Administration Linux - Pare-feu

Sommaire

*Un compte-rendu au format texte (**UTF-8**) devra être rédigé et envoyé à l'adresse*
tvaira@free.fr
La convention de nommage pour les compte-rendus est la suivante :
admin-linux-parefeu-nom.txt

Mise en situation

Vous devez disposer d'un PC possédant un système d'exploitation Linux ou Windows et du logiciel de virtualisation *VirtualBox*. Le système invité sera une installation du **serveur Ubuntu 12.10**.

Pare-feu (*firewall*)

Un système pare-feu (*firewall*) est un dispositif conçu pour examiner et éventuellement bloquer les échanges de données entre réseaux.

Le pare-feu joue le rôle de filtre et peut donc intervenir à plusieurs niveaux du modèle à couches.

Il existe trois types principaux de pare-feu :
- filtrage de paquets (*firewall stateless*)
- filtrage de paquets avec état (*firewall stateful*)
- *proxy* applicatif

Filtrage de paquets (*firewall stateless*)

Les pare-feu de filtrage de paquets sont généralement des routeurs qui permettent d'accorder ou de refuser l'accès en fonctions des éléments suivants :

- l'adresse source, l'adresse destination
- le numéro de port
- le protocole

Les limites de cette technique :
- Manque de souplesse : Si cette approche offre une défense simple et efficace, elle manque de souplesse pour que sa mise en place en protection d'un réseau, dans la majorité des cas, ne s'avère pas bloquante pour le bon fonctionnement des systèmes du réseau concerné. Pour contourner ce défaut, l'administrateur est rapidement contraint à autoriser trop d'accès pour que son pare-feu offre encore la moindre protection réelle.
- Difficulté pour gérer certains protocoles : Certains protocoles sont particulièrement délicats à gérer à l'aide de cette technique comme FTP (le suivi des échanges FTP est une opération complexe) ou TCP (protocole de type connecté).

Filtrage de paquets avec état (*firewall stateful*)

La technologie *stateful* est l'une des deux réponses possibles aux limites du filtre de paquets.

Un *firewall stateful* inclut toutes les fonctionnalités d'un filtrage de paquet, auxquelles il ajoute la capacité de conserver la trace des sessions et des connexions dans des tables d'état interne. Tout échange de données est soumis à son approbation et adapte son comportement en fonction des états.

Cette technique convient aux protocoles de type connecté (TCP). Certains protocoles (UDP et ICMP) posent un problème supplémentaire : aucune notion de connexion n'y est associée. Le pare-feu est donc amené à examiner les paquets, et peut seulement gérer des *timeout*, souvent de l'ordre d'une minute.

Les limites de cette technique :
- Risque d'accès illimité : Si cette approche apporte une amélioration certaine par rapport à la technique du filtrage simple de paquets, elle se borne cependant à autoriser ou interdire l'accès à un service donné. Dès lors que l'accès à un service est accordé, celui-ci est illimité.

– Faille interne au *firewall* : D'autre part, un tel système ne protège aucunement un serveur contre les attaques s'appuyant sur des failles du logiciel utilisé pour fournir le service.

Néanmoins, le pare-feu de type *stateful* est considéré, par les administrateurs réseau, comme la technologie minimum pour une sécurisation.

Proxy applicatif

Les *firewalls* de type *proxy* applicatif (ou passerelle applicative) ont pour ambition de répondre aux problèmes soulevés par les *firewall stateless* et les *firewall stateful*.

Ces systèmes se substituent au serveur ou au client qu'ils ont pour mission de défendre pour :
– traiter les requêtes et réponses à la place du système à protéger,
– les transmettre, après d'éventuelles modifications
– ou les bloquer

Le pare-feu de ce type joue le rôle de canal et d'interpréteur en agissant aux niveaux des protocoles de la couche Application. Cette approche permet, en principe, d'atteindre le plus haut niveau de sécurité.

De nombreux *firewalls* de ce type sont disponible sur le marché, comme Raptor de Axent, Gauntlet de NAI, racheté par Secure Computing, Sidewinder de Secure Computing, et M-Wall de Matranet, pour ne citer que les plus connus.

