iptables: un pare-feu pour une passerelle

- Objet : installer un pare-feu pour une passerelle
- Niveau requis :
  avisé
- Commentaires : Fignoler le routage, installer un pare-feu sur une passerelle debian.
- Suivi :
  à-tester
  - Création par Hypathie 14/10/2014
  - Testé par <...> le <...>
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Nota : Au programme, description avancée de la table NAT (iptables), et des protocoles ICMP et TCP (ssh, FTP … logés) ; pour la mise en place d'un pare-feu iptables sur un système debian faisant office de routeur, muni de deux cartes ethernet.

Nota : Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Pour ce qui concerne iptables, ce wiki prend la suite du wiki iptables-pare-feu-pour-un-client, où les connaissances de base sur les commandes iptables ont été abordées.

Pour ce qui concerne les exemplifications qui vont suivre, je considérerai qu'une passerelle a été mise en place et que les commandes de base d'iptables sont connues.

Si ce n'est pas le cas, suivre les liens internes du “petit rappel” ci-dessous est un pré-requis indispensable à ce wiki.

Petit rappel de la situation

- La configuration physique :

Nous sommes derrière un box-machin (avec routeur et serveur DHCP activé).
Le système **debian-routeur** (passerelle) a deux cartes ethernet

1. **eth0** (IP: **192.168.0.1**)
   est relié à la box-machin (par un câble droit) ;
2. un **ordinateur A** est relié lui aussi à la box-machin son IP est **192.168.0.22**
3. **eth1** (IP: **192.168.1.1**) est reliée à un ordinateur B[2)];
4. L'IP de **l'ordinateur B** est **192.168.1.2**

L'ordinateur A et le système debian-routeur appartiennent au sous-réseau 192.168.0.0/255.255.255.0

L'ordinateur B appartient au sous sous-réseau : 192.168.1.0/255.255.255.0

* Les services installés :

Sur l'ordinateur A (IP: 192.168.0.22), on a installé ce pare-feu.

Sur debian-routeur(IP: 192.168.0.1), on a installé:
- aucune interface graphique lors de son installation : on le configure en passant par ssh depuis l'ordinateur A
- un serveur DHCP basique qui fournit l'IP 192.168.1.2 a l'ordinateur B
- Un serveur DNS (bind9)
- le fichier **/etc/hosts de l'ordi B** est complété[3], par exemple:

```
127.0.0.1 localhost
127.0.1.1 debian-hpmondomaine.hyp  debian-hp
```

- le fichier **/etc/resolv.conf de l'ordi B** a connaissance de l'ip du serveur DNS :

```
domaine mondomaine.hyp
search mondomaine.hyp
nameserver 192.168.1.1
```

- De même le fichier **/etc/hosts de la passerelle** connaît l'ordi B : Par exemple `192.168.1.3 debian-hp` - La table NAT a été modifiée très simplement pour l'instant afin de le transformer en passerelle avec masquerade sur eth0 (interface web).
- le fichier /etc/sysctl.conf a été modifié : la ligne `net.ipv4.ip_forward=0` est devenue `net.ipv4.ip_forward=1`.

La table de routage

Communication entre le réseau A et B

Échec des ping entre A et B

Les pings, depuis ordinateur A vers 192.168.0.1 (eth0 de debian-routeur) et depuis ordinateur B vers 192.168.1.1 (eth1 de debian-routeur), fonctionnent.

De même les pings :

- Depuis debian-routeur vers l'ordinateur B:

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=64 time=0.132 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=64 time=0.123 ms
^C
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
```

```
rtt min/avg/max/mdev = 0.123/0.127/0.132/0.012 ms
```

- Depuis debian-routeur vers l'ordinateur A

```
ping 192.168.0.22
```

retour de la commande

```
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
^C
--- 192.168.0.22 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1007ms
```

Mais cela est loin d'être suffisant !

- Depuis l'ordinateur B (192.168.1.0/24) vers l'ordinateur A:

```
ping 192.168.0.22
```

retour de la commande

```
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
^C
--- 192.168.0.22 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 4999ms
```

- Depuis l'ordinateur A (192.168.0.0/24) vers l'ordinateur B:

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms
```

C'est que la configuration de la passerelle n'est pas suffisante pour que l'acheminement (route) des paquets puisse se faire dans les deux les sens.

La table de routage de la passerelle

La table de routage d'un routeur comporte les adresses des réseaux de destination, le masque, les adresses des passerelles (routeurs intermédiaires) permettant de les atteindre, l'adresse de la carte réseau (interface) par laquelle le paquet doit sortir du routeur.

La commande **`route`** permet d'afficher et de manipuler le contenu de la table de routage.
L'option **`-n`** permet de lister la table actuelle.

- La table de routage actuelle de debian-routeur :

```
route -n
```

retour de la commande

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use
Iface
default         192.168.0.254   0.0.0.0         UG    0      0        0
eth0

192.168.0.0     *               255.255.255.0   U     0      0        0
eth0
192.168.1.0     *               255.255.255.0   U     0      0        0
eth1
```

L'étoile * dans la colonne `Passerelle` indique que le réseau de destination `192.168.0.0` (i.e.192.168.0.0/24) n'a pas de passerelle.

De même (*) le réseau de destination 192.168.1.0/24 n'a pas de passerelle.

De plus ce retour indique (lignes n°1) que la route est active (indicateur U) et utilise la passerelle (indicateur G). Mais qu'aucune passerelle n'est activée lignes N°2 et n°3 (pas de G).

Il faut dresser une table de routage permettant au deux réseaux de communiquer.

```
route add -net 192.168.0.0 netmask 255.255.255.0 gw 192.168.0.1 dev eth0
```

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.1.1 dev eth1
```

- La table est maintenant :

```
/sbin/route -n
```

ou en root 

```
route -n
```

retour de la commande

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use
Iface
0.0.0.0         192.168.0.254   0.0.0.0         UG    0      0        0
eth0
192.168.0.0     192.168.0.1     255.255.255.0   UG    0      0        0
eth0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0
eth0
192.168.1.0     192.168.1.1     255.255.255.0   UG    0      0        0
eth1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0
eth1
```

- Ping de l'ordinateur B (192.168.1.0/24) vers l'ordinateur A (192.168.0.0/24)

```
ping 192.168.0.22
```

retour de la commande

```
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_req=1 ttl=63 time=0.546 ms
64 bytes from 192.168.0.22: icmp_req=2 ttl=63 time=0.338 ms
64 bytes from 192.168.0.22: icmp_req=3 ttl=63 time=0.344 ms
^C
--- 192.168.0.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.338/0.409/0.546/0.098 ms
```

- Par contre le ping de l'ordinateur A (192.168.0.0/22) vers l'ordinateur B (192.168.1.0/22)

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms
```

Et oui, l'ordinateur A n'est pas derrière la passerelle, et même si sa table de routage est correctement configurée, elle ne peut lui indiquer la route vers le sous réseau 192.168.1.0/24 dont les adresses IP

sont, pour tout réseau extérieur au réseau B, y compris le réseau A, identiques à l'IP de la passerelle. C'est là la fonction même de l'utilisation de MASQUERADE mis en place avec iptables.

