LinuxFocus article number 289
http://linuxfocus.org

Le filtrage de paquets sous Linux

par Vincent Renardias
<vincent(at)renardias.com>

L´auteur:

Utilisateur de GNU/Linux depuis 1993, Vincent Renardias a commencé a s'impliquer activement dans son développement à partir de 1996 : Développeur de la distribution Debian, auteur de la traduction Française de The GIMP et de l'environnement GNOME, créateur du groupe d'utilisateurs Linux de Marseille (PLUG),... Actuellement directeur R&D de la société EFB2, il continue à contribuer activement au système GNU/Linux.

Résumé:

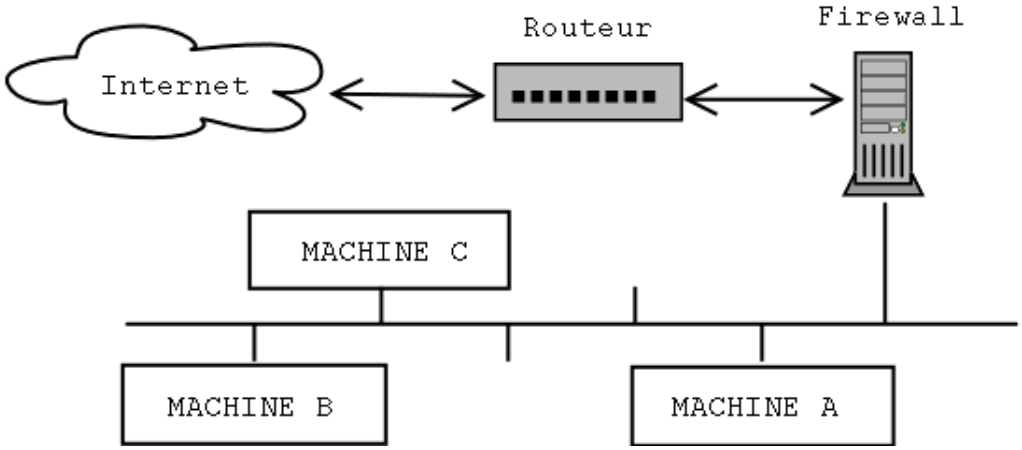
Cet article a été publié dans un numéro spécial sur la sécurité de Linux Magazine France. L'éditeur, les auteurs, les traducteurs ont aimablement accepté que tous les articles de ce numéro hors-série soient publiés dans LinuxFocus. En conséquence, LinuxFocus vous "offrira" ces articles au fur et à mesure de leur traduction en Anglais. Merci à toutes les personnes qui se sont investies dans ce travail. Ce résumé sera reproduit pour chaque article ayant la même origine.

Un des meilleurs moyens pour éviter les tentatives de piratage reste encore de filtrer dès l'entrée du réseau tout ce qui n'est pas sensé y être utile. Cette tâche est généralement remplie par une machine qui joue le rôle de "firewall" (pare-feu en français).
Dans cet article, nous allons détailler les bases indispensables pour la mise en place et la configuration d'un tel système.

Passerelle, Proxy-Arp ou Ethernet Bridge ?

Le dispositif de filtrage peut être considéré comme un filet qui va stopper un certain nombre de paquets indésirables. Tout l'art revient à trouver la taille adéquate des mailles du filet ainsi que son emplacement idéal.

Emplacement du firewall sur le réseau



Afin de pouvoir filtrer efficacement les paquets, le dispositif de filtrage doit être physiquement intercalé entre le réseau qu'il protège et le "reste du monde". Concrètement, cela se fait avec une machine possédant 2 interfaces réseau (Ethernet la plupart du temps), l'une connectée sur la partie interne du réseau et l'autre sur le routeur qui permet d'accéder à l'extérieur. De cette manière, les communications passeront obligatoirement par le firewall qui aura la charge de les bloquer ou non selon leur contenu.