Cette technologie, bien que prometteuse, est cependant confrontée à certaines difficultés.

Les limites de cette technique :
– Adapatabilité : En premier lieu, tout protocole transitant par le pare-feu doit être connu de celui- ci, pour pouvoir bénéficier de ses capacités de *proxy*, ce qui exclut l'usage d'applications et protocoles "maison", ou plus simplement peu répandus.
– Difficulté : En outre, la gamme de protocoles à examiner étant particulièrement vaste, il est extrêmement délicat d'obtenir un système éliminant réellement toutes les attaques envisageables, les développeurs de *firewalls* applicatifs ne pouvant que difficilement maîtriser tous ces protocoles.
– Failles du *proxy* : Un tel système a pour rôle, comme les clients et serveurs qu'il défend, d'interpréter, comprendre, et réécrire les requêtes et réponses qu'il reçoit. Or, c'est précisément ce type de fonctionnalités qui est à l'origine d'une vaste majorité des failles applicatives.
– Performance : Enfin, le gain de sécurité obtenu à l'aide du *proxy* applicatif se paie en termes de performances. Les tâches que ce système a pour rôle de réaliser étant nettement plus complexes que celles du *firewall stateful*, une machine puissante est nécessaire pour faire tourner ce type de logiciel, et il est rare que des débits réseau supérieurs au cinquième des débits gérés par un *firewall stateful* puissent être traités.

Autres possibilités des pare-feu

Les fabricants de pare-feu ont tendance à intégrer un maximum de fonctionnalités :
– filtrage de contenu (URL, *spam mails*, code ActiveX, applets Java, ...)
– réseau virtuel privé (VPN) : les VPN permettent de canaliser un trafic sécurisé d'un point à un autre sur des réseaux généralement hostile (Internet par exemple). Checkpoint et Cisco intègrent des services VPN à leur offres de pare-feu.
– la traduction d'adresse réseau NAT (*Network Address Translation*) : ce service permet de mettre en correspondance des adresses réservées ou illégales avec des adresses valides. Les premiers dispositifs NAT qui apparaissent en entreprise sont souvent des produits pare-feu.
– l'équilibrage de charge : va permettre de segmenter un trafic de façon répartie (orienter le trafic Web et FTP par exemple)

– la tolérance de pannes : certains pare-feu, comme CISCO/PIX ou Nokia/Checkpoint supportent ces fonctionnalités (généralement exécution des pare-feu par paire pour permettre une disponibilité élevée (*High-Availability*)
– la détection des intrusions (IDS)

Politique de sécurité

De manière générale, on distingue deux types de politique de sécurité :

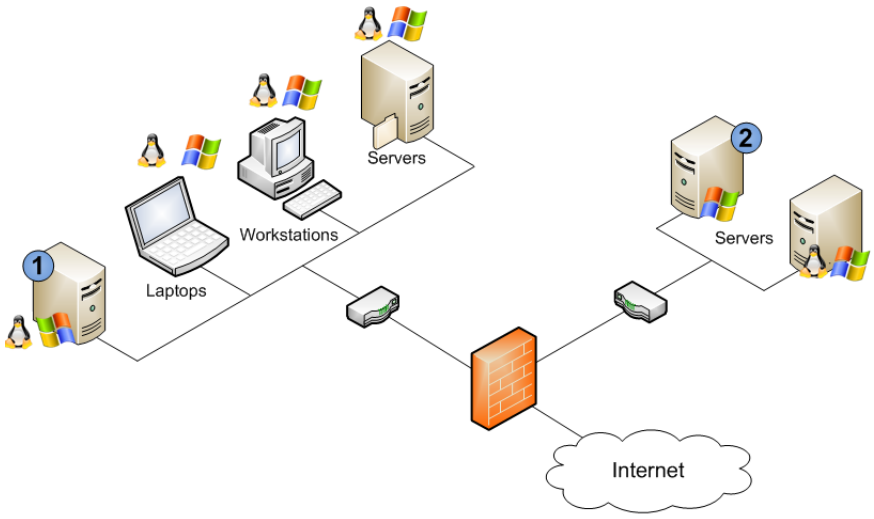
– politique permissive (*open config*) : Cette politique repose sur le principe que par défaut on laisse tout passer puis on va restreindre pas à pas les accès et les services mais la sécurité risque d'avoir des failles.
– politique stricte (*close config*) : Cette politique repose sur le principe inverse. On commence par tout interdire, puis on décide de laisser seulement passer les services ou adresses désirés ou indispensables. La sécurité sera meilleure mais le travail sera plus difficile et cela peut même bloquer plus longtemps que prévu les utilisateurs. C'est évidemment la politique conseillée pour un pare-feu.

