Sécurité/Firewall avec Netfilter et GNU/Linux

Retour

Fermer

Centre Paris Pleyel
153 bd Anatole France
93200 Saint-Denis, France

Tél. : + 33 1 49 22 68 00
Fax : + 33 1 49 22 68 01
E-mail : alcove@fr.alcove.com, Toile : www.fr.alcove.com

Table des matières




Rappels réseau TCP/IP

Notion de paquet

- ✓ Les datagrammes IP sont transmis sous forme de *paquets*
  - Une adresse IP *source*
  - Une adresse IP *destination*
- ✓ Plusieurs protocoles sont supportés, principalement :
  - ICMP
  - UDP
  - TCP

Adresses IP et réseau

- ✓ Une adresse IP (v4) est de la forme A.B.C.D
  - Quadruplet d'entiers sur 8 bits (0–255)
  - Identifie une machine unique
- ✓ Décrit le réseau et distingue une machine dans un réseau
  - Partie réseau/partie machine
  - Le plus souvent réseau au début, machine ensuite
  - Notion de masque, ex :
    10.16.123.21/255.255.0.0 signifie
    - Interface d'IP 10.16.123.21
    - Appartient au réseau 10.16.*.* (classe B)
  - Notation abrégée : 10.16.0.0/16

Adresses IP spéciales

- Partie machine à 0 (ex : 10.16.0.0) : adresse de réseau
- Partie machine à 1 (ex : 10.16.255.255) : adresse de diffusion

Routage

✓ Adresses de réseau et masque permettent le routage sur le réseau local

✓ Les autres adresses sont aiguillées sur des passerelles
- Route explicite pour un réseau (ex : pour 10.15.0.0/16 passer par 10.16.123.1)
- Route par défaut

✓ Un firewall IP est avant toute chose une passerelle

Protocoles

- ✓ ICMP : informations de connectivité (ex : ping)
- ✓ UDP : Transmission de données non fiable
- ✓ TCP : Transmission de données fiable (mode connecté)

Ports

- ✓ Les paquets UDP et TCP impliquent la notion de *port*
- ✓ Entier entre 1 et 65535 (16 bits)
- ✓ Chaque paquet a un port source et un port destination
- ✓ Permettent de différencier les connexions

Exemple d'applications

Quelques protocoles applicatifs avec les ports de destination :

- ✓ UDP : tftp (port 69), DNS (en partie, port 53)
- ✓ TCP : ftp (ports 20 et 21), http (port 80)

Points à retenir

- ✓ Un paquet IP se caractérise par :
  - ✓ Un protocole (ICPM, TCP, UDP, ...)
  - ✓ Un type dans le protocole (ICMP) ou pour UDP et TCP :
    - – Un numéro de port source
    - – Un numéro de port destination
  - ✓ Une adresse IP source
  - ✓ Une adresse IP destination
- ✓ On peut connaître de plus :
  - – L'interface source
  - – L'interface de destination (après décision de routage)

Bases de Netfilter

Qu'est-ce que Netfilter ?

- ✓ Composant du noyau Linux (2.4.x)
  - – Intervient au plus tôt
  - – Performant
  - – Sûr
- ✓ Peut nécessiter une reconfiguration/recompilation
- ✓ Paramètrable par l'outil *iptables*