La machine sur laquelle sera placé le dispositif de filtrage peut être configurée de 3 manières différentes :
- passerelle "simple" : C'est la configuration la plus courante. La machine fait office de passerelle entre deux réseaux ou sous-réseaux. Toutes les machines du réseau local devront être reconfigurées pour utiliser le firewall et non plus le routeur comme destination de la route par défaut.
- passerelle "Proxy-ARP" : La configuration précédente force le découpage du réseau en deux sous-réseaux supplémentaires, ce qui fait perdre la moitié des adresses IP disponibles ce qui est souvent gênant, étant donné qu'actuellement les adresses IP sont distribuées au compte-gouttes. Dans le cas d'un sous-réseau de 16 adresses (avec un masque de réseau sur 28 bits), seules 14 sont exploitables, car il faut retirer l'adresse de réseau et l'adresse de diffusion (broadcast). En étendant le masque de sous-réseau d'un bit, on passe de 14 à seulement 6 adresses disponibles (8 IP moins les 2 pour les adresses de réseau et de diffusion). Lorsque l'on ne peut pas se permettre de "perdre" la moitié des adresses IP disponibles, on utilise cette technique qui est détaillée un peu plus loin dans cet article. D'autre part, cette technique ne demande aucune modification des paramètres réseaux des machines existantes, aussi bien le routeur que les machines protégées.
- Ethernet bridge : En mettant une passerelle au niveau Ethernet (et non plus IP), il est possible de rendre le dispositif de filtrage totalement invisible depuis les autres machines. En effet une telle configuration peut se faire sans attribuer d'adresses IP aux interfaces Ethernet. La machine est alors indétectable par PING, Traceroute, etc.. Notons cependant que la mise en place d'un filtrage de paquets sur une telle configuration nécessite l'utilisation d'un kernel 2.2.x car le portage de cette fonctionnalité sur les noyaux 2.4.x n'est pas encore terminé.

Règles de base du filtrage

Maintenant que nous savons où placer notre filtre, il reste à déterminer ce qu'il va effectivement devoir éliminer ou au contraire accepter de laisser passer.
Il existe deux principales manières de configurer un tel filtre :
- La bonne manière : Par défaut, aucun paquet ne passe, sauf ceux qu'une règle autorise explicitement à passer.
- La mauvaise manière (malheureusement assez souvent employée) : seuls les paquets explicitement interdits sont stoppés, les autres passent.
L'explication est simple : dans le premier cas, l'oubli d'une règle de filtrage résulte en un service qui ne fonctionne pas comme il devrait, voire pas du tout. On s'en aperçoit alors généralement assez vite et il suffit d'ajouter la règle adéquate pour que tout rentre dans l'ordre.
Dans le deuxième cas, un oubli crée un problème de sécurité potentiel qui risque d'être long et difficile à mettre en évidence, si tant est qu'on le découvre.

Netfilter

Le logiciel de filtrage le plus utilisé sous Linux 2.4 est Netfilter; il remplace avantageusement 'ipchains' qui était utilisé par la version 2.2 du kernel Linux. Netfilter est composé de 2 parties : le support kernel qui doit être compilé dans votre noyau, ainsi que la commande `'iptables'` qui devrait déjà être disponible sur votre système.

Exemple de mise en place

Un exemple commenté valant mieux qu'un long discours, nous allons dans la suite de cet article décrire l'installation et la mise en place d'un dispositif de filtrage. La machine sera tout d'abord configurée comme une passerelle utilisant Proxy-ARP pour limiter l'utilisation d'adresses IP, ensuite le dispositif de filtrage sera mis en place.

L'auteur a une préférence marquée pour la distribution Debian pour effectuer ce genre d'installation, mais n'importe quelle autre distribution ferait aussi bien l'affaire.

Commencez par vous assurer que votre noyau comporte effectivement le support de Netfilter. Si c'est le cas les 2 lignes suivantes devraient être présentes dans les messages de boot :

```
ip_conntrack (4095 buckets, 32760 max)
ip_tables: (c)2000 Netfilter core team
```

Dans le cas contraire, vous devrez recompiler votre noyau après avoir activé le support Netfilter. Les options correspondantes se situent dans le sous-menu `"Network Packet Filtering"` du menu `"Networking Options"`. Dans la section `"Netfilter Configuration"` choisissez les options qui vous interessent. Dans le doute, vous pouvez les sélectionner toutes. Il est également préférable d'inclure Netfilter directement dans le noyau et de ne pas utiliser les modules. En effet, si pour une raison ou pour une autre un des modules de Netfilter venait à manquer ou n'était pas chargé, le filtrage serait inopérant avec tous les risques que cela comporte.

