iptables: un pare-feu pour un client

- Objet : Qu'est-ce qu'un pare-feu ? Comment le configurer ?
- Niveau requis :
  avisé
- Commentaires : *Contexte d'utilisation du sujet du tuto.*
- Suivi :
  à-tester
  - Création par Hypathie 08/10/2014
  - Testé par <...> le <...>
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Iptables est un logiciel libre grâce auquel l'administrateur système peut configurer les chaînes et règles du pare-feu en espace noyau.

Netfilter est un framework implémentant le pare-feu au sein du noyau Linux.
Voir le site de netfilter

Un pare-feu rejette les paquets du réseau selon des règles stockées dans la mémoire du PC.
Linux iptables travaille avec des «chaînes». Ces chaînes sont des regroupements de règles qui régissent le trafic réseau en fonction de trois axes : les paquets ; les protocoles ; les ports.

Un peu de théorie

Il faut connaître trois axes du flux d'une connexion internet pour comprendre le fonctionnement d'un pare-feu.

Paquet: un récipient de logique qui représente un flux de données.

Protocole: langage et ensemble de règles que les périphériques réseau utilisent.

Port: une désignation numérique qui représente un protocole particulier.

Un paquet ou un datagramme

est une représentation d'un phénomène physique. Il forme une unité de confinement où les données peuvent être examinées, acheminées et filtrées en pour qui concerne leur destination, leur origine et leur contenu.
Le schéma suivant représente un paquet de datagramme sur un réseau.

Voir le schéma d'un datagramme.

Un datagramme n'est pas un paquet réel, mais il représente visuellement le flux d'électrons ou de photons qui transmettent des données physiques sur un réseau. À nos yeux “nus”, nous ne voyons

que des variations de l'amplitude et de la fréquence tel que ce qui serait visible sur un oscilloscope.

Tous les services listés ci-dessous s'appuient sur des données de terrain transmises à chaque paquet. Le flux de données, appelé «trafic», est régi par des protocoles standards qui s'appuient sur des «ports» spécifiques. Chaque port peut être représenté comme ouvert ou fermé ; l'ouverture correspond à accepter un déplacement de flux (ou paquets), et la fermeture correspond à rejeter ce flux dont le champ de données correspond à ce port.

Ports TCP/UDP et Protocole couramment utilisés

Comme un pare-feu classique, iptables contrôle les ports sur une interface réseau par laquelle les paquets peuvent entrer, transiter, sortir, être rejetés.

Ce reporter à la documentation interne (lien en vert) pour la mise en place des services listés ci-dessous.
Vous y trouverez : leur installation et leur configuration, tel que la modification du port par défaut pour certains de ces protocoles.

Et à ce sujet, dans la suite de ce wiki, je considérerai qu'ils n'ont pas été modifiés.

FTP = TCP 21/20
SSH = TCP 22
Telnet = TCP 23
Web/http = TCP 80
SSL/https = UDP 443[2)]
DNS = UDP = 53
DHCP = UDP = 67 et 68
SAMBA = 137-139 et 445 (voir ci-dessous)
NETBIOS = 137-139
ACTIVE DIRECTORY = 445 NetBios/DNS
SMTP = (email out) 25[3)]
POP3 = email in) 110[4)]
IMAP = (email in) 143[5)]
VPN = 1723
KERBEROS = 88[6)]
SNMP =161/162

Un conseil ne pas mettre en place un pare-feu tout fait sans comprendre ce qu'on fait.
L'idéal est d'avoir quelques bases sur les réseaux.
Voici quelques liens indispensables pour acquérir quelques connaissances fondamentales :

- TCP/IP
- En-têtes TCP
- Caractéristiques ICMP
- Pour le pare-feu d'un client et le ping sur un client du réseau voisin : lire tout le chapitre
  - sur ICMP : http://www.inetdoc.net/guides/iptables-tutorial/icmpheaders.html
  - sur SCTP : les Caractéristiques SCTP et les En-têtes SCTP

Au sujet de la configuration réseau, commandes, fichiers de configuration pour GNU/linux:
http://www.linux-france.org/prj/edu/archinet/systeme/ch03.html

Enfin pour lister les protocoles connus du système :

```
cat /etc/protocols
```

Pour ce qui concerne le routage (table NAT), nous l'aborderons lors de la mise en place d'un pare-feu pour une machine faisant office de routeur, dans le wiki suivant.

Fonctionnement d'iptables

Pour apprendre à utiliser iptables et acquérir les différentes notions mises en œuvre lors de son utilisation, nous allons partir de l'exemple.

Nous allons commencer par analyser le retour de la commande iptables qui permet lister **la table filter**. C'est la table qui est listée par défaut quand aucune autre table n'a été mise en place.

Une **table** permet de définir le plan de “travail”. En effet, iptables ne sert pas uniquement à bloquer certains paquets et n'est pas uniquement utiliser pour dresser un pare-feu. On peut par exemple s'en servir modifier un paquet (table MANGLE), ou pour faire de la redirection de paquet (table NAT).

Mais c'est la table filter qui permet d'utiliser iptables en tant que pare-feu et qui nous intéresse ici. Avec cette table, on va indiquer à iptables un ensemble de règles dans un ordre précis afin qu'il sache s'il doit interdire ou autoriser le passage des paquets.

Dresser la table filter consistera à indiquer précisément à iptables pour quels paquets, pour quels ports ou protocoles, pour quelle direction... il y a autorisation ou interdiction de passage.