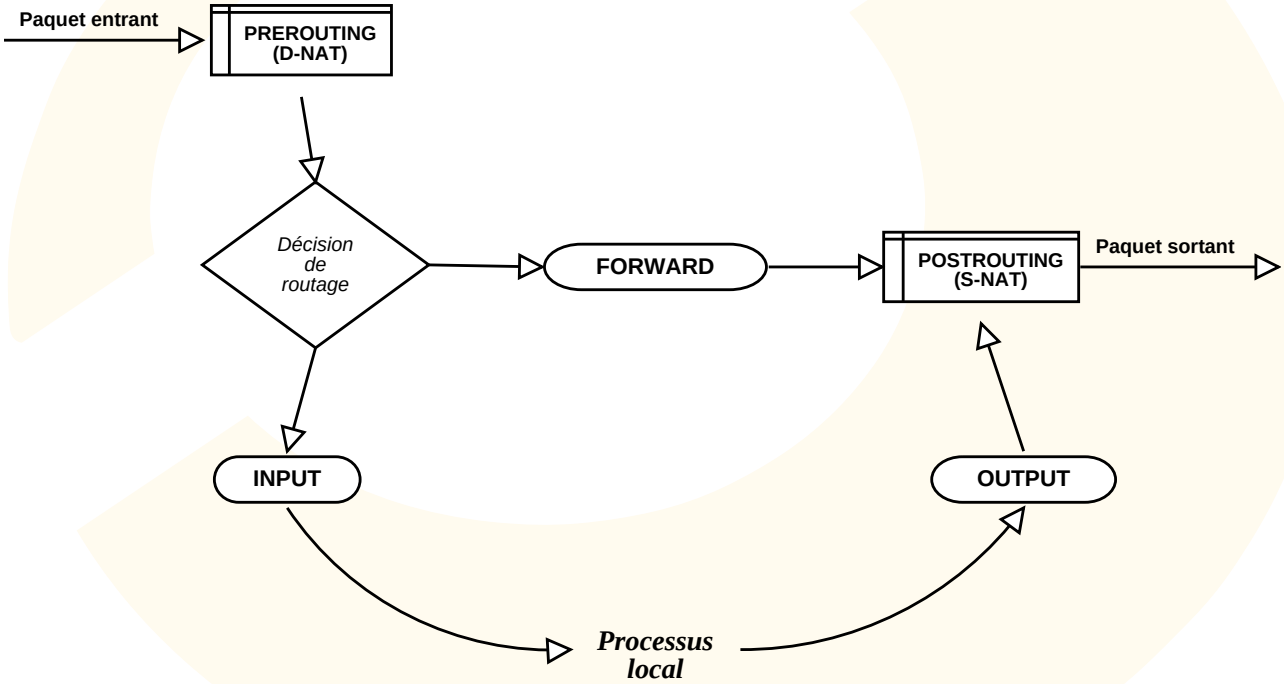



Retour
Fermer

Principe de Netfilter

- ✓ Notion de règles et de cibles
  - Règle : quels paquets ?
  - Cible : que faire ?
- ✓ Les règles sont dans des *chaînes*
  - Chaînes prédéfinies (point d'entrée dans la pile IP)
  - Chaînes définies par l'utilisateur (structuration)
- ✓ Pour le filtrage la première règle applicable est terminale
- ✓ Si aucune règle ne s'applique : politique par défaut

Le trajet d'un paquet



Chaînes prédéfinies

- ✓ PREROUTING (avant la décision de routage)
  - Permet de modifier l'adresse destination
- ✓ POSTROUTING (en fin de parcours)
  - Permet de modifier l'adresse source (ex : *masquerading*)
- ✓ INPUT : destination locale
- ✓ OUTPUT : source locale
- ✓ FORWARD : vient d'ailleurs, va ailleurs
  - La machine est configurée comme passerelle
  - Chaîne la plus importante pour un firewall

Exemples élémentaires

- ✓ `iptables -L` : montrer les chaînes de filtrage
- ✓ Politique par défaut :
  `iptables -P <CHAÎNE> <CIBLE>`
  ex : `iptables -P INPUT DROP` (interdit toute entrée)
  (essayez `ping localhost` !)
- ✓ Ajouter une règle :
  `iptables -A <CHAÎNE> <MATCH> <CIBLE>`
  ex : `iptables -A INPUT -p icmp -j ACCEPT` (autorise uniquement icmp)
  (réessayez `ping localhost`)
- ✓ Revenez à la normale :
  `iptables -P INPUT ACCEPT ; iptables -F`