Sur l'ordinateur A, IP:192.168.0.22 :

```
route -n
```

retour de la commande

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use
Iface
0.0.0.0         192.168.0.254   0.0.0.0         UG    0      0        0
eth0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0
eth0
```

Il faut donc instruire la table de routage de l'ordinateur A de l'existence du réseau B et lui indiquer que la route à prendre est la passerelle !

```
route add -net 192.168.1.0 gw 192.168.0.1 netmask 255.255.255.0 dev eth0
```

```
route -n
```

retour de la commande

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use
Iface
0.0.0.0         192.168.0.254   0.0.0.0         UG    0      0        0
eth0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0
eth0
192.168.1.0     192.168.0.1     255.255.255.0   UG    0      0        0
eth0
```

- Et maintenant le ping de l'ordinateur A vers l'ordinateur B:

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=63 time=0.504 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=63 time=0.308 ms
^C
--- 192.168.1.2 ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.308/0.406/0.504/0.098 ms
```

Au prochain redémarrage, ces commandes de routage seront à relancer.
Pour se l'éviter, il faut un script init.d (update-rc.d)
Ne le faites pas sur la passerelle, à la limite sur l'ordinateur A.
Sur la passerelle, le suivi de connexion pour le protocole ICMP devrait être suffisant.
Et pour l'ordinateur A, grâce à la passerelle et à la maîtrise du DNAT/SNAT, on pourrait faire en sorte que l'ordinateur reçoive un ping de la passerelle quand elle reçoit un ping à destination du réseau B.

La table NAT

NAT = transcription d'adresse (Network Address Translation ou NAT en anglais).

La traduction d'adresse est gérée avec les chaînes PREROUTING, POSTROUTING et OUTPUT.

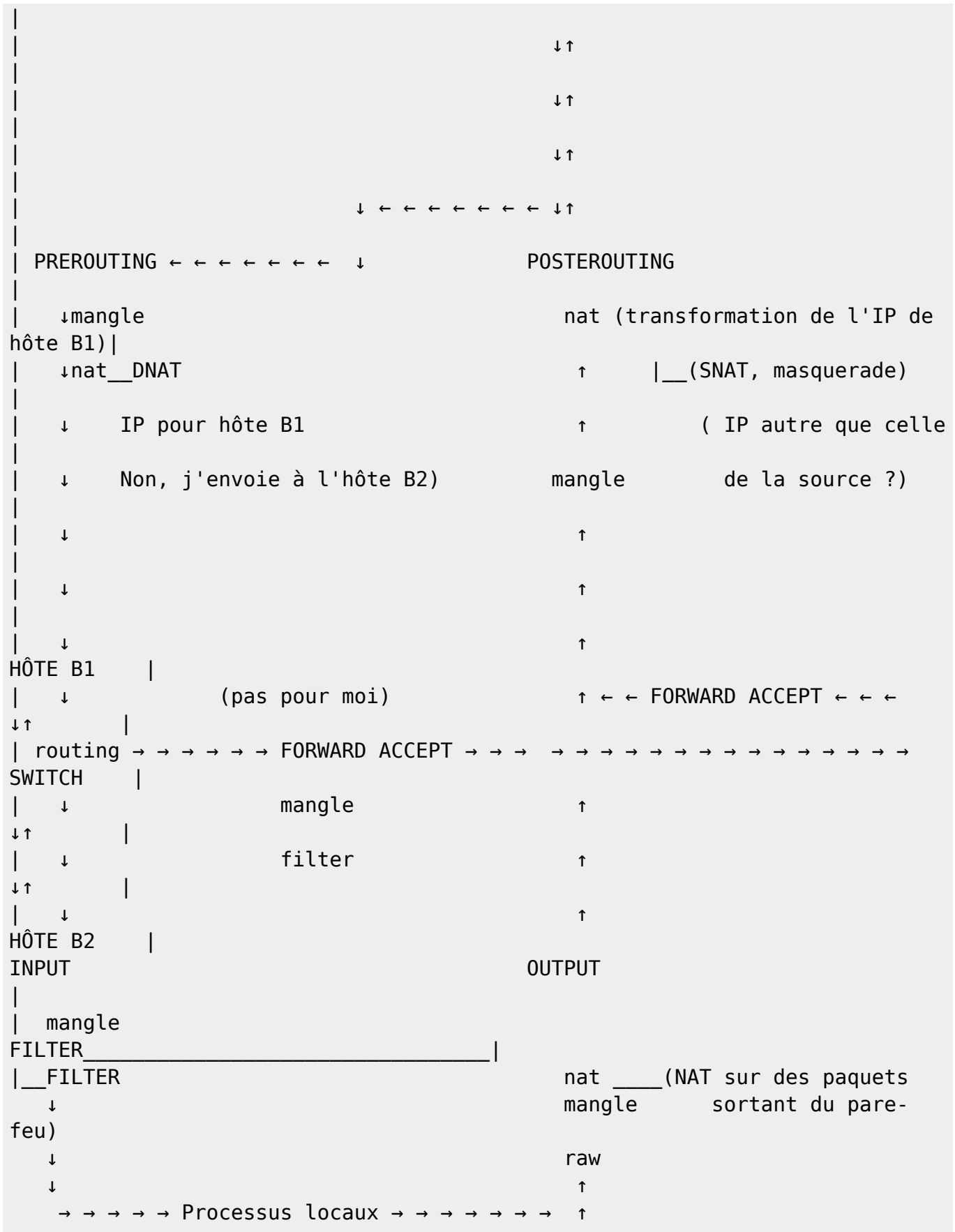
Dans la chaîne PREROUTING (avant routage), on peut modifier l'adresse de destination des paquets qui arrivent sur notre passerelle mais qui doivent aller sur le réseau B. S'ils ne lui sont pas destiné, et qu'au niveau de la passerelle on détermine une redirection pour l'hôte de notre choix, ce serait là faire du DNAT (du “NAT destination”).

Dans la chaîne POSTROUTING, (après routage), on peut modifier l'adresse source ; celle des hôtes du réseau B, ou d'une seul hôte du réseau B, la passerelle pourrait aussi d'indiquer au réseau A une autre adresse pour l'ordinateur que celle qui est véritablement la sienne. Ce serait là faire du SNAT (NAT source).

Dans la chaîne POSTROUTING, “l'IP Masquerade”.
Le masquage d'adresse IP est une forme de traduction d'adresse réseau (NAT), et ressemble au SNAT. Cela permet aux ordinateurs d'un réseau privé n'ayant aucune adresse IP d'accéder à Internet via l'adresser IP unique d'une machine Linux.
Voir : http://www.inetdoc.net/guides/iptables-tutorial/nattable.html

Un petit schéma