Lister les règles de la table filter

Installer sur le système, iptables n'est pas encore défini pour être un pare-feu et sa table filter est vide.

```
iptables -t filter --list
```

> **`-t table`** : pour demander une table en particulier[7)]

retour de la commande

```
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 première ligne `Chain INPUT (policy ACCEPT)`** concerne les paquets entrants **`INPUT`**. On voit que tout passe dans toutes les directions **`policy ACCEPT`**.
>
> **En dessous `target prot opt source destination`** ce sont les titres qui permettent de ranger le retour de la commande de listage dans un tableau bien lisible. Pratique pour lire plus facilement ce qui est mis en place :
>
> > **`target`** : c'est l'action à prendre, quand des règles seront mises en place, on trouvera dans cette colonne plusieurs lignes dont chacune aura l'une de ces valeurs :
> > **`ACCEPT`** (autoriser) ou **`DROP`** (interdire) ou **`REJECT`** (interdire en envoyant un signal d'interdiction).
> >
> > **`prot`** : pour protocole, par exemple UDP, TCP
> >
> > **`source`** : l'IP source de l'ordinateur distant qui se connecte à vous (pour INPUT)
> >
> > **`destination`** : l'IP de destination de l'ordinateur auquel on se connecte (pour OUTPUT)
>
> **La troisième ligne `Chain FORWARD (policy ACCEPT)`** concerne les paquets qui passent d'une interface à une autres d'un PC (**`FORWARD`**). La maîtrise par iptables de la chaîne FROWARD permet de rediriger le trafique.
> En dessous les mêmes titres de la futur table qui est vide pour l'instant.
>
> **La cinquième ligne `Chain OUTPUT (policy ACCEPT)`** concerne les paquets sortants (**`OUTPUT`**).

Lien utile

En complément, voir ce post sur le forum :

- https://debian-facile.org/viewtopic.php?id=18221

Quelques définitions

Une chaîne

Revenons sur notre table filter vide pour préciser ce qu'est une chaîne.

```
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
```

Nous voyons là trois chaînes (`chain`).
Une chaîne est un ensemble de règles définies dans un **ordre précis**.

Pour l'instant, il n'y n'a pas de règle mais une fois le pare-feu mis en place on pourrait avoir par exemple pour les entrées INPUT:

```
iptables -L --line-numbers Chain INPUT
```

retour de la commande

```
num  target   prot opt  source     destination
1    ACCEPT   tcp  --  anywhere    anywhere    tcp dpt:www
2    ACCEPT   tcp  --  anywhere    anywhere    tcp dpt:ssh
3    ACCEPT   tcp  --  anywhere    anywhere    tcp dpt:imap2
```

> **Il y a bien un ordre précis** :
>
> > La première règle (1) autorise ACCEPT, le protocole `tcp`, venant de partout `anywhere`, allant partout `anywhere`, et cela pour `www`, le web (ou port 80).
> >
> > La seconde (2) pour `ssh`
> >
> > la troisième (3)pour `imap2`
>
> Cet ordre est important car iptables lit de haut en bas, et le résultat n'est pas le même selon l'ordre des règles.

La dernière colonne indique après les deux point `:`, le protocole.
Avec l'option `-n`: `iptables -L -n` on aurait le numéro de port.

Une cible

Nous avons déjà énuméré les cibles possibles pour la table filter.
Il s'agit des valeurs qui apparaissent en dessous de `target`:
`ACCEPT` (autoriser) ou **`DROP`** (interdire) ou **`REJECT`** (interdire en envoyant un signal d'interdiction) .
Il faut savoir qu'il y en a d'autres selon les tables qu'on met en place.[8)]

Si target est l'action à mettre en place, les valeurs ACCEPT, DROP et REJECT sont des cibles parce que l'action d'iptables est d'envoyer un paquet quelque part, par exemple vers une application web quand le paquet est accepté; vers l'expéditeur quand il est refusé.

Nous allons mettre en place les chaînes INPUT et OUTPUT (c'est-à-dire plusieurs règles pour les entrées et les sorties des paquets), mais cela se fait en fonction du principe suivant.
D'abord on interdit tout, puis on laisse passer ce qui nous intéresse !

Pare-feu pour une station (client)

Effacer toutes nos règles

On sait jamais si quelqu'un avait mal configurer le pare-feu auparavant !
En plus au cours de l'apprentissage, mieux savoir défaire ce qu'on peut avoir mal fait...

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

-F : (flush) : vider toutes les chaînes existantes
-X : supprimer les chaînes personnelles

Remettre la police par défaut (ACCEPT)

```
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

Commandes pour tout interdire

- Suivons le principe énoncé plus haut : on commence par tout interdire.

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

Une autre commande d'iptables:
-P *chain* : définition de la politique par défaut (DROP : interdire) pour la chaîne INPUT ; puis pour OUTPUT ; puis pour FORWARD

- Voyons le résultat :

```
iptables -L -n --line-numbers
```

retour de la commande

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination

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

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

Autoriser les échanges avec les serveurs DNS

```
iptables -t filter -A OUTPUT -p udp --dport 53 -m conntrack --ctstate NEW
-j ACCEPT
iptables -t filter -A INPUT  -p udp --sport 53 -m conntrack --ctstate
ESTABLISHED -j ACCEPT
```

Signification de ces commandes

Les commandes précédentes autorisent un paquet sortant (chaine OUTPUT) et entrant (CHAINE INPUT), via le protocole udp sur le port 53 et selon l'état de la connexion (NEW ou ESTABLISHED).