Un exemple plus complexe

- ✓ Interdire les entrées par défaut :
```
iptables -P INPUT DROP
```
- ✓ Autoriser les réponses à des requêtes :
```
iptables -A INPUT -m state
--state ESTABLISHED,RELATED -j ACCEPT
```
- ✓ Autoriser tout ce qui vient de l'intérieur :
```
iptables -A INPUT -s 127.0.0.0/8 -j ACCEPT
```
- ✓ Garder une trace du reste :
```
iptables -A INPUT -j LOG
--log-prefix="DROP OUTPOUT :"
```
- ✓ Essayez de "pinguer" localhost, l'IP d'une interface, l'extérieur, à partir de l'extérieur...
- ✓ Le DNS fonctionne-t-il ? Pourquoi ?
- ✓ Quel est l'intérêt d'une configuration de ce type ?

Détails sur les états et les cibles

- ✓ Les cibles :
  - ACCEPT : autorise le passage du paquet
  - DROP : supprime le paquet comme s'il n'avait jamais été reçu
  - REJECT : supprime le paquet et génère une réponse ICMP vers la source pour lui dire que la destination n'est pas accessible
  - LOG : insère une ligne dans le journal noyau (règle non terminale)
- ✓ Filtrage par état :
  - NEW : nouveau paquet pour une connexion
  - ESTABLISHED : partie d'une connexion déjà existante
  - RELATED : nouveau mais lié à une connexion existante (exemple : ftp-data, erreur ICMP)

Principaux critères de correspondance

Les critères de sélection des paquets comprennent :

– Les entêtes IP
  ✓ IP source (machine ou réseau) : `-s`
  ✓ IP destination (machine ou réseau) : `-d`
  ✓ Type de protocole (ICMP, TCP, UDP) : `-p`
  ✓ Type dans le protocole, pour ICMP : `--icmp-type`
  ✓ Numéro(s) de port, pour TCP et UDP (source ou destination) : `--dport et --sport`

– La provenance physique
  ✓ L'interface : d'entrée ou de sortie : `-i` et `-o`
    ☞ `-i` n'a pas de sens dans OUTPUT et POSTROUTING
    ☞ `-o` n'a pas de sens dans INPUT et PREROUTING

Syntaxe des règles de correspondance

- ✓ Valeur exacte d'un critère :
  `-{i|o} <if>`
  `-{d|s} <ip>[/masque]`
  `--{dport|sport} <port>`
  `--icmp-type <type>`
- ✓ Toute valeur sauf une : `! <val>`
- ✓ Intervalle : `port1:port2`

Créer d'autres chaînes

En plus des chaînes prédéfinies (points d'entrée dans la pile IP), on peut créer ses propres chaînes.

- ✓ Une règle peut donner comme cible le nom d'une chaîne
- ✓ Équivalent d'un *goto* ou *jump* en programmation
- ✓ Permet de structurer les règles

Commandes de création/suppression de chaînes :

- ✓ Créer une chaîne : `iptables -N <CHAÎNE>`
- ✓ Supprimer une chaîne : `iptables -X <CHAÎNE>` (doit être vide)