Architecture réseau

Il existe plusieurs zones de sécurité commune aux réseaux. Ces zones déterminent un niveau de sécurité en fonction des accès réseaux et donnent les bases de l'architecture.

On considère en général trois zones ou réseaux :

– réseaux externes (*extranet*) : C'est le réseau généralement le plus ouvert (Internet par exemple). L'entreprise n'a pas ou très peu de contrôle sur les informations, les systèmes et les équipements qui se trouvent dans ce domaine.
– réseaux internes (*intranet*) : Les éléments de ce réseau doivent être sérieusement protégés. C'est souvent dans cette zone que l'on trouve les mesures de sécurité les plus restrictives et c'est donc le réseau le moins ouvert.
– réseaux intermédiaires (*dmz*) : Cette zone est un compromis entre les deux précédentes. Ce réseau est composé de services fournis aux réseaux internes et externes. Les services publiquement accessibles (serveurs de messagerie, Web, FTP et DNS le plus souvent) sont destinés aux utilisateurs internes et aux utilisateurs par Internet. Cette zone, appelée réseau de services ou de **zone démilitarisée DMZ** (*De-Militarized Zone*), est considérée comme la zone moins protégée de tout le réseau.



Iptables

Introduction

`iptables` est un logiciel libre de l'espace utilisateur Linux grâce auquel l'administrateur système peut configurer les chaînes et règles dans le pare-feu en espace noyau (et qui est composé par des modules `Netfilter`).

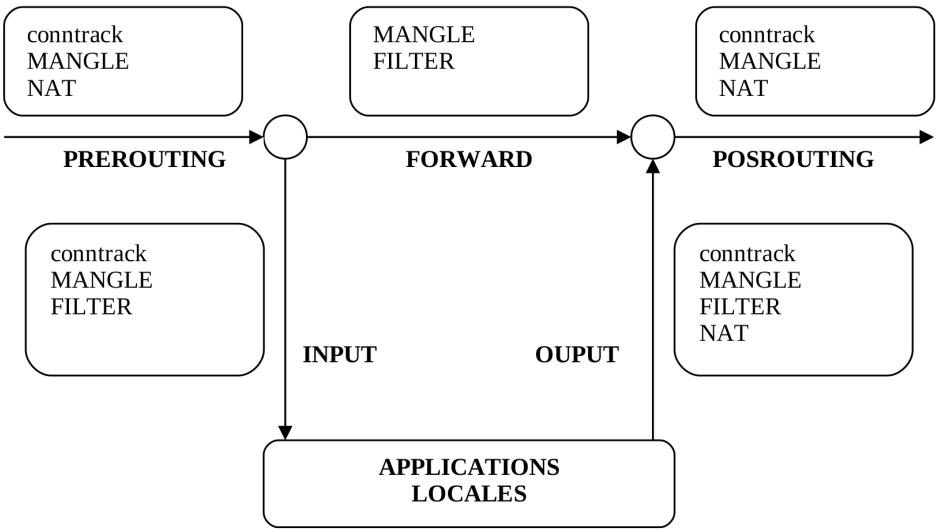
Différents programmes sont utilisés selon le protocole employé : `iptables` est utilisé pour le protocole **IPv4**, `ip6tables` pour **IPv6**, `arptables` pour **ARP** (*Address Resolution Protocol*) ou encore `ebtables`, spécifique aux trames *Ethernet*.