Les options

`-t filter` : pour préciser la table (facultatif avec la table filter).
`-A chain` : ajouter une règle à la fin de la chaîne (OUTPUT, puis INPUT)
`-p` : pour indiquer le type de trames utilisé dans le paquet ([!] “all”, “tcp”, “udp”, “icmp”, ou un numéro).
`–dport` : port de destination.
`–sport` : port source.
`-m module` : Demande d'utiliser un module particulier.
`conntrack` : c'est un module de netfilter.
`–ctstate` : liste des états conntrack.

La liste des états de connexion conntrack

- INVALID : signifie que le paquet est associé à aucune connexion connue
- NEW : signifie que le paquet a commencé une nouvelle connexion
- ESTABLISHED : ce qui signifie que le paquet est associé à une connexion qui a vu les paquets dans les deux sens.
- RELATED : signifie que le paquet commence une nouvelle connexion (comme l'état NEW), mais est associé à une connexion existante, commencée, en général, sur un autre port ou avec un autre protocole réseau. L'état RELATED ne s'applique que pour des protocoles réseaux bien précis (amanda, ftp, h323, icmp, irc, netbios, pptp, sane, sip, tftp).
- UNTRACKED : signifie que le paquet n'est pas suivi du tout.
- SNAT : Un état virtuel, le paquet correspond si l'adresse source diffère la réponse de destination.
- DNAT : Un état virtuel, le paquet correspond si l'adresse de destination diffère de l'adresse source.
- EXPECTED : Cette connexion est prévue (c'est à dire un assistant conntrack est activé)
- SEEN_REPLY : Conntrack a vu passer des paquets dans les deux directions.
- ASSURED : ok si l'entrée de conntrack n'a expiré
- CONFIRMED : Connexion est confirmée

Depuis debian **Stretch**, **conntrack** ne va plus marquer de paquets en **RELATED**, sauf pour le protocole **icmp**.

Par conséquent, si vous voulez faire du filtrage sur des paquets dans l'état **RELATED** pour d'autres protocoles qu'**icmp**, vous devrez créer une règle particulière, de **PREROUTING**, dans la table **raw**.

Cette règle va déclencher l'activation d'un module **conntrack** (appelé module **helper**) en cas de connexion sur un port particulier.

A noter, la cible de cette règle s'appelle **CT** (Pour **C**onn**T**rack ?).

Par exemple, pour le protocole **ftp**, si vous souhaitez qu'une nouvelle connexion **tcp** entrante, vers le port 21 du serveur, puisse permettre une réponse dans l'état **RELATED** :

```
iptables -t raw -A PREROUTING -p tcp --dport 21 -j CT --helper ftp
```

Pour information, la règle précédente va activer le module **helper** *nf_conntrack_ftp*.

Vous devez aussi autoriser de nouvelles connexions entrantes sur le port 21 :

```
iptables -A INPUT -p tcp --dport 21 -m conntrack --ctstate NEW -j ACCEPT
```

Conntrack associera l'état **RELATED** au nouveau paquet **tcp** sortant du port 20 du serveur (cas d'un serveur ftp actif). Vous devez, donc, autoriser la connexion :

```
iptables -A OUTPUT -p tcp --sport 20 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
```

Vous devez, donc, toujours, traiter les paquets étant dans l'état **RELATED** à partir du moment où ils existent, sinon ils seront perdus et cela entrainera des problèmes de communication !

Si vous ne souhaitez pas activer le module helper, qui gère l'état **RELATED**, **conntrack** associera les paquets à l'état **NEW**. Vous devrez, donc, utiliser la règle suivante :

```
iptables -A OUTPUT -p tcp --sport 20 -m conntrack --ctstate NEW,ESTABLISHED
-j ACCEPT
```

Pourquoi préférer l'état **RELATED** à l'état **NEW** ?
Parce qu'il est plus sûr car il n'autorise de nouvelles connexions que si une connexion préalable existait déjà sur un autre port ou avec un autre protocole.

Vous n'avez rien compris ? C'est normal, voyons ce qu'est *conntrack* !

La machine d'état
Cela permet le traçage de connexion. Le traçage de connexion est effectué afin que l'architecture de Netfilter puisse connaître l'état d'une connexion spécifique.
Les pare-feux qui implémentent ceci sont habituellement appelés pare-feux à état.
Le module conntrack permet justement d'effectuer du traçage de connexion !

Il prend en charge les protocoles TCP, UDP et ICMP.
Le traçage de connexion est entièrement pris en charge dans la chaîne PREROUTING, sauf pour les paquets générés en local, qui sont pris en charge dans la chaîne OUTPUT. Ceci signifie qu'iptable effectue tous les calculs d'état dans la chaîne PREROUTING. Si on envoie le premier paquet d'un flux, l'état est défini comme NEW dans la chaîne OUTPUT, et quand on reçoit un paquet de réponse, l'état passe à ESTABLISHED, et ainsi de suite. Si le premier paquet n'est pas envoyé par nous-mêmes, l'état NEW est naturellemt défini dans la chaîne PREROUTING. Ainsi, tous les changements d'état et calculs sont réalisés dans les chaînes PREROUTING et OUTPUT de la table nat.

Pour l'espace utilisateur les états valides sont NEW, ESTABLISHED, RELATED et INVALID.

Voir la traversée des tables et des chaînes.