```
                      INTERNET
                         ↓↑
                        modem
                         ↓↑
                         ↓↑
entrée                ← ↓↑ ← ←  ←  ←  ←  ←  ←  ← ←        sortie
INPUT = ACCEPT ?_____↓ eth0___PASSERELLE___eth1   ↑_______OUTPUT = ACCEPT ?
_________
|                     ↓ → → → → →  → → → → → ↓↑ → ↑
```



Le shéma ci-dessus décrit ce que fait le pare-feu de la passerelle par laquelle, les paquets reçus et envoyés par l'hôte de réseau B, passent pour être routés, transformés, et filtrés.

Pour ce qui concerne masquerade et snat, ce qui se passe pour un paquet est représenté par la partie

du schéma au dessus de la ligne transversale FORWARD.
Ce schéma tente de mettre en valeur deux pré-requis ;

- 1) la chaîne FORWARD laisse passer dans les deux sens le flux qu'on veut autoriser;
- 2) les chaînes INPUT et OUTPUT laisse entrer et sortir ces mêmes ces paquets, non seulement par **eth0** mais aussi par **eth1**.

En d'autres termes, la table **FILTER** et les chaînes FORWARD, INPUT et OUTPUT conditionnent le POSTEROUTING (et le PREROUTING).
Ré-préquis qui déterminent le fait que l'ordinateur B puisse recevoir des paquets, et en envoyer.

Ainsi, si La commande installée précédemment

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

permettait la transmission des paquets du réseau interne (eth1), au externe (eth0), et vice-versa c'est parce qu'on avait laissé la police par défaut pour la tables FILTER du pare-feu de la passerelle.

- Nous obtenons pour l'instant :

```
iptables -L -t nat && iptables -L
```

retour de la commande

```
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
MASQUERADE  all  --  anywhere    anywhere
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
```

Dépendance entre NAT et FILTER (FORWARD)

Nous commencerons par la mise au point des règles de la chaîne FORWARD, nous verrons en dernier les chaînes INPUT et OUTPUT afin de pouvoir bien distinguer certaines interdépendances, mais aussi

pour pouvoir comprendre les commandes pour le pare-feu et les protocoles qu'elles cibleront étape par étape.

DROP sur la chaîne FORWARD de la table FILTER

Comme vous vous en souvenez, la “philosophie” d'un pare-feu et de tout interdire par défaut, puis d'autoriser le passage de ce dont nous avons besoin.

- **Allez faisons la preuve de cette dépendance entre POSTROUTING (NAT) et FORWARD (FILTER)**

```
iptables -F
```

```
iptables -X
```

```
iptables -t nat -F
```

```
iptables -t nat -X
```

```
iptables -P FORWARD DROP
```

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

- Et maintenant l'ordinateur B ne plus accéder à internet :

-Ping de l'ordinateur A vers B :

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2017ms
```

-Ping de l'ordinateur A vers l'IP de google.fr :

```
ping 64.233.166.94
```

retour de la commande

```
PING 64.233.166.94 (64.233.166.94) 56(84) bytes of data.
^C
--- 64.233.166.94 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2014m
```

De même on peut tester : depuis l'ordinateur B au moyen du navigateur pour constater qu'il n'y a plus de navigation.

Les commandes pour tout annuler et revenir à l'état précédent :

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

INPUT et OUTPUT, pour ne pas avoir à répéter ce rappel, plus bas, quand nous mettrons en place les règles des chaînes INPUT et OUTPUT de la table FILTER.

Et pour redonner accès à internet à l'ordinateur B.

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

Les règles du pare-feu pas à pas

Maîtriser la chaîne FORWARD

Donc dans notre pare-feu, il faudra :

1. Interdire par défaut (DROP) la chaîne FORWARD
2. Filter ce qui passe par FORWARD
3. Par exemple pour autoriser la connexion au web du LAN on autorise tcp
4. On autorise ICMP pour les pings
5. Et bien sûr, ne pas oublier si on a “flushé” la table NAT, la commande avec MASQUERADE

(*Puisque la politique par défaut de la table FILTER des chaînes INPUT et OUTPUT n'ont pas encore été modifiées, on peut déjà tester ces commandes.*)

- Comme ceci :

```
iptables -F
iptables -X
iptables -t nat -F
```

```
iptables -t nat -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

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

iptables -P FORWARD DROP

iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p\
 tcp -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p\
 tcp -m state --state ESTABLISHED -j ACCEPT
```

- Ce qui donne :

```
iptables -L
```

retour de la commande

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

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.0/24       anywhere             state
NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.1.0/24       state
ESTABLISHED

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

> La navigation web fonctionne
> Le ping vers google et vers 192.168.0.22 est bloqué car on a autorisé seulement tcp

- On ajoute une autorisation pour icmp :

```
iptables -t filter -A FORWARD -p icmp -j ACCEPT
```

- Ce qui donne :

```
iptables -L FORWARD
```

retour de la commande

```
Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.0/24       anywhere             state
NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.1.0/24       state
ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
```

- OU de manière plus précise :

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p icmp\
 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p icmp\
 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

- Ce qui donnerait cette fois :

```
iptables -L FORWARD
```

retour de la commande

```
Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.0/24       anywhere             state
NEW,RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.1.0/24       state
RELATED,ESTABLISHED
ACCEPT     icmp --  192.168.1.0/24       anywhere             state
NEW,RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             192.168.1.0/24       state
RELATED,ESTABLISHED
```

> Le web et le ping fonctionne :

- De la machine A vers la machine B :

```
ping 192.168.1.2
```

retour de la commande

```
<code>PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=63 time=0.402 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=63 time=0.315 ms
64 bytes from 192.168.1.2: icmp_req=3 ttl=63 time=0.306 ms
^C
```

```
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.306/0.341/0.402/0.043 ms
```

- De la machine B vers la machine A :

```
ping 192.168.0.22
```

retour de la commande

```
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_req=1 ttl=63 time=0.370 ms
64 bytes from 192.168.0.22: icmp_req=2 ttl=63 time=0.341 ms
64 bytes from 192.168.0.22: icmp_req=3 ttl=63 time=0.326 ms
^C
--- 192.168.0.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.326/0.345/0.370/0.028 ms
```

Quelques exlications :

- On autorise toute connexion pour le LAN

```
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -o eth1 -j ACCEPT
```

```
iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

> Ce qui entre `-i` et ce qui sort `-o`
> La navigation internet depuis un ordi du réseau 192.168.1.0/24 est possible
> ping de l'un de ces ordi vers google est ok

- Dans les deux sens d'une interface à l'autre :

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

```
iptables -A FORWARD -o eth1 -i eth0 -j ACCEPT
```

> On constate que sans préciser de protocole ni de réseau, cela revient au même que précédemment

```
iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

> On autorise le passage de eth1 à eth0 et de eth0 à eth1

- Avec suivi de connexion d'une interface à l'autre avec des règles différentes :

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24
-d 0.0.0.0/0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

> On autorise le passage de tous protocoles avec suivi de connexion, de eth1 à eth0, et inversement, en précisant les réseaux source et destination.

```
iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT       all   --   192.168.1.0/24         anywhere
state NEW,RELATED,ESTABLISHED
ACCEPT     all  --  anywhere             192.168.1.0/24
state RELATED,ESTABLISHED
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

> Là encore la navigation est possible pour un ordi du sous-réseau 192.168.1.0/24
> Et le ping fonctionne vers google et l'ordi 192.168.0.22 sans besoin d'ajouter de règle imcp car on a autorisé TOUS les protocoles

Enfin il est inutile d'ajouter une règle udp pour la résolution de nom si le serveur de nom est installé et correctement paramétré.
Dans le cas contraire on ajouterait :

```
iptables -t filter -A FORWARD -p udp --dport 53 -j ACCEPT
```

Ou pour être plus précis :

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24
-d 0.0.0.0/0 -p udp\
 --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p udp\
 --dport 53 -m state --state ESTABLISHED -j ACCEPT
```

Dépendance du NAT et des chaînes FILTER sur l'interface interne (eth1)

Rappelons que la mise en place du pare-feu nécessitera de poser une police par défaut à DROP pour les chaînes INPUT et OUTPUT de la table FILTER.

