5
iptables -t nat -A PREROUTING -j DNAT -d $RELAIS_SMTP_PUB\      --to-destination $RELAIS_SMTP
iptables -t nat -A PREROUTING -j DNAT -d $SRV_WEB_PUB\          --to-destination $SRV_WEB
```

Chaque serveur est désormais joignable sur sa propre adresse publique. Mais si le proxy va récupérer une page sur un site web, l'adresse IP source visible du serveur web sera celle du firewall. Réalisons la translation d'adresse dans les deux sens :

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT -s $RELAIS_SMTP\         --to-source $RELAIS_SMTP_PUB
iptables -t nat -A POSTROUTING -o eth0 -j SNAT -s $SRV_WEB\     --to-source $SRV_WEB_PUB
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Si vous souhaitez que tous les surfeurs passent par votre cache Internet et que tous les mails circulent via la passerelle de messagerie, ce n'est pas un problème. Il n'y aura même pas à configurer les clients. Cela est particulièrement utilisé pour des entreprises ayant un antivirus sur le relais de messagerie.

```
    iptables -t nat -A PREROUTING -j DNAT -i eth2 -p tcp --dport\ http --to-destination 192.168.1.3:3128
    iptables -t nat -A PREROUTING -j DNAT -i eth2 -p tcp --dport\ smtp --to-destination 192.168.1.1
```

NB : Pensez à configurer Squid en proxy transparent avec en particulier httpd_accel_uses_host_header on.

Bloquer les scans SSH

Si votre machine est connectée à Internet, il est vraisemblable que chaque jour vous soyez victime d'un scan de port SSH. Voici ce que j'ai pu observer dans les logs de mon PC :

```
Mar 21 12:07:11 adsl sshd[9463]: Failed password for root from 217.141.18.137 port 58674 ssh2
Mar 21 12:07:13 adsl sshd[9469]: Invalid user postgres from 217.141.18.137
Mar 21 12:07:13 adsl sshd[9469]: error: Could not get shadow information for NOUSER
Mar 21 12:07:13 adsl sshd[9469]: Failed password for invalid user postgres from 217.141.18.137 port 58748 ssh2
Mar 21 12:07:16 adsl sshd[9475]: Invalid user accept from 217.141.18.137
Mar 21 12:07:16 adsl sshd[9475]: error: Could not get shadow information for NOUSER
Mar 21 12:07:16 adsl sshd[9475]: Failed password for invalid user accept from 217.141.18.137 port 58828 ssh2
Mar 21 12:07:18 adsl sshd[9481]: Invalid user leo from 217.141.18.137
Mar 21 12:07:18 adsl sshd[9481]: error: Could not get shadow information for NOUSER
Mar 21 12:07:18 adsl sshd[9481]: Failed password for invalid user leo from 217.141.18.137 port 58906 ssh2
Mar 21 12:07:20 adsl sshd[9487]: Invalid user zeppelin from 217.141.18.137
Mar 21 12:07:20 adsl sshd[9487]: error: Could not get shadow information for NOUSER
Mar 21 12:07:20 adsl sshd[9487]: Failed password for invalid user zeppelin from 217.141.18.137 port 58982 ssh2
Mar 21 12:07:22 adsl sshd[9493]: Invalid user hacker from 217.141.18.137
Mar 21 12:07:22 adsl sshd[9493]: error: Could not get shadow information for NOUSER
```

Pour freiner ces scans, il est possible d'utiliser Netfilter. Dès qu'une connexion SSH est initiée, on enregistre l'adresse IP source dans une liste, ici la liste SSH.

On utilisera cette liste pour bloquer temporairement cette adresse IP. Comme une adresse IP peut être usurpée, nous prenons soin de valider précédemment les connexions venant d'un réseau de confiance, ainsi nous sommes sûrs de ne jamais le bloquer à tort. Nous serions alors victime d'un Déni de Service (DoS).