La machine d'état: connexion TCP </note>

Commandes pour autoriser le réseau local

```
iptables -t filter -A OUTPUT -o lo -j ACCEPT
iptables -t filter -A INPUT -i lo -j ACCEPT
```

Pour pouvoir pinguer sur son voisin

Y compris sur la passerelle 192.168.0.1 depuis ce client 192.168.0.22.
Il faut autoriser le protocole icmp sur l'interface ethernet du client 192.168.0.22 sur lequel on installe le pare-feu.

```
iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A OUTPUT -o eth0 -p icmp -j ACCEPT
```

Commandes précédentes, ou celles-ci plus précises que l'on préférera :

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

iptables -A INPUT -i eth0 -p icmp -m conntrack\
 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT
```

Pour naviguer sur le web

```
iptables -t filter -A OUTPUT -p tcp -m multiport\
 --dports 80,443,8000 -m conntrack --ctstate\
 NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -p tcp -m multiport\
 --sports 80,443,8000 -m conntrack --ctstate\
 ESTABLISHED -j ACCEPT
```

Pour IMAP et SMTP (utilisation de messagerie icedove)

```
iptables -A INPUT -m multiport -p tcp --sport 25,2525,587,465,143,993,995 -m
state --state ESTABLISHED -j ACCEPT

iptables -A OUTPUT -m multiport -p tcp --dport 25,2525,143,465,587,993,995 -
m state --state NEW,ESTABLISHED -j ACCEPT
```

Pour la liste des ports selon les serveurs de messagerie utilisés voir : list of smtp and imap servers mailserver

Pour installer une imprimante partagée

- Si l'on veut brancher son imprimante sur la machine d'IP 192.168.0.22 faisant alors office de serveur d'impression pour le réseau local 192.168.0.0/24 :

```
iptables -A INPUT -i eth0 -s 192.168.0.0/24 -d 192.168.0.22 -p tcp\
 --dport 631 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth0 -s 192.168.0.22 -d 192.168.0.0/24 -p tcp\
 --sport 631 -m state ! --state INVALID -j ACCEPT
```

Remarque sur la place du point d'exclamation, qui ne se place pas devant `INVALID` comme on le voit souvent !
Si l'on veut que cette imprimante soit partagée aussi par un autre réseau, par exemple 192.168.1.0/24 on ajoutera ceci au pare-feu du client :

```
iptables -A INPUT -i eth0 -s 192.168.1.0/24 -d 192.168.0.22 -p tcp\
 --dport 631 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth0 -s 192.168.0.22 -d 192.168.1.0/24 -p tcp\
 --sport 631 -m state ! --state INVALID -j ACCEPT
```

Dans ce cas au niveau de la passerelle : Si elle dotée d'un pare-feu, il faudra que **son** pare-feu permette le trafic pour le port 631, c'est-à-dire accepter sur eth0 (réseau 192.168.0.0/24) les entrées et les sorties tcp sur le port 631.

Par exemple :

```
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
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
```

Voir : Script d'une passerelle

Vérifier et faire des tests

Lister ce qu'on a mis en place

```
iptables -t filter -L -n -v
```

ou

```
iptables -L -n --line-numbers
```

retour de la commande

```
Chain INPUT (policy DROP 80 packets, 19279 bytes)
 pkts bytes target  prot opt in    out   source        destination
    1   131 ACCEPT  udp  --  *      *    0.0.0.0/0     0.0.0.0/0   udp\
     spt:53 ctstate RELATED,ESTABLISHED

    3   252 ACCEPT  icmp --  *      *    192.168.0.0/24  0.0.0.0/0
 1296 1026K ACCEPT  tcp  --  *      *    0.0.0.0/0       0.0.0.0/0
multiport\
sports 80,443,8000 ctstate RELATED,ESTABLISHED

    4   336 ACCEPT  all  --  lo     *    0.0.0.0/0      0.0.0.0/0

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

Chain OUTPUT (policy DROP 1 packets, 67 bytes)
 pkts bytes target   prot opt in   out   source        destination
   1  56 ACCEPT   udp  --  *     *    0.0.0.0/0        0.0.0.0/0   udp
dpt:53\
ctstate NEW,RELATED,ESTABLISHED

   3  252 ACCEPT   icmp --  *     *    0.0.0.0/0      192.168.0.0/24
 815  639K ACCEPT   tcp  --  *     *   0.0.0.0/0      0.0.0.0/0
multiport\
dports 80,443,8000 ctstate NEW,RELATED,ESTABLISHED

   4  336 ACCEPT   all  --  *     lo    0.0.0.0/0     0.0.0.0/0
```

Faire des tests

- Sur l'interface lo

```
ping localhost
```

retour de la commande

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_req=1 ttl=64 time=0.050 ms
64 bytes from localhost (127.0.0.1): icmp_req=2 ttl=64 time=0.042 ms
^C
--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.042/0.046/0.050/0.004 ms
```

Ça marche

- Vers un autre client :

```
ping 192.168.0.1
```

retour de la commande

```
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=0.226 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=0.209 ms
^C
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.209/0.217/0.226/0.017 ms
```

Tout va bien

- Ping sur un domaine :

```
ping google.fr
```

retour de la commande

```
PING google.fr (173.194.40.152) 56(84) bytes of data.
64 bytes from par10s10-in-f24.1e100.net (173.194.40.152): icmp_req=1
ttl=55 time=34.0 ms
^C
--- google.fr ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1000ms
rtt min/avg/max/mdev = 34.027/34.027/34.027/0.000 ms
```

Connexions ssh entrantes/sortantes journalisées

Création de chaînes utilisateur

- Création de la chaîne utilisateur sshchain :

```
iptables -t filter -N OutGoingSSH
```

- On transfère l'étude de tous les paquets destinés à ssh (ou 22 si on a laissé le port ssh par défaut) depuis la chaîne INPUT vers OutGoingSSH.