Si nous voulons protéger le réseau B (derrière la passerelle) nous voulons aussi qu'il puisse accéder à Internet.

Or, si l'on met la police de de la table FILTER à DROP sur notre passerelle cela vaudra autant pour l'interface connectée au web (eth0) que pour l'interface connectée au réseau interne (B).

Dans beaucoup de wiki cette question est contournée en laissant la police par défaut (ACCEPT) sur la chaîne OUTPUT (FILTER).

Pour suivre notre “philosophie” : on met tout à DROP, il faut alors crée une règle ouvrante pour l'interface interne (eth1) comme on le fait pour l'interface lo.

Mais cela ne suffit pas il faut pour chaque règle (ouverture du web, ouverture pour le DNS) le faire pour chacune des interfaces (eth0 mais aussi eth1).

On le voit bien sur le schema : les règles sont au niveau de chaque interface !

Si vous testez, ne lancez pas les commandes suivantes sur la passerelle au moyen d'un connexion ssh, vous vous enfermeriez dehors.

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

iptables -t nat -P PREROUTING ACCEPT
```

```
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P INPUT ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

#Maintenant que tout est à DROP il faut s'occuper de la boucle local
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Et notre interface interne :
iptables -A INPUT -i eth1 -j ACCEPT
iptables -A OUTPUT -o eth1 -j ACCEPT

##On garde nos règles concernant le DROP sur FORWARD (FILTER)
#mais on oublie pas eth1 !
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p\
 tcp -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p\
 tcp -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p icmp\
 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p icmp\
 -m state --state ESTABLISHED,RELATED -j ACCEPT

#décocher les deux règles ci-dessous si un serveur DNS n'est pas installé
sur la passerelle

#iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d
0.0.0.0/0 -p udp\
# --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT

#iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p udp\
# --dport 53 -m state --state ESTABLISHED -j ACCEPT

##Règles icmp pour INPUT et OUTPUT
iptables -t filter -A INPUT -p icmp -i eth0 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A OUTPUT -p icmp -o eth0 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A INPUT -p icmp -i eth1 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -t filter -A OUTPUT -p icmp -o eth1 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

# Et on prend soin de laisser entrer et sortir (INPUT, OUTPUT de FILTER)
# le flux nécessaire au DNS (53) et au web (80, 443..)
# et là encore on n'oublie pas eth1

iptables -t filter -A OUTPUT -o eth0 -p udp -m udp --dport 53\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth0 -p udp -m udp --sport 53\
 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -o eth1 -p udp -m udp --dport 53\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth1 -p udp -m udp --sport 53\
 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -o eth0 -p tcp -m multiport --dports\
 80,443,8000 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth0 -p tcp -m multiport --sports\
 80,443,8000 -m state --state ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth1 -p tcp -m multiport --dports 80,443,8000 -j
ACCEPT
iptables -A INPUT -i eth1  -p tcp -m multiport --sports 80,443,8000 -j
ACCEPT
```

Super ! Tout fonctionne 😃

Nous avons un pare-feu rudimentaire mais dont la politique par défaut pour FILTER est à DROP ; il permet de filtrer les protocoles, entrant et ressortant, essentiels à notre réseau B pour qu'il accède au web, et rejette les autres !

Mais ce n'est pas suffisant. Nous allons détailler davantage ce qu'on laisse entrer, et surtout nous allons envoyé dans les logs certains protocoles qu'on utilise.

Un pare-feu avancé pour le routeur-debian

Revisiter ICMP

Puisqu'on a abordé la question du ping, revenons sur cette règle

```
iptables -A FORWARD -p icmp -j ACCEPT
```

un peu trop permissive par rapport à la bonne vieille attaque doS (denial of service attack), ou "attaque par déni de service".

Cette chaîne va être utilisée pour bloquer seulement certains paquets ICMP provenant de l'internet (WAN).
Le **retour de type 8 (Echo Request)** ne sera pas accepté, car on ne veut pas répondre au ping que dans certaines conditions.

Les packets ICMP donne l'état d'une requête, par exemple d'une connexion refusée etc...
Ils proviennent de la couche 2 (Liaison de données) du modèle OSI et selon le standard, il ne devrait y avoir de paquets qui utilisent la fragmentation.
Lorsque cette situation se présente, nous sommes en présence d'un déni de service (DoS). Nous allons donc essayer de réduire les risques en rejetant les paquets fragmentés qui n'ont pas lieu d'être.

- Créons une chaîne utilisateur :

```
iptables -N icmp_packets
```

- On repère les paquets icmp fragmentés et on les interdit :

```
iptables -A icmp_packets --fragment -p ICMP -j LOG\
 --log-prefix "icmp_packets:=DROP "
iptables -A icmp_packets --fragment -p ICMP -j DROP
```

- On interdit certains ping (8):