Quelques autres options utiles

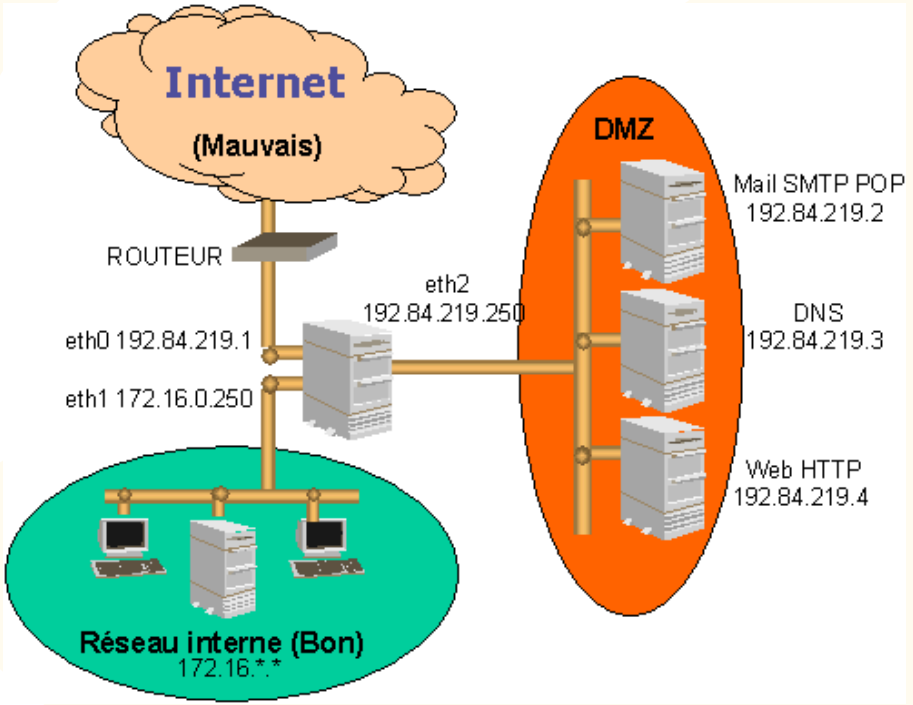
- ✓ `iptables -L -v -n`
  Montre les chaînes et les règles, avec statistiques (-v) et sans résolution de nom (-n)
- ✓ `iptables -Z`
  Remet les statistiques à zéro
- ✓ `iptables -D <CHAÎNE> <DESCRIPTION>`
  ou
  `iptables -D <CHAÎNE> N`
  Efface la règle décrite, ou la N^{e} règle
- ✓ `iptable -F <CHAÎNE>`
  Supprime toutes les règles d'une chaîne

Un exemple pratique

- ✓ Un routeur/firewall connecte trois réseaux :
  - Un réseau privé
  - Une “zone démilitarisée” (DMZ) incluant les serveurs publics
  - La connection vers un routeur Internet (RNIS, PPP, ADSL, Cable, ...)
- ✓ Nous allons configurer ce firewall ”à trois pattes”




L'architecture illustrée



Les protections spécifiques du noyau

- ✓ Des paramètres réseau peuvent être fixé par `/proc`
- ✓ On peut passer par *sysctl* (cf. `/etc/sysctl.conf`)
- ✓ Certains sont indispensables :

  `echo 1 > /proc/sys/net/ipv4/ip_forward`

  ☞ l'acheminement IP (routeur !)

  `echo 1 > /proc/sys/net/ipv4/ip_dynaddr`

  ☞ IP dynamique (PPP, DHCP)
- ✓ D'autres améliorent la sécurité

Ignorer les redirections ICMP

- ✓ `echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects`
- ✓ `echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects`
- ✓ `echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects`
- ✓ `echo 0 > /proc/sys/net/ipv6/conf/all/send_redirects`

Ignorer les diffusions ICMP

- ✓ `echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`

Protection anti *spoofing* et traçage des *martiens* (paquets dont la source n'est pas routable)

- ✓ `echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter`
- ✓ `echo 1 > /proc/sys/net/ipv4/conf/all/log_martians`

Ignorer le routage par la source

- ✓ `echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route`
- ✓ `echo 0 > /proc/sys/net/ipv4/conf/all/forwarding`
- ✓ `echo 0 > /proc/sys/net/ipv4/conf/all/mc_forwarding`

Résistance au *SYN flood*

- ✓ `echo 1280 > /proc/sys/net/ipv4/tcp_max_syn_backlog`
- ✓ `echo 1 > /proc/sys/net/ipv4/tcp_syn_cookies`

Configuration générale du firewall

- ✓ Installer le moins de choses possibles sur le firewall
- ✓ Pas de services réseau (web, ftp, etc.)
- ✓ Sauf éventuellement ssh (*Secure Shell*) accessible de l'intérieur