Penser à mettre le port choisi s'il a été modifié dans /etc/ssh/sshd_config pour remplacer le port par défaut (22). De même, s'il a été modifié on ne peut pas utiliser "ssh" à la place du port.

```
iptables -I INPUT -p tcp --dport 22 -j OutGoingSSH
```

- On facilite la lecture de ses logs en ajoutant un préfixe :

```
iptables -A OutGoingSSH -j LOG --log-prefix '[OUTGOING_SSH] : '
```

- On logue dans une autre chaîne utilisateur les connexions sortantes :

```
iptables -t filter -N InComingSSH
iptables -I OUTPUT -p tcp --sport 22 -j InComingSSH
iptables -A InComingSSH -j LOG --log-prefix '[INCOMING_SSH] : '
```

Autorisations entrantes/sortantes de connexion ssh

- On autorise les connexions ssh à sortir vers un serveur ssh :

```
iptables -t filter -A INPUT -i eth0 -s 192.168.0.0/24\
 -p tcp -m tcp --dport 22 -m conntrack\
  --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -o eth0 -d 192.168.0.0/24\
 -p tcp -m tcp --sport 22 -m conntrack\
 --ctstate ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 22 -j ACCEPT
```

- Pour être connecté en ssh sur le client sur lequel est installé à la fois le pare-feu et openssh-server:

```
iptables -t filter -A OUTPUT -o eth0 -p tcp -m tcp --dport 22 -m conntrack\
  --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -t filter -A INPUT -i eth0 -p tcp --sport 22 -m conntrack\
 --ctstate ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 22 -j ACCEPT
```

Explications pour ce qui concerne l'autorisation des connexions ssh entrantes et sortantes :

Il faut savoir ceci : Dans les en-têtes TCP, le bit 16 - 31 concerne le destinataire.
Et certes, à l'origine, il était directement (et seulement) lié au processus du système de réception. Mais aujourd'hui, pour augmenter la sécurité de l'échange, et pour nous permettre d'avoir d'avantage de connexions ouvertes en même temps, une empreinte numérique est utilisée.
Quand un paquet est reçu, les ports source et destination sont renversés dans la réponse à l'hôte d'origine, afin d'être mémorisé. Ainsi le port de destination est maintenant le port source, et le port source devient le port de destination. (Voir RFC 793 ; RFC 1349 ; RFC 2474 ; RFC 3168)

Donc façon générale (sans tenir compte de la journalisation et de l'utilisation du module *conntrack*):

- Pour les connexions sortantes la troisième ligne de commandes ci-dessous est **nécessaire** pour établir la connexion:

```
iptables -A INPUT -i eth0 -p tcp --dport 22\
 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A OUTPUT -o eth0 -p tcp --sport 22\
 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 22 -j ACCEPT
```

Que l'on peut affiner en spécifiant la plage d'adresse du réseau :

```
iptables -A INPUT -i eth0 -p tcp -s 192.168.0.0/24\
 --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --sport 22\
 -m state --state ESTABLISHED -j ACCEPT
```

Dans l'exemple ci-dessus, au lieu de /24 (IP/mask), vous pouvez également utiliser le masque de sous-réseau complet (i.e. “192.168.0.0/255.255.255.0”).

- **Pour les connexions entrantes** une troisième ligne de type
  `iptables -A INPUT -i eth0 -p tcp –sport 22 -j ACCEPT`
  serait **inutile**.
  Les commandes ci-dessous sont suffisantes pour établir la connexion ssh d'un voisin sur la machine sur laquelle on installe ce pare-feu :

```
iptables -A OUTPUT -p tcp --dport 22 -m state\
 --state NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp -s 192.168.0.0/24 --sport 22 -m state\
 --state ESTABLISHED -j ACCEPT
```

- Pour vérifier ses logs :

On se connecte en ssh du client sur lequel on a installé le pare-feu vers un client de notre réseau; puis dans l'autre sens d'un client du réseau vers le client “pare-feuté” qui est aussi serveur ssh.

```
less /var/log/messages | grep OUTGOING_SSH
```

retour de la commande

```
<...>
Oct  8 12:01:07 debian-pc1 kernel: [16793.633030] [OUTGOING_SSH] :
IN=eth0 OUT= MAC=xxx..
SRC=192.168.0.1 DST=192.168.0.22 LEN=52 TOS=0x10 PREC=0x00 TTL=64
ID=13465
 DF PROTO=TCP SPT=48542 DPT=22 WINDOW=1523 RES=0x00 ACK URGP=0
<...>
```

```
less /var/log/messages | grep INCOMING_SSH
```

retour de la commande

```
<...>
Oct  8 12:01:07 debian-pc1 kernel: [16793.632822] [INCOMING_SSH] : IN=
OUT=eth0
 SRC=192.168.0.22 DST=192.168.0.1 LEN=52 TOS=0x10 PREC=0x00 TTL=64
 ID=35238 DF PROTO=TCP SPT=22 DPT=48542 WINDOW=431 RES=0x00 ACK FIN
URGP=0
<...>
```

Sauvegarder ses règles

En l'état actuel, si on éteint le système au redémarrage nos règles et tables utilisateur auront disparu. Pour les sauvegarder, il y a plusieurs méthodes. On peut créer un script de démarrage, ou se servir de les commandes iptables-save et iptables-restore.
On choisira d'abord la seconde méthode car c'est la plus simple !