```
iptables -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j DROP
```

Détail sur ICMP

- ICMP echo-reply (type 0)

Le balayage Ping est généralement utilisé pour déterminer les hôtes qui sont en place sur un réseau. Typiquement, cela se fait par l'envoi de paquets ICMP ECHO REQUEST à l'hôte cible.

- ICMP destination-inaccessible (type 3)

Le filtrage "fragmentation nécessaire" du trafic est une mauvaise idée.

codes :
0 réseau inaccessible
1 hôte inaccessible
2 protocole inaccessible
3 port inaccessible
4 fragmentation nécessaire
5 Échec de la route Source
6 réseau inconnu
7 hôte inconnu
8 Source Host isolé
9 réseau interdit

10 hôte interdit
11 TOS-réseau inaccessible
12 TOS-hôte inaccessible
13 communication interdites [RFC1812]
14 violation de la priorité de l'hôte [RFC1812]
15 priorité de coupure [RFC1812]

- ICMP Source Quench (type 4)

Ceci est détaillé dans la RFC 792 : le filtrage de ce type de trafic est généralement considéré comme une mauvaise idée.

- ICMP Redirect (type 5)

codes
0 Redirection du datagramme pour le réseau (ou sous-réseau)
1 Redirection du Datagramme de l'hôte
2 Redirection du datagramme pour le type de service et de réseau
3 Redirection du Datagramme pour le type de service et d'accueil

- Le type ICMP 6 : Alternate Host Address

Pour modifier l'adresse hôte.

- Type ICMP 7 : ceux non-assignés
- Type ICMP 9 “Router Advertisement” : voir RFC 1256
- Type ICMP 10 “Routeur ICMP sollicitation” : voir aussi RFC 1256
- Type ICMP 11 “ICMP Time Exceeded” : pour le temps dépassé

codes
0 Temps de vie dépassé pour le transit 1 Temps dépassé pour le remontage des fragments

- type 11 “ICMP Time Exceeded” (temps dépassé)

Codes 0 Temps de vie dépassé (pour le passage)
1 Temps dépassé pour le réassemblage des fragments

- Problème de paramètre ICMP (type 12)

codes
0 pointeur (indique une erreur)
1 pointeur pour l'absence d'une option obligatoire [de RFC1108]

2 pointeur mauvaise longueur

Horodatage ICMP (type 13)
Horodatage ICMP (Réponce) (type 14)
Requête ICMP (pour obtenir des information)(type 15)
Requête ICMP (pour obtenir des réponses) (type 16)
Requête ICMP (requête d'adresse masque) (type 17)
Requête ICMP (répondre d'adresse masque) (type 18)
ICMP réservés (types 19-29)
ICMP “Traceroute” (type 30)
Erreur de conversion ICMP (Datagramme) (type 31)

Bugs dans les réponses ICMP d'erreur
Parfois, les routeurs qui envoient des réponses non valides à des trames de diffusion. Il s'agit d'une violation de RFC1122, “Exigences pour les hôtes Internet - couches de communication”.
En conséquence, ces événements sont enregistrés par le noyau. Pour éviter de remplir votre fichier de log avec l'encombrement inutile, vous pouvez dire au noyau de ne pas émettre ces avertissements.
Fix Me!

Il y a beaucoup plus de types ICMP que cela mais on peut dire que tout ce qui n'est pas expressément autorisé ci-dessus doit être bloqué.

- Nous allons donc accepter tout ceux-ci et bloquer les autres dans la chaîne FORWARD [4]:

(Nous laissons passer aussi **provisoirement** une ouverture béante pour les chaînes INPUT et OUTP (FILTER) afin de tester tout cela.

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

iptables -P FORWARD DROP

#On remets "masquerade" car on a flushé:
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p tcp\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p tcp\
 -m state --state ESTABLISHED -j ACCEPT
```

```
#Si pas de DNS configurer on dé-commente les règles udp suivantes :

#iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d
0.0.0.0/0 -p udp\
# --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT

#iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p udp\
# --dport 53 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -p icmp -j ACCEPT

iptables -N icmp_packets
iptables -A icmp_packets --fragment -p ICMP -j LOG\
 --log-prefix "icmp_packets:DROP: "
iptables -A icmp_packets --fragment -p ICMP -j DROP
iptables -A icmp_packets -p ICMP --icmp-type 8 -j DROP
iptables -A icmp_packets -p ICMP --icmp-type 11 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 3/4 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 3/3 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 3/1 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 4 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 12 -j ACCEPT
iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.0.0/24 -p icmp\
 --icmp-type echo-request -j ACCEPT

# Pour le retour nous utilisons la dernière règle
iptables -A FORWARD -s 192.168.0.0/24 -d 192.168.1.0/24 -p icmp\
 --icmp-type echo-reply -j DROP
```

> La navigation web fonctionne pour l'ordi B
> Les ping de même

DROP sur INPUT et OUTPUT (chaîne filter)

Cette fois, il va falloir interdire (DROP) les chaînes INPUT et OUTPUT de la table FILTER excepté pour tous les protocoles dont on a besoin.
D'abord pour ICMP qu'on ne laisse plus massivement entrer et sortir (table FILTER à DROP pour INPUT et OUTPUT, en plus de FORWARD).

Attention l'ordre des règles pour IMCP sur INPUT, OUTPUT et FORWARD, compte ici.

On conserve ce qu'on a fait précédemment en ajoutant simplement les commandes du script pour un client du LAN mais en tenant compte du fait qu'il y a cette fois deux interfaces réseau.

```
iptables -F
iptables -X
```

```
iptables -t nat -F
iptables -t nat -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P INPUT ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

#Maintenant que tout est à DROP il faut s'occuper de la boucle local
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Et notre interface interne :
iptables -A INPUT -i eth1 -j ACCEPT
iptables -A OUTPUT -o eth1 -j ACCEPT

# On garde nos règles concernant FORWARD (FILTER)

iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
-p tcp\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d 192.168.1.0/24
-p tcp\
 -m state --state ESTABLISHED -j ACCEPT

#Si pas de DNS configurer on dé-commente les règles udp suivantes :

#iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d
0.0.0.0/0 -p udp\
# --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT

#iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p udp\
# --dport 53 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -p icmp -j ACCEPT

# règles icmp sur INPUT et OUTPUT
iptables -t filter -A INPUT -p icmp -i eth0 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A OUTPUT -p icmp -o eth0 -m conntrack\
```

```
 --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A INPUT -p icmp -i eth1 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -t filter -A OUTPUT -p icmp -o eth1 -m conntrack\
 --ctstate ESTABLISHED,RELATED -j ACCEPT

# Et on prend soin de laisser entrer et sortir (INPUT, OUTPUT de FILTER)
# le flux nécessaire au DNS (53) et au web (80, 443..)
# et là encore on n'oublie pas eth1

iptables -t filter -A OUTPUT -o eth0 -p udp -m udp --dport 53\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth0 -p udp -m udp --sport 53\
 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -o eth1 -p udp -m udp --dport 53\
 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth1 -p udp -m udp --sport 53\
 -m state --state ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -o eth0 -p tcp -m multiport --dports\
 80,443,8000 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth0 -p tcp -m multiport --sports\
 80,443,8000 -m state --state ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth1 -p tcp -m multiport --dports 80,443,8000 -j
ACCEPT
iptables -A INPUT -i eth1  -p tcp -m multiport --sports 80,443,8000 -j
ACCEPT


# Les règles ICMP pour OUTPUT et INPUT
#(en y intégrant celles de testées pour FORWARD)

iptables -A OUTPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 0 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 3/4 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 3/4 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 3/4 -j ACCEPT

iptables -A FORWARD -p icmp --icmp-type 3/3 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 3/3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3/3 -j ACCEPT
```

```
iptables -A FORWARD -p icmp --icmp-type 3/1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3/1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 3/1 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 4 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 4 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 4 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 8 -m limit --limit 2/s -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j LOG --log-prefix "ICMP/in/8
Excessive: "
iptables -A INPUT -p icmp --icmp-type 8 -j DROP
iptables -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 8 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 11 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 12 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 12 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type 12 -j ACCEPT

iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.0.0/24 -p icmp\
 --icmp-type echo-request -j ACCEPT

# Pour le retour nous utilisons la dernière règle
iptables -A FORWARD -s 192.168.0.0/24 -d 192.168.1.0/24 -p icmp\
 --icmp-type echo-reply -j DROP

iptables -A INPUT -p icmp -m limit -j LOG --log-prefix "ICMP/IN: "
iptables -A OUTPUT -p icmp -m limit -j LOG --log-prefix "ICMP/OUT: "
```

On est content, l'ordination B peut naviguer sur internet et les pings sont fonctionnels :

- De l'ordinateur A vers B[5)]

```
ping 192.168.1.2
```

retour de la commande

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=63 time=0.341 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=63 time=0.316 ms
^C
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.316/0.328/0.341/0.022 ms
```

- Et de l'ordinateur B vers l'ordinateur A :