Vous devrez également installer le package ’iproute2’ (Celui-ci n’est pas rigoureusement indispensable, mais notre exemple va l’utiliser car il permet de simplifier le script de configuration). Sous Debian, il suffit de taper la commande ’apt-get install iproute’.
Sur les autres distributions, procurez-vous le package correspondant selon la méthode habituelle, ou installez le logiciel à partir des sources que vous pourrez télécharger à l’adresse suivante :
`ftp://ftp.inr.ac.ru/ip-routing/`

Il faut maintenant configurer les 2 cartes Ethernet. Il est important de noter que le kernel Linux, lors de l’auto-détection du matériel arrête la recherche des cartes réseaux dès qu’il en a trouvé une. Par défaut, seule la première carte sera donc détectée.
Ce problème est facilement contourné en ajoutant la ligne suivante dans le fichier `lilo.conf` :
`append="ether=0,0,eth1"`

Nous devons maintenant configurer les interfaces Ethernet. La méthode que nous avons choisi d’utiliser permet d’utiliser la même adresse IP sur les 2 cartes permettant ainsi d’économiser une adresse suplémentaire.
Dans ce qui suit, nous allons supposer que nous disposons du sous-réseau `10.1.2.96/28`, c’est-à-dire des adresses allant de 10.1.2.96 à 10.1.2.111 incluses. Le routeur aura pour adresse 10.1.2.97 et notre machine filtrante 10.1.2.98. D’autre part l’interface `eth0` sera connectée au routeur, au moyen d’un cable RJ45 croisé si les 2 cartes sont reliées directement sans passer par un hub ou un switch; l’interface `eth1` sera donc connectée au hub/switch reliant entre elles les machines du réseau local.

Chacune des 2 interfaces sera donc initialement configurée avec les paramètres suivants :

```
address  : 10.1.2.98
netmask  : 255.255.255.240
network  : 10.1.2.96
broadcast: 10.1.2.111
gateway  : 10.1.2.97
```

Ensuite on utilise le script suivant, à lancer juste après la configuration initiale des cartes réseaux pour terminer la mise en place.

```
net.vars : Variables de configuration

PREFIX=10.1.2
DMZ_ADDR=$PREFIX.96/28
# Interface definitions
BAD_IFACE=eth0
DMZ_IFACE=eth1
ROUTER=$PREFIX.97

net-config.sh : Script de configuration réseau

#!/bin/sh
# Décommenter la ligne suivante pour afficher les commandes
lors de leur exécution
# set -x
# On lit les variables définies dans le fichier précédent
source /etc/init.d/net.vars
# On enlève les routes actuelles sur le réseau local
ip route del $PREFIX.96/28 dev $BAD_IFACE
ip route del $PREFIX.96/28 dev $DMZ_IFACE
# On indique que le réseau local est accessible via eth1,
sauf
```

```
# le routeur qui est accessible via eth0.
ip route add $ROUTER dev $BAD_IFACE
ip route add $PREFIX.96/28 dev $DMZ_IFACE
# On active Proxy-ARP pour chacune des 2 interfaces
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
# On active le "forwarding" pour permettre aux paquets arrivant
sur une carte
# d'être routés vers l'autre.
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Revenons un instant sur le mécanisme Proxy-ARP qui est indispensable pour faire fonctionner notre configuration.
Lorsqu'une machine doit communiquer avec une autre sur le même réseau, on a besoin de déterminer l'adresse Ethernet (ou adresse MAC ou encore adresse hardware) correspondant à son adresse IP. La machine source envoie donc en broadcast la question "Quelle est l'adresse MAC de l'interface qui répond à l'adresse IP 1.2.3.4 ?", ce a quoi l'intéressée doit répondre.

Voici un exemple d'une telle discussion vue par tcpdump :
- la requête : la machine 172.16.6.72 demande l'adresse MAC répondant à l'adresse IP 172.16.6.10.