Site officiel : http ://netfilter.org/

Principe

Iptables/Netfilter fonctionne selon un système de tables :
- Les tables sont composées d'un nombre arbitraire et non limité de chaînes.
- Une chaîne est une suite linéaire de règles.
- Une règle est constituée d'un motif (*pattern*) destiné à reconnaître des paquets selon un nombre indéterminé de critères (*matches*) et d'une décision, appelée cible (*target*), à prendre en cas de reconnaissance du paquet.

Il existe à ce jour cinq tables dont les trois principales sont :
- La table `FILTER` : elle permet les opérations de filtrage IP. Les paquets y sont acceptés (`ACCEPT`), refusés (`DROP` ou `REJECT` avec renvoi d'un paquet erreur), logués (`LOG`) ou encore mis en queue (`QUEUE`), mais jamais modifiés. Cette table possède trois chaînes de base : `INPUT`, `FORWARD` et `OUTPUT`.
- La table `NAT` (*Network Address Translation*) : elle permet les opérations de traduction d'adresses. Cette table contient les chaînes : `PREROUTING`, `INPUT`, `OUTPUT` et `POSTROUTING`. Cette table dispose de cibles propres (`SNAT`, `DNAT`, `MASQUERADE` et `REDIRECT`) pour la mise en oeuvre de NAT.
- La table `MANGLE` : elle permet d'altérer les en-têtes des paquets. La table MANGLE modifie la structure qui représente le paquet dans la couche réseau du noyau. Il est donc possible d'agir sur les en-têtes mais aussi sur les champs des structures utilisées par le système. Cette table possède les cinq chaînes de base : `PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT` et `POSTROUTING`.



Installation

Il suffit d'installer le paquet `iptables`, mais normalement celui-ci est déjà installé :

```
# apt-get install iptables
...

// Vérifier si le paquet est installé
# dpkg --list iptables
Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé/W=attend-traitement-
    déclenchements
|/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais)
||/ Nom                                Version              Architecture          Description
+++-====================================-====================-====================-==========

ii  iptables                           1.4.12-2ubuntu2      amd64                 administration
    tools for packet filtering and NAT

# dpkg --status iptables
Package: iptables
Status: install ok installed
Priority: important
Section: net
Installed-Size: 1422
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Version: 1.4.12-2ubuntu2
Depends: libc6 (>= 2.14), libnfnetlink0 (>= 1.0.0)
Description: administration tools for packet filtering and NAT
...

# dpkg --get-selections iptables
iptables                                        install
```

Configuration par défaut

Par défaut, `iptables` est réglé sur une politique permissive (*open config*) où on laisse tout passer (*policy* `ACCEPT`).

```
// La table FILTER
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

// La table NAT
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```



```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

// La table MANGLE
# iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
```

On rappelle que la table `FILTER` contient trois chaînes de base :
- INPUT : les paquets à destination de la machine
- FORWARD : les paquets traversant (routés par) la machine
- OUTPUT : les paquets émis par la machine

Tout paquet traité par la table `FILTER` traversera une et une seule de ces trois chaînes.

Tests

On interdit tout ce qui arrive sur l'interface `lo` (*loopback*) :

```
// Ajout d'une règle à la chaîne INPUT de la table FILTER
# iptables -A INPUT -i lo -j DROP

# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target     prot opt source               destination
1   DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
num target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num target     prot opt source               destination

// On teste :
# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
^C
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

// On visualise les 2 paquets rejetés :
# iptables -L -v
Chain INPUT (policy ACCEPT 19 packets, 1420 bytes)
```



```
 pkts bytes target   prot opt in    out    source              destination
    2   168 DROP     all  --  lo    any    anywhere            anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target   prot opt in    out    source              destination

Chain OUTPUT (policy ACCEPT 12 packets, 1536 bytes)
 pkts bytes target   prot opt in    out    source              destination

// On supprime la règle
# iptables -D INPUT 1

// L'interface lo est de nouveau accessible
# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.068 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.057/0.062/0.068/0.009 ms
```

On interdit tout ce qui sort de l'interface `lo` (*loopback*) :

```
// Ajout d'une règle à la chaîne OUTPUT de la table FILTER
# iptables -A OUTPUT -o lo -j DROP