```
ping 192.168.0.22
```

retour de la commande

```
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_req=1 ttl=63 time=0.538 ms
64 bytes from 192.168.0.22: icmp_req=2 ttl=63 time=0.312 ms
64 bytes from 192.168.0.22: icmp_req=3 ttl=63 time=0.316 ms
^C
--- 192.168.0.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.312/0.388/0.538/0.108 ms
```

Détail pour les protocole TCP et UDP

Allez, il reste encore qu'une étape 

Il faut revenir sur les règles INPUT et OUTPUT que nous avons récupérée provisoirement du script pour une station du LAN.
Pour la passerelle on va affiner le filtrage des règles concernant TCP et UDP.
Ce sera l'occasion de mettre en place des règles pour le protocole FTP que nous n'avons pas encore abordé.

Nous aurons besoin pour cela d'un rappel sur le fonctionnement du protocole TCP

TCP un protocole en mode connecté

Cela signifie qu'avant tout échange de données applicatives, le client demande une connexion à un serveur et que le serveur accepte (ou refuse) la connexion.

Une fois connecté les possibilités d'échanges sont multiples :

1. Le client peut envoyer une requête et attendre la réponse, puis se déconnecter ou être déconnecté par le serveur.
2. Le client peut envoyer une requête, attendre la réponse et recommencer.
3. Le client peut envoyer plusieurs requêtes avant d'attendre les réponses, et recommencer ou pas.
4. Ce pourrait aussi être le serveur qui envoie des requêtes et le client qui répond.

Finalement, le seul point formalisé et obligatoire c'est la connexion d'un client à un serveur.

La connexion TCP

Voilà comment ça se passe en résumer :

1. Un port TCP local est attaché au client, ainsi que l'adresse IP locale de l'interface réseau utilisée.
2. Le client demande une connexion à une adresse IP distante et un port IP distant.

Il s'ensuit ce que l'on appelle : “the three-way handshake” ou “la poignet de main en trois messages” qui est déroulée.

- la poignet de main en trois messages :

1. La machine cliente génère une trame TCP/IP avec l'indicateur **SYN** (synchronisation) positionné.
2. La machine serveur répond avec une trame TCP/IP avec l'indicateur **ACK** (acknowledgement pour accusé de réception) positionné pour acquitter la trame SYN et l'indicateur **SYN** positionné.
3. Le client acquitte lui aussi l'indicateur **SYN** du serveur avec une trame TCP/IP avec l'indicateur **ACK** positionné.

Alors les échanges de données applicatives peuvent avoir lieu.

La déconnexion TCP

1. L'un des partenaires de la connexion, envoi une trame TCP/IP avec l'indicateur **FIN** positionné.
2. L'autre partenaire répond avec une trame TCP/IP avec l'indicateur **ACK** positionné.
3. Quand l'autre partenaire est prêt à terminer la connexion, il envoie une trame TCP/IP avec l'indicateur **FIN** positionné.
4. Cette trame est elle aussi répondue avec une trame TCP/IP avec l'indicateur **ACK** positionné.

Donc pour que la déconnexion soit effective, il faut que de chaque côté, il y ait eu une demande de déconnexion.

Le FTP

Le protocole ftp nécessite deux connexions

L'une pour échanger des commandes.
L'autre pour échanger les données à transférer. De plus, il existe le ftp actif et le ftp passif.

L'établissement d'une connexion TCP pour les commandes ftp vers le port 21 (ftp) est classique :

```
SYN (port ip cible 21) >
                          < ACK + SYN
                     ACK >
                        ...
           changes des commandes et réponses
                        ...
```

Le ftp actif

Dans ce cas, le client va envoyer au serveur une commande port avec le port à utiliser pour l'établissement d'une connexion pour le transfert des données.
Le serveur va alors se connecter vers le client sur le port indiqué par la commande port (3333 dans l'exemple ci-dessous).
Le port source pour l'échange des données sera la port 20 (ftp-data) :

```
Client / serveur -échange des commandes
   SYN (port ip cible 21) >
                           < ACK + SYN
                     ACK >
                         ...
                port 3333 >
                         ...

Client / Server - échange des données
                           < SYN (port ip cible 3333, port ip source 20)
                ACK + SYN >
                           < ACK
                         ...
               transfert des données
                         ...
```

Le ftp passif

Dans ce cas, c'est le serveur qui va envoyer au client une commande port avec le port à utiliser pour l'établissement d'une connexion pour le transfert des données.
Le client va alors se connecter vers le serveur sur le port indiqué par la commande port (3333 dans l'exemple ci-dessous).
Le port source pour l'échange des données n'est pas spécifique :

```
Client / Server - échange des commandes
   SYN (port ip cible 21) >
                           < ACK + SYN
                     ACK >
                         ...
                           < port 3333
                         ...

Client / Server - échange des données
  SYN (port ip cible 3333,
       port ip source x)  >
                           < ACK + SYN
                     ACK >
                         ...
               transfert des données
```