Avec les commandes iptables-save et iptables-restore

La commande iptables-save crée un fichier et la commande iptables-restore charge la dernière sauvegarde à partir du fichier crée par iptables-save.

Pour sauvegarder toutes les règles

```
iptables-save > /etc/firewall-client
```

Pour les charger après le redémarrage

```
iptables-restore < /etc/firewall-client
```

Bon il ne faut pas oublier de charger son pare-feu à chaque redémarrage !

Évitons ce risque et surtout de nous fatiguer…

Pour restaurer automatiquement les règles au démarrage

On va installer la commande iptables-restore avant que les interfaces ethernet ne soient chargés via le fichier /etc/network/interfaces.

```
vim /etc/network/interfaces
```

interfaces

```
# The loopback network interface
```

```
auto lo
iface lo inet loopback
post-up iptables-restore < /etc/firewall-client
```

Avantages de cette méthode

- La possibilité de créer des règles personnalisées pour chaque interface.

Si les règles sont indépendantes des interfaces, on place la commande iptables-restore en pre-up de la boucle locale.

On peut créer plusieurs fichiers à restaurer, un pour chaque interface par exemple, ou un fichier particulier pour un service particulier (un pour masquerade et squid, un autre pour un pare-feu)...

Ainsi par exemple le jour où on ajoutera une carte wifi sur son vieil ordi fixe, on se teste quelques nouvelles règles iptables, puis quand elles sont tip-top, on les sauvegarde en créant un `iptables-save > /etc/iptables-wifi`.

On ne restera plus qu'à ajouter `pre-up iptables-restore < /etc/iptables-wifi` avant la configuration de l'interface wlan dans /etc/network/interfaces sans avoir à modifier ce qui fonctionnait déjà...

- Une grande facilité de modification des règles du pare-feu :

Si pour une raison ou une autre il est nécessaire d'ajouter une règle au pare-feu, ou d'en supprimer une pour la modifier, on peut utiliser la commande iptables de suppression :

Par exemple :
Lister les règles avec un numéro de ligne :

```
iptables -L INPUT -n --line-numbers
```

Supprimer une ligne particulière (avec le numéro de la colonne de gauche) :

```
iptables -D INPUT numeroDeLaLigneASupprimer
```

Dans ce cas pour sauvegarder les nouvelles règles il n'y a qu'à relancer :

```
iptables-save > /etc/même_nom_de_fichier_que_celui_modifier
```

Ainsi cela évite de modifier un quelconque fichier, et l'on peut se servir de toutes les commandes fournies par sa distribution.

Pour les fadas du "scripting" shell

Le script suivant permettra de mettre en place le pare-feu à chaque démarrage du système.
Donc c'est soit la méthode de sauvegarde des règles iptables précédente, soit celle-ci.

Si vous avez testé la première méthode, ne pas oublier de supprimer ou de commenté la ligne **#post-up iptables-restore < /etc/firewall-client** du fichier /etc/network/interfaces.

- Ce script permettra :

```
/etc/init.d/firewall-client.sh
```

```
Usage: /etc/init.d/firewall-client.sh { start | stop | restart | status }
```

- Par exemple pour supprimer toutes les règles de toutes les tables :

```
/etc/init.d/firewall-client.sh stop
```

- Avec “restart” pour permettre de remettre en place toutes les règles (via le fichier de sauvegarde d'iptables-restore)
- Avec “start” pour relancer toutes les règles avec le script lui-même et les sauvegarder
- Avec “status” pour l'affichage des règles FILTER et NAT)

Il faut prendre conscience que si l'on souhaite modifier les règles du pare-feu, il est nécessaire après avoir effectué ses tests d'aller éditer le fichier /etc/init.d/firewall-client pour y apporter les modifications souhaitées aux commandes iptables du script.

Pour ce faire :

update-rc.d -f firewall-client.sh remove

vim /etc/init.d/firewall-client.sh

→ On modifie ce qu'on veut

update-rc.d -f firewall-client.sh default

Si l'on n'inscrit pas toute modification dans /etc/init.d/firewall-client.sh ainsi qu'une réinitialisation de ipdate-rc, comme ci-dessus, la simple commande iptables-save lancée après toutes modifications de règles iptables seraient écrasées au prochain redémarrage du système qui ré-installera les règles du script /etc/init.d/firewall-client laissées en l'état.

firewall-client.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-client
case "$1" in
'start')
##Set up firewall-client
# Clear any FILTER existing rules
/sbin/iptables -F
# Delete all User-specified chains
/sbin/iptables -X
#set default policy to DROP
/sbin/iptables -P INPUT DROP
/sbin/iptables -P OUTPUT DROP
/sbin/iptables -P FORWARD DROP
# Allow trafic with DNS server
/sbin/iptables -t filter -A OUTPUT -p udp --dport 53 -m conntrack --
ctstate NEW         -j ACCEPT
/sbin/iptables -t filter -A INPUT  -p udp --sport 53 -m conntrack --
ctstate ESTABLISHED -j ACCEPT
#Allow trafic on internal network
/sbin/iptables -t filter -A INPUT -i lo -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -o lo -j ACCEPT
#Allow ping to internal network
/sbin/iptables -A OUTPUT -o eth0 -p icmp -m conntrack --ctstate
NEW,ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A INPUT -i eth0 -p icmp -m conntrack --ctstate
NEW,ESTABLISHED,RELATED -j ACCEPT
#Get web
/sbin/iptables -t filter -A OUTPUT -p tcp -m multiport --dports
80,443,8000 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -p tcp -m multiport --sports
80,443,8000 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#Allow mailing protocols (IMAP and SMTP)
/sbin/iptables -A INPUT -m multiport -p tcp --sport
25,2525,587,465,143,993,995 -m state --state ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -m multiport -p tcp --dport
25,2525,143,465,587,993,995 -m state --state NEW,ESTABLISHED -j ACCEPT
#Allow cups
iptables -A INPUT -i eth0 -s 192.168.0.0/24 -d 192.168.0.22 -p tcp --
dport 631 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth0 -s 192.168.0.22 -d 192.168.0.0/24 -p tcp --
sport 631 -m state ! --state INVALID -j ACCEPT
#Allow cups from sub-net
/sbin/iptables -A INPUT -i eth0 -s 192.168.1.0/24 -d 192.168.0.22 -p
tcp --dport 631 -m state --state NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -o eth0 -s 192.168.0.22 -d 192.168.1.0/24 -p
tcp --sport 631 -m state ! --state INVALID -j ACCEPT
#Set up a user chain for ssh outgoing
```