# iptables -L --line-numbers
Chain INPUT (policy ACCEPT 10 packets, 760 bytes)
num target    prot opt in    out    source              destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num target    prot opt in    out    source              destination

Chain OUTPUT (policy ACCEPT 5 packets, 596 bytes)
num target    prot opt in    out    source              destination
1   DROP      all  --  any   lo     anywhere            anywhere

// On teste :
# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1006ms

// On visualise les 2 paquets rejetés :
# iptables -L -v
Chain INPUT (policy ACCEPT 43 packets, 3038 bytes)
 pkts bytes target   prot opt in    out    source              destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target   prot opt in    out    source              destination

Chain OUTPUT (policy ACCEPT 23 packets, 3372 bytes)
 pkts bytes target   prot opt in    out    source              destination
    2   168 DROP     all  --  any   lo     anywhere            anywhere

// On supprime la règle
# iptables -D OUTPUT 1
```



```
// L'interface lo est de nouveau fonctionnelle
# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.052 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.070 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.052/0.061/0.070/0.009 ms
```

Manipulations

Séquence 1 : règles basiques

Question 1. Interdire tout ce qui rentre du réseau.

```
# iptables -A INPUT -s 0/0 -j DROP
```

Question 2. Interdire tout ce qui rentre ou qui sort du réseau `192.168.0.0/24`.

```
# iptables -A INPUT -s 192.168.0.0/24 -j DROP

# iptables -A OUTPUT -d 192.168.0.0/24 -j DROP
```

Question 3. Interdire toutes les requêtes *echo* (`ping`).

```
# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
```

Question 4. Interdire toutes les réponses *echo* (`ping`).

```
# iptables -A INPUT -p icmp --icmp-type 0 -j DROP
```

Séquence 2 : politique stricte (*close config*)

Dans cette politique, on commence par tout interdire, puis on décide de laisser seulement passer les services ou adresses désirés ou indispensables. La sécurité sera meilleure mais le travail sera plus difficile et cela peut même bloquer plus longtemps que prévu les utilisateurs.
C'est évidemment la politique conseillée pour un pare-feu.

Question 5. Effacer (*flush*) toutes les règles existantes pour les trois tables `FILTER`, `NAT` et `MANGLE`.

```
# iptables -F
# iptables -t nat -F
# iptables -t mangle -F
```

Question 6. Effacer toutes les chaînes "utilisateurs".

```
# iptables -X
# iptables -t nat -X
# iptables -t mangle -X
```

Question 7. Mettre en place les politiques *close config* par défaut pour la table `FILTER`.

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
```

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

Question 8. Logger le trafic entrant vers votre serveur sur les interfaces `lo` et `eth0`.

```
# iptables -A INPUT -i lo -j LOG
# iptables -A INPUT -i eth0 -j LOG
```

```
# tail /var/log/syslog
Mar 31 16:49:21 server-tv kernel: [ 1511.032589] IN=eth0 OUT= MAC=08:00:27:4a:b6:18:00:4f:4e
    :08:25:1a:08:00 SRC=192.168.52.2 DST=192.168.52.9 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=38250 DF
    PROTO=TCP SPT=56594 DPT=80 WINDOW=14600 RES=0x00 SYN URGP=0
...
```

Question 9. Autoriser maintenant les connexions `ssh` vers ce serveur sur l'interface `eth0`.

```
# iptables -A INPUT -p TCP --dport ssh -j ACCEPT
```

Question 10. Interdire le trafic entrant vers votre serveur `http`.

```
# iptables -A INPUT -p TCP --dport http -j DROP
```

Question 11. Autoriser le flux entrant de l'interface *loopback* (`lo`).

```
# iptables -A INPUT -i lo -j ACCEPT
```

```
# iptables -L --line-numbers -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1       55  7706 LOG        all  --  lo     any     anywhere             anywhere            LOG level
    warning
2      352 26028 LOG        all  --  eth0   any     anywhere             anywhere            LOG level
    warning
3       76  5392 ACCEPT     tcp  --  eth0   any     anywhere             anywhere            tcp dpt:ssh
4        6   360 DROP       tcp  --  any    any     anywhere             anywhere            tcp dpt:http
5       49  7346 ACCEPT     all  --  lo     any     anywhere             anywhere
```



```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target   prot opt in    out    source            destination

Chain OUTPUT (policy ACCEPT 4 packets, 464 bytes)
num   pkts bytes target   prot opt in    out    source            destination
```

Question 12. Proposer des tests des règles ci-dessus.