Variables de réseaux, d'interfaces et de machines

- ✓ Script lancé au démarrage (`/etc/init.d/`)
- ✓ Conventions de nommage : DMZ, GOOD, BAD pour DMZ, privé, Internet
- ✓ Utilisation de variables pour la lisibilité

```
DMZ_NET=192.84.219.0/24
GOOD_NET=172.16.0.0/16
IF_BAD=192.84.218.1
IF_DMZ=192.84.219.250
IF_GOOD=172.16.0.250

BAD_INT=eth0
DMZ_INT=eth2
GOOD_INT=eth1

DMZ_SMTP=192.84.219.2
DMZ_DNS=192.84.219.3
DMZ_WWW=192.84.219.4
```




Les chaînes créées

- ✓ Nommée sur le modèle *source-destination* :

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

- ✓ Chaîne spéciale pour les paquets d'erreur ICMP :

```
iptables -N icmp-acc
```

Règles d'autorisation des connexions déjà établies

- ✓ Pour TCP en général :

```
iptables -A INPUT -p TCP -m state --state ESTABLI-
SHED -j ACCEPT
iptables -A FORWARD -p TCP -m state --state ESTA-
BLISHED -j ACCEPT
```

- ✓ Pour les protocoles spéciaux et le masquage :

```
iptables -A FORWARD -p TCP -m state --state RELA-
TED -j ACCEPT
```

  - ☞ Pour ftp penser à insérer le module `ip_nat_ftp`

Les règles de saut

- ✓ On insère ensuite les règles qui dirigent les paquet vers les chaînes que nous avons créées :

```
# iptables -A FORWARD -s $GOOD_NET -d $DMZ_NET
-i $GOOD_INT -o $DMZ_INT -j good-dmz
# iptables -A FORWARD -s $GOOD_NET -i $GOOD_INT
-o $BAD_INT -j good-bad
# iptables -A FORWARD -s $DMZ_NET -i $DMZ_INT
-o $BAD_INT -j dmz-bad
# iptables -A FORWARD -s $DMZ_NET -d $GOOD_NET
-i $DMZ_INT -o $GOOD_INT -j dmz-good
# iptables -A FORWARD -d $DMZ_NET -i $BAD_INT
-o $DMZ_INT -j bad-dmz
# iptables -A FORWARD -d $GOOD_NET -i $BAD_INT
-o $GOOD_INT -j bad-good
```

✓ On garde une trace et on rejette le reste :
(équivalent à une politique par défaut de refus)

```
iptables -A FORWARD -j LOG
iptables -A FORWARD -j DROP
```

La chaîne des paquets ICMP

- ✓ On insère les règles d'acceptation de certaines erreurs ICMP

```
# 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
parameter-problem -j ACCEPT
```

- ✓ Cette chaîne sera appelée à partir des chaînes *source-destination*

Du réseau privé à la DMZ

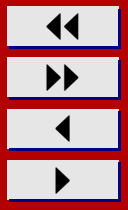
- ✓ Connexions issues de notre réseau privé vers les serveurs (WWW, DNS, ...) de notre DMZ
- ✓ On peut nommer les port par nom de service (cf. `/etc/services`)
- ✓ exemple, autoriser SMTP :

```
# iptables -A good-dmz -p tcp -d $DMZ_SMTP
-m state --state NEW --dport smtp -j ACCEPT
```

- ✓ De même pour WWW, FTP, etc...
- ✓ Attention aux services plus complexes, comme le DNS :

```
# iptables -A good-dmz -p udp -d $DMZ_DNS
--dport domain -j ACCEPT
# iptables -A good-dmz -p tcp -d $DMZ_DNS
-m state --state NEW --dport domain -j ACCEPT
```

☞ Utilise UDP et TCP !