```
/sbin/iptables -t filter -N OutGoingSSH
/sbin/iptables -I INPUT -p tcp --dport 22 -j OutGoingSSH
/sbin/iptables -A OutGoingSSH -j LOG --log-prefix '[OUTGOING_SSH] : '
#Set up a user chain for ssh incoming
/sbin/iptables -t filter -N InComingSSH
/sbin/iptables -I OUTPUT -p tcp --sport 22 -j InComingSSH
/sbin/iptables -A InComingSSH -j LOG --log-prefix '[INCOMING_SSH] : '
#Allow ssh connection from inside to outside
#Change in nexts lines this range ip "192.168.0.0/24" with your
internal network
/sbin/iptables -t filter -A INPUT -i eth0 -s 192.168.0.0/24 -p tcp -m
tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A OUTPUT -o eth0 -p tcp -m tcp --sport 22 -m
conntrack --ctstate ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -o eth0 -p tcp --dport 22 -j ACCEPT
# Allow ssh connection form external to inside
/sbin/iptables -t filter -A OUTPUT -o eth0 -p tcp -m tcp --dport 22 -m
conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
/sbin/iptables -t filter -A INPUT -i eth0 -s 192.168.0.0/24 -p tcp --
sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
echo "set up firewall-client .........> [OK]"
/sbin/iptables-save > /etc/firewall-client
echo "iptables-save > /etc/firewall-client .........> [OK]"
RETVAL=$?
;;
'stop')
# delete any existing rules
/sbin/iptables -t filter -F
/sbin/iptables -t nat -F
/sbin/iptables -t mangle -F
/sbin/iptables -t raw -F
/sbin/iptables -t filter -P INPUT ACCEPT
/sbin/iptables -t filter -P OUTPUT ACCEPT
/sbin/iptables -t filter -P FORWARD ACCEPT
echo "FILTER [ALL RULES .... [FLUSH] ..... POLICY ......> [ACCEPT]"
/sbin/iptables -t nat -P PREROUTING ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P OUTPUT ACCEPT
/sbin/iptables -t mangle -P PREROUTING ACCEPT
/sbin/iptables -t mangle -P OUTPUT ACCEPT
/sbin/iptables -t mangle -P POSTROUTING ACCEPT
/sbin/iptables -t mangle -P FORWARD ACCEPT
/sbin/iptables -t mangle -P INPUT ACCEPT
/sbin/iptables -t raw -P OUTPUT ACCEPT
/sbin/iptables -t raw -P PREROUTING ACCEPT
echo "ALL TABLES ....[FLUSH] ..... ALL POLICY .......> [ACCEPT]"
RETVAL=$?
;;
'restart')
/sbin/iptables-restore < /etc/firewall-client
echo "/etc/firewall-client .........[OK]"
```

```
RETVAL=$?
;;
'status')
/sbin/iptables -L -n --line-numbers
/sbin/iptables -t nat -L -n --line-numbers
RETVAL=$?
;;
*)
echo "Usage: $0 { start | stop | restart | status }"
RETVAL=1
;;
esac
exit $RETVAL
```

Instalation de firewall-client.sh comme script init

- Téléchargé firewall-client.sh.
- Déplacer le fichier firewall-client.sh dans le répertoire d'init :

Attention de bien changer $USER par votre nom d'utilisateur !
Pour pouvoir écrire (i.e. créer un fichier dans un dossier root, il faut lancer la commande en tant que root. Et donc $USER serait l'utilisateur root qui n'a pas de fichier “Téléchargements”.

```
mv /home/$USER/Téléchargements/firewall-client.sh /etc/init.d/
```

- Définir les droits d'exécution et l'appartenance du fichier /etc/init.d/firewall-client.sh

Ils doivent être :

```
chmod 0755 /etc/init.d/firewall-client.sh
```

```
chown root:root /etc/init.d/firewall-client.sh
```

- Faire en sorte qu'init le prenne en compte à chaque redémarrage

```
update-rc.d firewall-client.sh defaults
```

- Pour démarrer :

```
update-rc.d firewall-client.sh defaults
```

Et si tout c'est bien passé

Au prochain redémarrage, pendant la mise en route du système :

- Le pare-feu se met en place pendant le démarrage du système

```
set up firewall-client .........> [OK]
```

```
iptables-save > /etc/firewall-client .........> [OK]
```

- On peut le vérifier :

```
iptables -L -n --line-numbers
```

Ou (plus complet) :

```
/etc/init.d/firewall-client.sh status
```

Tout est là ! 😎

- Pour simplement annuler :