```
...
```

Le script de la passerelle

Tenons compte de ces rappels, pour ajouter à tout ce qu'on a fait jusqu'à présent, une connexion FTP et SSH loggées, et puisqu'on y est un petit filtrage supplémentaire au niveau des Flags TCP.

On ajoute au besoin les règles pour un serveur d'impression (cups ; port 631) ;
et de même, si l'on veut (commenter/décommenter), les règles NAT pour un proxy transparent pour le sous-réseau 162.168.1.0/24.
Pour plus de détail sur l'installation d'un proxy transparent voir proxy-transparent.

firewall_gateway.sh

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:       iptables
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:  2 3 4 5
# Default-Stop:        0 1 6
# Short-description:  iptables
# Description:        Firewall
### END INIT INFO
# start/stop iptables
#
# Author: hypathie <hypathie@debian-facile>
#
##Set up /etc/init.d/firewall_gateway.sh
case "$1" in
'start')
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -P INPUT DROP
/sbin/iptables -P OUTPUT DROP
/sbin/iptables -P FORWARD DROP
/sbin/iptables -t nat -P PREROUTING ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P INPUT ACCEPT
/sbin/iptables -t nat -P OUTPUT ACCEPT
/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
##commenter / décommenter et adapter les quatre lignes suivantes pour
ne pas mettre en place / mettre en place
##un proxy transparent (squid)
/sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT -
-to 192.168.0.1:3129
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j
```

```
REDIRECT --to-port 3129
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 3128 -j DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 3129 -j DROP
#accepter l'interface lo
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -o lo -j ACCEPT
#accepter le sous-réseau
/sbin/iptables -A INPUT -i eth1 -j ACCEPT
/sbin/iptables -A OUTPUT -o eth1 -j ACCEPT
#permettre le passage de tcp entre les deux interfaces eternet de la
passerelle
/sbin/iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -
d 0.0.0.0/0 -p tcp\
 -m state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p tcp\
 -m state --state ESTABLISHED -j ACCEPT
#Si pas de DNS configurer on dé-commente les règles udp suivantes :
#iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -d
0.0.0.0/0 -p udp\
# --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
#iptables -t filter -A FORWARD -i eth0 -o eth1 -s 0.0.0.0/0 -d
192.168.1.0/24 -p udp\
# --dport 53 -m state --state ESTABLISHED -j ACCEPT

#autoriser le ping avec le sous-réseau 192.168.1.0/24
/sbin/iptables -t filter -A FORWARD -p icmp -j ACCEPT

#accepter le ping entre les réseaux locaux
/sbin/iptables -t filter -A INPUT -p icmp -i eth0 -m conntrack --
ctstate ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -p icmp -o eth0 -m conntrack --
ctstate ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -t filter -A INPUT -p icmp -i eth1 -m conntrack --
ctstate ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -p icmp -o eth1 -m conntrack --
ctstate ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 0 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 0 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 3/4 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 3/4 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 3/4 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 3/3 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 3/3 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 3/3 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 3/1 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 3/1 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 3/1 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 4 -j ACCEPT
```

```
/sbin/iptables -A OUTPUT -p icmp --icmp-type 4 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 4 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 8 -m limit --limit 2/s -j
ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 8 -j LOG --log-prefix
"ICMP/in/8 Excessive: "
/sbin/iptables -A INPUT -p icmp --icmp-type 8 -j DROP
/sbin/iptables -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 8 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 11 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 11 -j ACCEPT
/sbin/iptables -A INPUT -p icmp --icmp-type 12 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp --icmp-type 12 -j ACCEPT
/sbin/iptables -A FORWARD -p icmp --icmp-type 12 -j ACCEPT
/sbin/iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.0.0/24 -p icmp -
-icmp-type echo-request -j ACCEPT
/sbin/iptables -A FORWARD -s 192.168.0.0/24 -d 192.168.1.0/24 -p icmp -
-icmp-type echo-reply -j DROP
/sbin/iptables -A INPUT -p icmp -m limit -j LOG --log-prefix "ICMP/IN:
"
/sbin/iptables -A OUTPUT -p icmp -m limit -j LOG --log-prefix
"ICMP/OUT: "
/sbin/iptables -N syn_flood
/sbin/iptables -I INPUT -p tcp --syn -j syn_flood
/sbin/iptables -A syn_flood -m limit --limit 1/s --limit-burst 3 -j
RETURN
/sbin/iptables -A syn_flood -j LOG --log-prefix '[SYN_FLOOD] : '
/sbin/iptables -A syn_flood -j DROP
#autoriser la connexion avec les serveurs DNS
/sbin/iptables -t filter -A OUTPUT -o eth0 -p udp -m udp --dport 53 -m
state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -i eth0 -p udp -m udp --sport 53 -m
state --state ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -o eth1 -p udp -m udp --dport 53 -m
state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -i eth1 -p udp -m udp --sport 53 -m
state --state ESTABLISHED -j ACCEPT
#autoriser la navigation web
/sbin/iptables -t filter -A OUTPUT -o eth0 -p tcp -m multiport --dports
80,443,8000 -m state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -i eth0 -p tcp -m multiport --sports
80,443,8000 -m state --state ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -o eth1 -p tcp -m multiport --dports
80,443,8000 -j ACCEPT
/sbin/iptables -A INPUT -i eth1  -p tcp -m multiport --sports
80,443,8000 -j ACCEPT
#Si le serveur cups est branché sur un ordinateur du réseau
192.168.0.0/24, par exemple sur 192.168.0.22
# laisser décommenter les deux lignes suivantes :
/sbin/iptables -A INPUT -i eth0 -s 192.168.0.22 -d 192.168.0.1 -p tcp -
```

```
-sport 631 -m state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -o eth0 -s 192.168.0.1 -d 192.168.0.22 -p tcp
--dport 631 -m state --state NEW,ESTABLISHED -j ACCEPT
#créer une chaîne utilisateur pour les connexion ssh, les loguer et les
accepter
/sbin/iptables -t filter -N InComingSSH
/sbin/iptables -I INPUT -i eth0 -s 192.168.0.0/24 -p tcp -m tcp --dport
22 -m conntrack --ctstate NEW,ESTABLISHED -j InComingSSH
/sbin/iptables -A InComingSSH -j LOG --log-prefix '[INCOMING_SSH] : '
/sbin/iptables -A InComingSSH -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -o eth0 -p tcp -m tcp --sport 22 -m
conntrack --ctstate ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -o eth1 -p tcp -m tcp --dport 22 -m
conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -i eth1 -s 192.168.0.0/24 -p tcp --
sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#créer une chaîne utilisateur pour les connexions ftp, et les accepter
/sbin/iptables -N ftp_in_accept
/sbin/iptables -I INPUT -i eth0 -p tcp --sport 21 -m state --state
ESTABLISHED,RELATED -j ftp_in_accept
/sbin/iptables -I INPUT -i eth0 -p tcp --sport 20 -m state --state
ESTABLISHED,RELATED -j ftp_in_accept
/sbin/iptables -I INPUT -i eth0 -p tcp --sport 1024:65535 --dport
1024:65535 -m state --state ESTABLISHED -j ftp_in_accept
/sbin/iptables -A ftp_in_accept -p tcp -j ACCEPT
/sbin/iptables -A INPUT -i eth1 -p tcp --sport 21 -m state --state
ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A INPUT -i eth1 -p tcp --sport 20 -m state --state
ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -I INPUT -i eth1 -p tcp --sport 1024:65535 --dport
1024:65535 -m state --state ESTABLISHED -j ACCEPT
echo "set up firewall_gateway.sh .........> [OK]"
/sbin/iptables-save > /etc/firewall_gateway.sh
echo "iptables-save > /etc/firewall_gateway.sh .........> [OK]"
RETVAL=$?
;;
'stop')
# Supprime toutes les règles de la tables FILTER et pose la police
ACCEPT pour toutes les chaînes
/sbin/iptables -t filter -F
/sbin/iptables -t filter -X
/sbin/iptables -t filter -P INPUT ACCEPT
/sbin/iptables -t filter -P OUTPUT ACCEPT
/sbin/iptables -t filter -P FORWARD ACCEPT
echo "FILTER [ALL firewall_gateway.sh's rules .... [FLUSH] ..... POLICY
......> [ACCEPT]"
echo "NAT ...POSTROUTING ... MASQUERADE .... [STILL SET UP]"
RETVAL=$?
;;
'restart')
```

```
#ré-installe le pare-feu complet, y compris NAT (masquerade), DNAT
(port 631)
/sbin/iptables-restore < /etc/firewall_gateway.sh
echo "/etc/firewall-client ........> [OK]"
echo "NAT (masquerade) ........> [OK]"
echo "DNAT (port 631) ........> [OK]"
RETVAL=$?
;;
'status')
/sbin/iptables -L -n --line-numbers
/sbin/iptables -t nat -L -n --line-numbers
RETVAL=$?
;;
'flush')
#supprime toutes les règles de toutes les tables ; accepte tout
/sbin/iptables -t filter -F
/sbin/iptables -t nat -F
/sbin/iptables -t mangle -F
/sbin/iptables -t raw -F
/sbin/iptables -t filter -P INPUT ACCEPT
/sbin/iptables -t filter -P OUTPUT ACCEPT
/sbin/iptables -t filter -P FORWARD ACCEPT
echo "FILTER [ALL RULES .......> [FLUSH]"
echo "WARNING ........ ALL POLICY ......> [ACCEPT]"
RETVAL=$?
;;
'deletnat')
/sbin/iptables -t nat -F
/sbin/iptables -t nat -X
/sbin/iptables -t mangle -F
/sbin/iptables -t nat -P PREROUTING ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P OUTPUT ACCEPT
/sbin/iptables -t mangle -P PREROUTING ACCEPT
/sbin/iptables -t mangle -P OUTPUT ACCEPT
/sbin/iptables -t mangle -P POSTROUTING ACCEPT

echo "NAT/MANGLE [ALL RULES .... [FLUSH] ..... POLICY ......> [ACCEPT]"
echo "INFO ......> [NAT/DNAT is OFF]"
echo "INFO ......> [FILTER STILL SET UP]"
RETVAL=$?
;;
*)
echo "Usage: $0 { start | stop | restart | status | flush | deletnat }"
RETVAL=1
;;
esac
exit $RETVAL
```