De l'extérieur vers la DMZ

- ✓ Nous avons déjà une règle qui permet au trafic de réponse vers l'extérieur de passer
- ✓ Pour le courrier électronique :
  - Accepter SMTP venant de l'extérieur
  - Éventuellement accepter POP3 ou IMAP de l'extérieur
- ✓ Pour le serveur de nom :
  - Accepter les requètes DNS venant de l'extérieur
- ✓ Pour le serveur WWW :
  - Accepter les requètes venant de l'extérieur

Ce qui se traduit par les règles suivantes (à compléter) :

✓ SMTP :

```
# iptables -A bad-dmz -p tcp -d $DMZ_SMTP
-m state --state NEW --dport smtp -j ACCEPT
```

✓ DNS (Attention : UDP et TCP !)

...

✓ WWW

...

✓ Règles finales :

```
# iptables -A bad-dmz -p icmp -j icmp-acc
# iptables -A bad-dmz -j LOG
# iptables -A bad-dmz -j DROP
```

Comment dire non ?

- ✓ DROP détruit le paquet sans autre forme de procès
- ✓ REJECT envoie une notification (ICMP) à la source
- ✓ REJECT est de rigueur si la source est de confiance (réseau privé)
- ✓ DROP découragera un assaillant (mais pourra indiquer la présence d'un filtre)
- ✓ DROP ne découragera pas un programme tentant des requètes par erreur !

Du réseau privé vers l'Internet

- ✓ Nous souhaitons autoriser le Web, FTP et PING vers l'extérieur
- ✓ Nous pourrions mettre en place un mandataire (*proxy*) tel SQUID (une seule machine à autoriser) pour FTP et le Web
- ✓ Il faut réécrire les adresses IP source au passage (les IPs de la zone privée ne sont pas routables sur l'Internet)

Principe du NAT

- ✓ La traduction d'adresses réseau permet à un routeur de modifier les IPs à la volée
- ✓ On peut modifier la source (SNAT) ou la destination (DNAT)
  - Le *masquerading* est une forme de SNAT
  - Le *port forwarding* est une forme de DNAT
  - Le SNAT a lieu dans POSTROUTING
  - Le DNAT dans PREROUTING
- ✓ Il faut garder la mémoire des connexions en cours pour traduire correctement les paquets dans l'autre sens
  ☞ visible dans `/proc/net/ip_conntrack`

✓ La règle de réécriture :

```
# iptables -t nat -A POSTROUTING -o $BAD_INT -s $GOOD_NET
-j SNAT --to $IF_BAD
```

☞ Dans le cas d'une IP dynamique utilisez MASQUERADE au lieu de SNAT et ne précisez pas l'IP

✓ Les règles de filtrages :

```
# iptables -A good-bad -p tcp --dport www -m state
--state NEW -j ACCEPT
# iptables -A good-bad -p tcp --dport ftp -m state
--state NEW -j ACCEPT
# iptables -A good-bad -p icmp --icmp-type ping -j ACCEPT
# iptables -A good-bad -j LOG
# iptables -A good-bad -j REJECT
```

✓ Les règles pour le reste :

```
# iptables -A good-bad -j LOG
# iptables -A good-bad -j REJECT
```

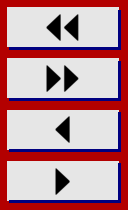
De la DMZ vers le réseau privé

- ✓ La plupart des protocoles TCP sont utilisés avec le client dans le réseau privé et le serveur dans la DMZ (mail, Web, ...)
  ☞ L'autorisation des connexions établies dans la chaîne FORWARD les prend déjà en compte
- ✓ Reste la partie UDP du DNS et ICMP :

```
# iptables -A dmz-good -p udp -s $DMZ_DNS
--sport domain -j ACCEPT
# iptables -A dmz-good -p icmp -j icmp-acc
```

- ✓ Et les règles finales :

```
# iptables -A dmz-good -j LOG
# iptables -A dmz-good -j REJECT
```



De l'Internet vers le réseau privé

- ✓ À part les retours de connexions déjà établies (déjà autorisés) ne rien laisser passer :

```
# iptables -A bad-good -j LOG
# iptables -A bad-good -j DROP
```