```
update-rc.d -f firewall-client.sh remove
```

- Pour supprimer (après avoir annuler) et préférer la méthode précédente:

```
cd /etc/init.d/
```

Puis pour la commande suivante (avec l'auto-complétion c'est rassurant) :

```
rm firewall-client.sh
```

Et voilà c'est déjà fini 🙂

Prochain n° sur iptables : “un pare-feu pour une passerelle Debian”.

Récapitulatif, commandes, options, syntaxe

Les commandes

commandes	**descriptions**
-A chain	ajouter une chaîne chain où **chain** est soit `INPUT` soit `OUTPUT` soit `FORWARD` ...
-D chain	effacer une chaîne chain (idem)
-I chain [no-regle]	insérer une règle dans la chaîne chain
-R chain no-regle	remplacer une règle dans la chaîne chain
-L chain	lister les règles (chaîne filter par défaut)
-L -t typetable	lister les règles d'un type de table où typetable est soit `FILTER` soit `NAT` soit `MANGLE` ...
-E [chain]	renommer la chaîne chain
-F [chain]	effacer (flush) les règles de la chaîne chain

commandes	**descriptions**
-N chain	créer une nouvelle chaîne chain par l'usager
-X chain	effacer la chaîne chain de l'usager
-P chain target	définir la politique par défaut de la chaîne chain où **target** est soit ACCEPT soit DROP soit REJECT
-Z [chain]	remise à zéro des compteurs dans la chaîne chain si elle est indiquée

Les options

options	**descriptions**
-p [!] protocle	où protocole est : TCP, UDP, ICMP, ALL
-s [!] adresse [/mask]	adresse source
–sport [!] [!][port[:port]]	port source
-d [!] adresse [/mask]	adresse destination
–dport [!] [!][port[:port]]	port de destination
-i [!] interface	interface d'entrée
-o [!] interface	interface de sortie
-j target	cible

Pour aller plus loin sur les commandes et options : `man iptables`
http://www.inetdoc.net/guides/iptables-tutorial/commands.html#table.commands
http://olivieraj.free.fr/fr/linux/information/firewall/fw-03-05.html

La syntaxe générale

Commande iptables basique

- Ouvrir une connexion vers sa machine

```
iptables -A INPUT -i <interface_d'entrée> -p <nom_du_protocole> --dport
<nom_du_port_de_destination> -j ACCEPT
```

> Où <nom_du_protocole> est à remplacer par TCP, UDP, all ;
> une règle pour chaque protocole
> et <nom_du_port> est le **nom du service**, tels FTP, DNS ... **ou le numéro** qui leur correspond tels 21, 53...

Règles plus précise

- Sur l'entrée :

```
iptables -A INPUT -i <interface_d'entrée> -s <reseau_local/masque_de_reseau>
-d <ip_locale> -p <nom_du_protocole> --dport <nom_du_port> -m state --state
NEW -j ACCEPT
```

> Ce qui rentre sur sa carte ethernet, dont la source est seulement <reseau_local/masque_de_reseau> et seulement à destination de <ip_locale>, dont l'état est une nouvelle connexion par tel protocole et sur tel port, sera accepté.

- Sur la sortie :

```
iptables -A OUTPUT -o eth0 -s <ip_locale> -d <reseau_local/masque_de_reseau>
-p <nom_du_protocole> --sport <nom_du_port> -m state --state NEW -j ACCEPT
```

> Ce qui sort de sa carte ethernet, dont la source est seulement <reseau_local/masque_de_reseau> et seulement à destination de <ip_locale>, dont l'état est une nouvelle connexion par tel protocole et sur tel port, sera accepté.

Filtre encore plus restrictif

- En entrée :

```
iptables -A INPUT -i eth0 -s <reseau_local/masque_de_reseau> -d <ip_locale>
-p <nom_du_protocole> --dport <nom_du_port> -m state --state ! INVALID -j
ACCEPT
```

> On précise que l'on accepte, en entrée sur l'interface réseau, tout ce qui vient de notre réseau local à destination de l'adresse ip, correspondant à notre interface réseau utilisant tel protocole, vers tel numéro de port (ou service) dont l'état du paquet n'est pas invalide, c'est-à-dire un paquet ayant l'état 'nouveau', 'établi', 'relatif' ou 'non traqué'.

- En sortie :

```
iptables -A OUTPUT -o eth0 -s <ip_locale> -d <reseau_local/masque_de_reseau>
-p <nom_du_protocole> --sport <nom_du_port> -m state --state ESTABLISHED -j
ACCEPT
```

> On précise que l'on accepte de laisser sortir de l'interface réseau, tout paquet dont la source est l'adresse ip correspondant à notre interface réseau, à destination de notre réseau local utilisant tel protocole, vers tel numero de port (ou service) dont l'état est 'relatif' ou 'établi' - en rapport avec la règle d'entrée.

→ Tous les paquets qui ne correspondent pas à ces critères d'entrée ou sortie n'entrent pas ou ne sortent pas !

Pour dresser des définitions avec des arguments plus précis car tenant compte des particularités de chacun des protocoles, voir la documentation **netfilter:Documentation**.

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
voir samba
3) , 4) , 5)
Pour SMTP, POP3, IMAP
6)
Une introduction en français http://varrette.gforge.uni.lu/download/polys/Tutorial_Kerberos.pdf

7)

Par défaut c'est la table filter qui est listée, donc on peut aussi utiliser la commande: `iptables -L` pour le même résultat.

8)

par exemple PREROUTING pour la table NAT