Pour autoriser la plateforme de jeux STEAM

```
/sbin/iptables -A INPUT -i eth1 -p udp -m udp --sport 27000 -m\
 state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -A INPUT -i eth0 -p udp -m udp --sport 27000 -m\
 state --state ESTABLISHED -j ACCEPT
```

Merci à robert2a pour cet ajout

Voir ce fil

- Pour l'installation du script

Suivre la même méthode que pour firewall-client comme script init.

- Ou si l'on préfère utiliser les commandes iptables-save et iptables-restore

Il n'y qu'à récupérer les commandes iptables du script, en commentant le reste.
Puis lancer les commandes nécessaires à l'exécution du script (droits d'exécution et en root `bash /chemin/du/script`)
Enfin éditer /etc/network/interfaces :

/etc/network/interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback

#carte vers la box
auto eth0
iface eth0 inet static
address 192.168.0.1
network 192.168.0.0
netmask 255.255.255.0
gateway 192.168.0.254
post-up iptables-restore < /etc/firewall_gateway

#carte vers le LAN
auto eth1
iface eth1 inet static
address 192.168.1.1
network 192.168.1.0
netmask 255.255.255.0
```

Le cas du serveur DHCP

Aurions-nous oublié une règle pour DHCP ?

Rassurez-vous, il est inutile d'ouvrir les ports UDP (67, 68) pour que le serveur DHCP installé sur cette passerelle puisse continué d'attribuer des IP à notre réseau B.

- “Ne pas ajouter de règle aveuglément !”

Voici une autre maxime à laquelle il faudra se tenir.

On voit dans de nombreux wiki l'ajout aberrant de cette règle :

```
IPTABLES -I INPUT -i <interface-interne> -p udp –dport 67:68 –sport 67:68 -j ACCEPT
```

Nous ne l'ajouterons pas !

- Dans notre script nous avons mis en effet :

```
iptables -A INPUT -i eth1 -j ACCEPT
iptables -A OUTPUT -o eth1 -j ACCEPT
```

Et nous allons à apprendre pour finir, une technique pour n'ajouter une règle que si elle fait réellement défaut. Il s'agit de l'outil tcpdump.

Utiliser tcpdump

- On l'installe :

```
apt-get install tcpdump
```

- Puis on lance sur le serveur :

```
tcpdump -i eth1 port 67 or port 68
```

Rien ne se passe... et c'est normal !

Pour voir quelque chose, il faut générer du flux depuis le client du réseau B.
On laisse le terminal (de la passerelle) en état d'attente, et on passe sur le terminal du client B.

- Côté **client** DHCP (ordinateur B) on annule le bail DHCP :

```
dhclient -r eth0
```

> Il n'y a plus d'IP :

```
eth0      Link encap:Ethernet  HWaddr 00:1e:0b:67:9b:b7
          adr inet6: xxxxxxxxxxxx Scope:Lien
```

- On lance une requête DHCP

```
dhclient eth0
```

- Côté serveur DHCP :

Observez ce que nous dit “tcpdump” :

```
tcpdump: verbose output suppressed, use -v or -vv for full
 protocol decode listening on eth1, link-type EN10MB (Ethernet),
capture size 65535 bytes 07:47:00.535348
IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
 Request from xx:xx:xx:xx:xx:xx (oui Unknown), length 300
07:47:00.535553 IP debian-serveur.mondomaine.hyp.bootps >
debian-hp.local.bootpc: BOOTP/DHCP, Reply, length 302
07:47:00.535745 IP 0.0.0.0.bootpc >
255.255.255.255.bootps: BOOTP/DHCP, Request from xx:xx:xx:xx:xx:xx
 (oui Unknown), length 300
07:47:00.626390 IP debian-serveur.mondomaine.hyp.bootps
 > debian-hp.local.bootpc: BOOTP/DHCP, Reply, length 302
^C
4 packets captured
4 packets received by filter
```

- Et l'IP de l'ordinateur client DHCP a bien changé !

```
INET_IP=`ifconfig $INET_IFACE | grep inet | \
cut -d : -f 2 | cut -d ' ' -f 1` && echo $INET_IP
```

```
192.168.1.4 127.0.0.1
```

Comme promis un petit rappel sur tcpdump.

(*Tout se lance en root*)

- Affichage standard: `tcpdump`
- Affichage verbeux (verbose): `tcpdump -v`
- Interfaces réseaux disponibles pour la capture: `tcpdump -D`
- Affichage des adresses numériques[6]): `tcpdump -n`
- Affichage rapide: `tcpdump -q`
- Capture du traffic d'une interface particulière: `tcpdump -i eth1`
- Capture du trafic UDP: `tcpdump udp`
- Capture du trafic du port TCP 80: `tcpdump port http`
- Capture du trafic à partir d'un fichier stocké dans un fichier: `tcpdump -F file_name`
- Envoi du résultat dans un fichier à la place de directement à l'écran: `tcpdump -w capture.log`
- Lecture d'un fichier de capture: `tcpdump -r capture.log`
- Arrêter la capture après n paquets : `tcpdump -c n`[7]

Pour aller plus loin : `man tcpdump`.

Et voilà, c'est fini ! 😁

La suite concernera le DNAT et SNAT, pour un serveur web apache installé sur un client du réseau B, et enfin, l'installation sur cette passerelle d'un proxy transparent pour l'ensemble su réseau B.

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

via un câble droit, et pourquoi pas, un câble droit, un switch, des câbles droits torsadés vers plusieurs ordinateurs B1, B2, B3 ...

3)

FQDN (Fully Qualified Domain Name

4)

nous ferons de même concernant ICMP pour les chaînes INPUT et OUTPUT plus bas quand toutes les chaînes de FILTER seront à DROP.

5)

on n'oublie sa table de routage (en root)

```
route add -net 192.168.1.0 gw 192.168.0.1 netmask 255.255.255.0 dev eth0
```

6)

plutôt que des adresses symboliques (DNS

7)

où n est un nombre, 10 par exemple

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:reseau:iptables-pare-feu-pour-une-passerelle

Last update: **01/11/2015 18:20**