```
19:46:15.702516 arp who-has 172.16.6.10 tell 172.16.6.72
```

```
- la réponse : la machine 172.16.6.10 donne son numéro de carte.
19:46:15.702747 arp reply 172.16.6.10 is-at 0:a0:4b:7:43:71
```

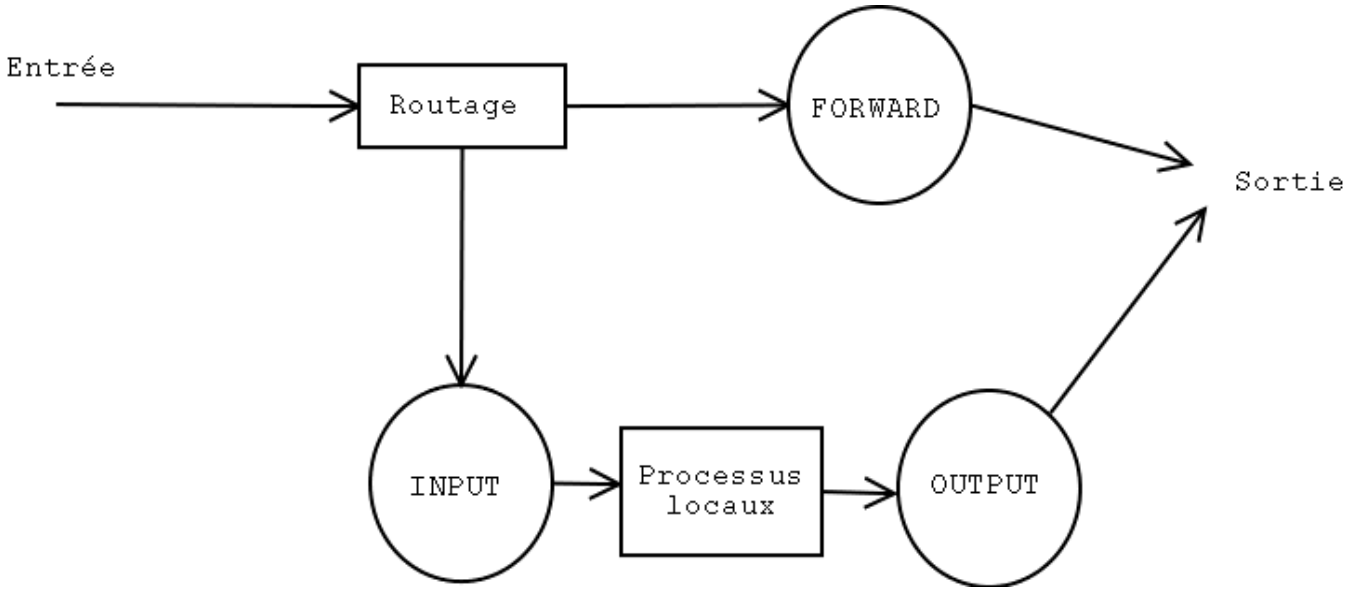
Ce qui nous amène au terme de cette petite explication : les requêtes ARP étant faites par broadcast, celles-ci sont limitées à un seul réseau physique. La requête d'une machine protégée pour trouver l'adresse MAC du routeur serait donc normalement bloquée par la machine filtrante. L'activation de la fonctionnalité Proxy-ARP permet de pallier à ce problème en demandant explicitement à ce que les requêtes et réponses ARP arrivant par une carte soient propagées vers l'autre et vice-versa.

A ce stade, vous devriez à nouveau avoir un réseau en état de fonctionnement, avec une machine supplémentaire par laquelle tout le trafic entre le réseau local et l'extérieur passe.

Il s'agit maintenant de mettre le filtrage en place grâce à Netfilter.
Netfilter permet d'agir directement sur le cheminement des paquets. Dans la configuration de base, les paquets passent par 3 chaînes de règles :
- INPUT : par laquelle passent tous les paquets entrant par une interface,
- FORWARD : par laquelle passent tous les paquets qui sont transmis d'une interface à une autre,
- OUTPUT : par laquelle passent les paquets avant de sortir par une interface.

La commande 'iptables' permet d'ajouter, modifier ou retirer des règles dans chacune de ces chaînes pour modifier le comportement du filtrage.
De plus, chaque chaîne possède une politique (policy) par défaut, c'est-à-dire le comportement à suivre lorsqu'aucune règle de la chaîne n'a correspondu au paquet.
Les quatre comportements les plus courants sont :
- ACCEPT : On laisse passer le paquet,
- REJECT : On rejette le paquet et on envoie le paquet d'erreur associé (ICMP Port Unreachable, TCP RESET, selon les cas),
- LOG : Enregistre une notification du paquet dans syslog,
- DROP : Le paquet est tout simplement ignoré et aucune réponse n'est envoyée.

Cheminement des paquets dans les chaînes standards



Voici les principales options d'iptables permettant de manipuler des chaînes entières. Nous reviendrons ensuite un peu plus en détail sur chacune d'entre elles :

-N : Création d'une nouvelle chaîne.
-X : Suppression d'une chaîne vide.
-P : Changement de la politique par défaut d'une chaîne.
-L : Liste les règles d'une chaîne.
-F : Élimine toutes les règles d'une chaîne.
-Z : Remet à zéro les compteurs d'octets et de paquets ayant traversé la chaîne.

En ce qui concerne les modifications de chaînes, les commandes suivantes sont disponibles :
-A : Ajoute une règle à la fin d'une chaîne.
-I : Insére une nouvelle règle à une position donnée dans une chaîne.
-R : Remplace une règle donnée dans une chaîne.
-D : Efface une règle dans une chaîne, soit par son numéro d'ordre, soit en décrivant la règle.

Voyons tout de suite un petit exemple pratique : nous allons interdire les réponses PING (c'est-à-dire les paquets ICMP de type 'echo-reply') venant d'une machine donnée.
Commençons par nous assurer que pour le moment, on peut effectivement "pinger" la machine en question :