```
 iptables -A bad-me -p tcp --dport ssh -s $TRUSTED_NET -j ACCEPT
iptables -A bad-me -p tcp --dport ssh -m recent --update --seconds\ 60 --hitcount 4 --name SSH -j LOG --log-prefix «SSH_brute_force «
iptables -A bad-me -p tcp --dport ssh -m recent --update --seconds\ 60 --hitcount 4 --name SSH -j DROP
iptables -A bad-me -p tcp --dport ssh -m recent --set --name SSH
iptables -A bad-me -p TCP --destination-port ssh  -j ACCEPT
```

Si 4 connexions ont déjà été vues pendant les 60 secondes précédentes, alors le firewall enregistre l'événement dans les logs et bloque la connexion. Sinon la connexion est enregistrée puis autorisée.

```
Pirate --> Serveur      1ere Connexion SSH établie _
Pirate --> Serveur      2eme Connexion SSH établie |
Pirate --> Serveur      3eme Connexion SSH établie | 60 secondes
Pirate --> Serveur      4eme Connexion SSH établie |
Pirate --> Serveur      5eme Connexion SSH bloquée-|
```

A ce sujet, quelques erreurs souvent trouvées sur Internet :

- Placez toutes les IP sources dans la liste puis retirez celles correspondant à la whitelist : ce n'est pas performant, autant ne pas mettre les adresses IP dans la liste dans ce cas-là.
- Placez toutes les IP sources dans la liste avant de vérifier le hitcount : la mise à jour a lieu deux fois. On va utiliser plus de ressource mémoire en comptabilisant systématiquement un enregistrement de trop.
- Utiliser ~~--ttl~~ pour vérifier le TTL dans le but de ne pas bloquer une adresse IP valide si celle-ci est spoofée : le problème est qu'il devient alors trivial de contourner le module et de réaliser l'attaque par brute force dont on cherche à se prémunir. Il suffit au pirate d'envoyer un paquet avec un TTL différent régulièrement entre les tentatives de login ~~hping2 -S -p 22 -c 1 --ttl 123 cible~~.

```
Pirate --> Serveur      1ere Connexion SSH établie TTL reçu=61
Pirate --> Serveur      2eme Connexion SSH établie TTL reçu=61
Pirate --> Serveur      3eme Connexion SSH établie TTL reçu=61
Pirate --> Serveur      4eme Connexion SSH établie TTL reçu=61
Pirate --> Serveur      5eme Connexion SSH bloquée TTL reçu=61
Pirate --> Serveur      Utilisation de hping, TTL reçu=120, paquet autorisée
Pirate --> Serveur      6eme Connexion SSH établie TTL reçu=61
```

Une autre solution est de mettre son serveur ssh sur un port non standard. Mais il faut être clair, un pirate n'a aucune chance de trouver le mot de passe par une attaque par brute force s'il ignore votre login (~~PermitRootLogin~~ No désactive les connexions au compte root.) et votre mot de passe ne figure pas dans un dictionnaire. De plus, vous pouvez aussi utiliser uniquement une authentification par clé (~~PasswordAuthentication No~~).

```
[root@christophe ~]# sh -x /etc/init.d/fw
...
+ iptables -A FORWARD -m limit --limit 4/s -j LOG --log-prefix 'FORWARD bad '
+ iptables -A FORWARD-j DROP
Bad argument `DROP'
Try `iptables -h' or 'iptables --help' for more information.
+ exit 0
```

Astuces

Pour débugger le script de firewall, lancez ~~sh -x /etc/init.d/fw~~. Les erreurs seront plus faciles à identifier.

Conclusion

Netfilter est un firewall très puissant capable de rivaliser en performance et en efficacité avec de nombreux firewalls commerciaux.

Malheureusement, le projet haute-disponibilité ~~netfilter ha~~ et le module de synchronisation de la table des connexions ~~ct_sync~~ ont peu évolué l'an dernier et Netfilter ne dispose toujours pas aujourd'hui d'une solution de firewalling actif/passif considérée comme stable.