Question 13. Commenter les règles ci-dessous.

```
# iptables -P OUTPUT ACCEPT

# iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

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

# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# iptables -A OUTPUT -m state ! --state INVALID -j ACCEPT

# iptables -I INPUT -i lo -j ACCEPT

# iptables -A INPUT -j LOG

# iptables -A OUTPUT -p TCP --dport domain -j ACCEPT
# iptables --A OUTPUT -p UDP --dport domain -j ACCEPT

# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source $INTER_IP

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

# iptables -t nat -A PREROUTING -j DNAT -i eth0 -p TCP --dport http --to-destination $HTTP_IP
```

Séquence 3 : les chaînes utilisateurs

Création des chaînes utilisateurs (par exemple pour la table FILTER) :

```
# iptables -N nom_chaîne
```

Insertion d'une règle dans une chaîne utilisateur :

```
# iptables -A nom_chaîne -m state --state INVALID -j DROP
```

Les politiques par défaut ne s'applique pas pour les chaînes utilisateurs. On peut par contre en émuler une en plaçant une règle en fin de chaîne :

```
# iptables -A nom_chaîne -j DROP
```

Utilisation d'une chaîne utilisateur comme cible :

```
# iptables -A INPUT -i eth0 -j nom_chaîne
```

```
// cf. rc.firewall.txt
# iptables -A INPUT -p tcp -j bad_tcp_packets
# iptables -A INPUT -p all -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
# iptables -A INPUT -p udp -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT
```

```
# iptables -A INPUT -p all -d $INET_IP -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -p tcp -i $INET_IFACE -j tcp_packets
# iptables -A INPUT -p udp -i $INET_IFACE -j udp_packets
# iptables -A INPUT -p icmp -i $INET_IFACE -j icmp_packets
# iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level DEBUG --log-prefix
    "IPT INPUT packet died: "
```

Question 14. Créer les chaînes "utilisateurs" `bad_tcp_packets` et tester.

```
# iptables -N bad_tcp_packets
# iptables -N allowed
# iptables -N tcp_packets
# iptables -N udp_packets
# iptables -N icmp_packets
```

Séquence 4 : les règles au démarrage

Depuis au moins Ubuntu 12.04, le paquet `iptables-persitent` gère les règles au démarrage. Il propose de sauvegarder les règles dans le dossier `/etc/iptables` :
– fichier `rules.v4` pour les règles IPv4 et
– fichier `rules.v6` pour les règles IPv6.

Le script peut s'appeler via :

```
# service iptables-persistent
```

Il prend les arguments : `save` pour sauvegarder les règles, `flush` pour vider toutes les règles et `reload` pour les recharger depuis les fichiers précités.

Séquence 5 : UFW (*Uncomplicated FireWall*)

`UFW` (*Uncomplicated FireWall*) est un nouvel outil de configuration simplifié en ligne de commande de Netfilter, qui donne une alternative à l'outil `iptables`. Il existe aussi `Gufw` qui est une interface graphique pour `ufw`.

Uncomplicated Firewall est pré-installé sous Ubuntu, mais si besoin il faudrait installer le paquet `ufw`.

```
# ufw status verbose
État : inactif
```

UFW n'est pas activé par défaut, il faut donc l'activer :

```
// USAGE: ufw [--dry-run] enable|disable|reload

// Désactiver UFW :
# ufw disable
Le pare-feu est arrêté et désactivé lors du démarrage du système

// Activer UFW :
# ufw enable
Le pare-feu est actif et lancé au démarrage du système
```

```
# ufw status verbose
État : actif
Journalisation : on (low)
Par défaut : deny (entrant), allow (sortant)
...
```