```
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255 time=0.6 ms
--- 172.16.6.74 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

Maintenant, nous allons ajouter une règle dans la chaîne INPUT qui va intercepter les paquets ayant

pour source la machine 172.16.6.74 (’-s 172.16.6.74’), de type ICMP-Reply (’-p icmp --icmp-type echo-reply’). Ces paquets seront tout simplement ignorés (’-j DROP’).

```
# iptables -A INPUT -s 172.16.6.74 -p icmp --icmp-type echo-reply -j DROP
```

Maintenant essayons à nouveau un PING vers cette machine :

```
# ping -c 3 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
--- 172.16.6.74 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

Comme nous pouvions nous y attendre, cette fois les réponses au PING ne passent plus. D’autre part, nous pouvons vérifier que les 3 réponses ont bien été bloquées (3 paquets, pour un total de 252 octets) :

```
# iptables -L INPUT -v
Chain INPUT (policy ACCEPT 604K packets, 482M bytes)
pkts bytes target     prot opt in    out      source       destination
 3   252   DROP       icmp --  any   any      172.16.6.74  anywhere
```

Pour revenir à la situation initiale, il nous suffit de demander la suppression de la première règle de la chaîne INPUT :

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

Et maintenant, le PING devrait à nouveau fonctionner correctement

```
:
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255 time=0.6 ms
--- 172.16.6.74 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
#
```

Ça marche !

En plus des 3 chaînes pré-existantes (qu’il est d’ailleurs impossible de supprimer), il est également possible de créer d’autres chaînes et d’y faire passer une certaine partie du trafic. Ceci est très utile pour, par exemple, éviter de dupliquer des règles communes entre plusieurs chaînes.

Attachons-nous maintenant à mettre en place les règles nécessaires pour un firewall minimaliste. Celui-ci laissera passer les services ssh, domain (DNS), http et smtp à l’exception de tous les autres. Pour simplifier les choses, les commandes pour mettre les règles en place sont enregistrées dans un script shell pour rendre la configuration plus pratique. Le script commence par enlever toute la configuration actuelle du filtrage avant de mettre la nouvelle en place. Cette petite astuce permet au script d’être "idempotent", c’est-à-dire que l’on peut le relancer même si la configuration est déjà active sans risque d’avoir de règle en double.

rc.firewall

```
#!/bin/sh
```

```
# Effacement de toutes les règles
iptables -F
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD

# On construit une chaîne par direction.
# bad = eth0 (extérieur)
# dmz = eth1 (intérieur)
iptables -X bad-dmz
iptables -N bad-dmz
iptables -X dmz-bad
iptables -N dmz-bad
iptables -X icmp-acc
iptables -N icmp-acc
iptables -X log-and-drop
iptables -N log-and-drop

# Chaîne spéciale destinée a logguer les paquets
avant de les bloquer
iptables -A log-and-drop -j LOG --log-prefix "drop "
iptables -A log-and-drop -j DROP

# On élimine les paquets ayant tous les flags TCP activés ainsi que ceux
# avec aucun flag activé (souvent utilisé par les scans de Nmap)
iptables -A FORWARD -p tcp --tcp-flags ALL ALL -j log-and-drop
iptables -A FORWARD -p tcp --tcp-flags ALL NONE -j log-and-drop

# On bloque les paquets provenant des classes d’adresses réservées
# ainsi que le multicast
iptables -A FORWARD -i eth+ -s 224.0.0.0/4 -j log-and-drop
iptables -A FORWARD -i eth+ -s 192.168.0.0/16 -j log-and-drop
iptables -A FORWARD -i eth+ -s 172.16.0.0/12 -j log-and-drop
iptables -A FORWARD -i eth+ -s 10.0.0.0/8 -j log-and-drop