- ✓ Nous avons paramètré l'ensemble des connexions inter-réseaux

Protection du firewall lui-même

- ✓ La chaîne INPUT nous permet de protéger le firewall lui-même
- ✓ Nous créons trois chaînes : une par interface :

```
# iptables -N bad-if
# iptables -N dmz-if
# iptables -N good-if
```

- ✓ Nous insérons les trois règles de saut correspondantes :

```
# iptables -A INPUT -d $IF_BAD -j bad-if
# iptables -A INPUT -d $IF_DMZ -j dmz-if
# iptables -A INPUT -d $IF_GOOD -j good-if
```

L'interface vers l'extérieur

- ✓ Protection contre l'*IP spoofing* :

```
# iptables -A bad-if -i ! $BAD_INT -j LOG
# iptables -A bad-if -i ! $BAD_INT -j DROP
```

  ☞ L'IP correspond à l'extérieur mais vient d'une autre interface !

- ✓ Accepter les réponses aux ping que nous pourrions faire :

```
# iptables -A bad-if -p icmp --icmp-type pong -j ACCEPT
```

- ✓ Règles habituelles pour ICMP, sinon rejet :

```
# iptables -A bad-if -p icmp -j icmp-acc
# iptables -A bad-if -j LOG
# iptables -A bad-if -j DROP
```

L'interface vers la DMZ

- ✓ Protection contre l'*IP spoofing* (à compléter) :
  ...

- ✓ Laisser passer les réponses DNS en UDP provenant du serveur DNS :

```
# iptables -A dmz-if -p udp -s $DMZ_DNS domain -j ACCEPT
```

- ✓ Accepter les réponses aux pings, politique ICMP habituelle, et noter et interdire le reste (à compléter) :
  ...

L'interface vers le réseau privé

✓ Protection contre l'*IP spoofing* (à compléter) :

...

✓ Accepter les pings et les **réponses** aux pings :

...

✓ Noter et interdire le reste :

...

Règles pour les paquets émis du firewall

- ✓ Autoriser ping vers tout réseau

  ...

- ✓ Autoriser les requètes DNS vers tous les réseaux sauf privé (TCP et UDP)

  ...

Références et liens

Notre firewall est maintenant opérationnel. Voici des références documentaires sur le sujet :

- ✓ La page de manuel d'iptables : `man iptables`!
- ✓ Le site principal de *netfilter* : http://netfilter.samba.org/
- ✓ Des ressources sur *netfilter* : http://www.linuxguruz.org/iptables/
- ✓ La page sécurité de *Linux Weekly News*, cf. http://lwn.net/

La détection d'intrusion

Des outils permettent d'être averti en cas d'intrusion potentielle.

✓ Par analyse en temps réel du trafic réseau entrant, comme *snort*

✓ Par détection de changements suspects dans le système de fichier, comme *tripwire*

Snort

- ✓ Détecte les scans de ports TCP et UDP
- ✓ Règles pour détecter des tentatives d'intrusion :
  Par exemple : tentative de connexion FTP en tant que *root* :

```
alert tcp any any -> 192.168.1.0/24 21 \
(content : "USER root"; nocase; \
msg : "FTP root user access attempt";)
```

- ✓ Snort est fourni avec des règles de surveillance pour beaucoup de services réseau SMTP, Telnet, Netbios, DNS, Web, X11, RPC, etc.
- ✓ L'outil *snort-stat* génère un rapport envoyé par mail à l'administrateur (une fois par jour typiquement).
- ✓ http://www.snort.org/

Tripwire

Tripwire analyse le système de fichier pour détecter une éventuelle intrusion réussie.

- ✓ Détecte les changements suspects dans les droits d'accès (en particulier le bit *suid*)
- ✓ Détecte les modifications douteuses de contenus de fichiers
- ✓ À lancer périodiquement en période de faible charge (gros consommateur de ressource)
- ✓ Version libre : http://www.tripwire.org/, version commerciale : http://www.tripwire.com/