Il est possible d'utiliser les règles par défaut :

```
// USAGE: ufw default allow|deny|reject [incoming|outgoing]

// politique close config :
# ufw default deny incoming
La stratégie par défaut pour le sens « incoming » a été remplacée par « deny »
(veillez à mettre à jour vos règles en conséquence)

# ufw default deny outgoing
La stratégie par défaut pour le sens « outgoing » a été remplacée par « deny »
(veillez à mettre à jour vos règles en conséquence)

# ufw status verbose
État : actif
Journalisation : on (low)
Par défaut : deny (entrant), deny (sortant)

// politique open config :
# ufw default allow incoming
La stratégie par défaut pour le sens « incoming » a été remplacée par « allow »
(veillez à mettre à jour vos règles en conséquence)

# ufw default allow outgoing
La stratégie par défaut pour le sens « outgoing » a été remplacée par « allow »
(veillez à mettre à jour vos règles en conséquence)

# ufw status verbose
État : actif
Journalisation : on (low)
Par défaut : allow (entrant), allow (sortant)
```

Et pour gérer la journalisation :

```
// Activer la journalisation :
# ufw logging on
Journalisation activée

// Désactiver la journalisation :
# ufw logging off
Journalisation désactivée
```

L'ordre de déclaration des règles est très important, le système utilisant une politique « premier arrivé, premier servi ». Prenez donc soin d'ajouter vos règles spécifiques avant les règles générales lorsqu'elles concernent des éléments communs.

Les exemples ci-dessous montrent l'utilisation de règles simples, par défaut les règles s'appliquent sur le trafic entrant (***in**coming*) :

```
// USAGE: ufw [insert NUM] allow|deny|reject|limit [in|out] [log|log-all] PORT[/protocol]
```



```
// Ouverture du port 22 en TCP et UDP :
# ufw allow 22

// Ouverture du port 22 en TCP uniquement :
# ufw allow 22/tcp

// Ouverture du service ssh (port 22 en TCP et UDP) :
# ufw allow ssh

// Autorise les requêtes DNS (port 53 en TCP et UDP) en sortie :
# ufw allow out domain

// Visualisation des règles et de leurs numéros
# ufw status numbered
État : actif

     Vers                       Action     Depuis
     ----                       ------     ------
[ 1] 22                         ALLOW IN   Anywhere
[ 2] 53                         ALLOW OUT  Anywhere (out)
[ 3] 22                         ALLOW IN   Anywhere (v6)
[ 4] 53                         ALLOW OUT  Anywhere (v6) (out)

// Suppression de la règle 1
# ufw delete 1
Suppression de :
 allow 22
Exécuter l'opération (o|n) ? o
La règle a été supprimée

...
```

UFW regarde dans la liste de services connus (`/etc/services`) pour appliquer les règles standards associées à des services.

Et quelques règles plus complexes :

```
// Autoriser les accès du réseau local 10.0.0.0 :
# ufw allow from 10.0.0.0/8

// Interdire les accès au port 5000 pour la machine 192.168.0.1
# ufw deny proto udp from 192.168.0.1 to any port 5000

# ufw status numbered
État : actif

     Vers                       Action     Depuis
     ----                       ------     ------
[ 1] Anywhere                   ALLOW IN   10.0.0.0/8
[ 2] 5000/udp                   DENY IN    192.168.0.1
```

UFW applique des règles `iptables` par défaut lors de son lancement. Vous pouvez les consulter et les modifier directement dans le fichier `/etc/ufw/before.rules`. Par exemple, cela peut s'avérer utile si vous voulez interdire les requêtes de `ping` (`ICMP` *Echo Request*).

Question 15. Mettre en place une politique *close config* par défaut.

```
// politique close config :
# ufw default deny incoming
# ufw default deny outgoing
```

Question 16. Autoriser maintenant les connexions `http` vers ce serveur.

```
# ufw allow http
```

Question 17. Autoriser seulement le trafic local du réseau `192.168.0.0/24`.

```
# ufw allow to 192.168.0.0/24 from 192.168.0.0/24
# ufw allow out to 192.168.0.0/24 from 192.168.0.0/24
```

Question 18. Supprimer la règle autorisant les connexions `http` vers ce serveur.