# On accepte les paquets appartenants à une connexion déjà établie
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
# Selon le sens d’arrivée des paquets on transmet à la chaîne correspondante
iptables -A FORWARD -s $DMZ_ADDR -i $DMZ_IFACE -o $BAD_IFACE -j dmz-bad
iptables -A FORWARD -o $DMZ_IFACE -j bad-dmz
# On ignore tout le reste
iptables -A FORWARD -j log-and-drop

# ICMPs acceptés
iptables -A icmp-acc -p icmp --icmp-type destination-unreachable -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type source-quench -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type time-exceeded -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type echo-request -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A icmp-acc -j log-and-drop

# Chaîne Extérieur -> Intérieur
# On accepte les services mail, DNS, http(s) et SSH
iptables -A bad-dmz -p tcp --dport smtp -j ACCEPT
iptables -A bad-dmz -p udp --dport domain -j ACCEPT
iptables -A bad-dmz -p tcp --dport domain -j ACCEPT
iptables -A bad-dmz -p tcp --dport www -j ACCEPT
iptables -A bad-dmz -p tcp --dport https -j ACCEPT
iptables -A bad-dmz -p tcp --dport ssh -j ACCEPT
iptables -A bad-dmz -p icmp -j icmp-acc
iptables -A bad-dmz -j log-and-drop
```

```
# Chaîne Intérieur -> Extérieur
# On accepte les services mail, DNS, http(s) et telnet
iptables -A dmz-bad -p tcp --dport smtp -j ACCEPT
iptables -A dmz-bad -p tcp --sport smtp -j ACCEPT
iptables -A dmz-bad -p udp --dport domain -j ACCEPT
iptables -A dmz-bad -p tcp --dport domain -j ACCEPT
iptables -A dmz-bad -p tcp --dport www -j ACCEPT
iptables -A dmz-bad -p tcp --dport https -j ACCEPT
iptables -A dmz-bad -p tcp --dport telnet -j ACCEPT
iptables -A dmz-bad -p icmp -j icmp-acc
iptables -A dmz-bad -j log-and-drop

# Chaines pour la machine elle-même
iptables -N bad-if
iptables -N dmz-if
iptables -A INPUT -i $BAD_IFACE -j bad-if
iptables -A INPUT -i $DMZ_IFACE -j dmz-if

# Interface externe
# On accepte uniquement SSH sur la machine elle-même
iptables -A bad-if -p icmp -j icmp-acc
iptables -A bad-if -p tcp --dport ssh -j ACCEPT
iptables -A bad-if -p tcp --sport ssh -j ACCEPT
ipchains -A bad-if -j log-and-drop

# Interface Interne
iptables -A dmz-if -p icmp -j icmp-acc
iptables -A dmz-if -j ACCEPT
```

Un dernier point sur la qualité de service. Linux peut modifier le champ ToS ("Type of Service") des paquets et éventuellement modifier sa valeur pour rendre le paquet plus ou moins prioritaire. Par exemple, la commande suivante modifie les paquets ssh sortants afin d'améliorer la réactivité des connexions.

```
iptables -A OUTPUT -t mangle -p tcp --dport ssh -j TOS --set-tos Minimize-Delay
```

De même, pour les connexions FTP vous pourrez utiliser l'option `'--set-tos Maximize-Throughput'` pour améliorer le débit au prix d'une possible détérioration de l'interactivité de la session.

Voilà. Maintenant, vous êtes en possession des bases pour mettre en place un filtrage de paquets efficace. Gardez cependant en tête qu'un firewall n'est pas la panacée en matière de sécurité, mais simplement une précaution supplémentaire. La mise en place d'un dispositif de filtrage ne dispense en rien l'utilisation de mots de passe non triviaux, de logiciels équipés des derniers patchs de sécurité, de la mise en place d'un dispositif de détection d'intrusion, etc...

Références

- Proxy-ARP Mini-HOWTO: http://www.linuxdoc.org/HOWTO/mini/Proxy-ARP-Subnet/index.html
- Netfilter: http://netfilter.samba.org/

Site Web maintenu par l´équipe d´édition
LinuxFocus

http://www.LinuxFocus.org

Translation information:
fr --> -- : Vincent Renardias <vincent(at)renardias.com>

2005-01-14, generated by lfparser_pdf version 2.51