iNFO

IUT GRAND OUEST NORMANDIE

R 3.06

2025 – 2026

Architecture des réseaux

Documentation

IPTables

ANNE Jean-François

IPTables

A. Iptables

Iptables est une interface en ligne de commande permettant de configurer Netfilter. En plus de iptables, depuis la version 8.04, Ubuntu est installé avec la surcouche UFW qui permet de contrôler simplement Netfilter, UFW est toutefois moins complet que iptables.

Cette documentation est une introduction à iptables, elle est destinée à ceux qui souhaitent mettre en place un pare-feu et/ou un partage de connexion, sur une machine Linux, sans passer par une interface graphique. Seule la table par défaut (Filter) d'iptables est présentée ici et seules les chaînes utilisées par Filter (Input, Forward et Output) y sont exposées.

Les lecteurs désirant approfondir leur recherche et aborder l'utilisation des autres tables (Nat, Mangle, Row et Security) ainsi que des chaînes non utilisables par Filter (Prerouting et Postrouting) se tourneront vers les nombreuses documentations disponibles sur l'Internet (voir notamment ici). Ceux désirant configurer un pare-feu par l'intermédiaire d'une interface graphique se tourneront vers Gufw ou encore vers Shorewall pour une utilisation sur serveur.

Iptables existe aussi pour ipv6, pour cela il suffit d'utiliser la commande ip6tables au lieu de iptables.

Pour une bonne compréhension d'iptables (et des pares-feux en général) il est conseillé d'avoir des notions en réseaux informatiques, au minima connaître le principe de fonctionnement des protocoles TCP et UDP.

B. La table filter

Iptables fonctionne selon un système de tables, ces tables sont composées de chaînes. Dans le cadre de la configuration et de l'utilisation de Netfilter comme pare-feu, c'est la table **Filter** qui est utile, elle permet de filtrer les paquets réseaux. Tout paquet entrant est analysé afin de déterminer notamment sa source et sa destination.

Elle est composée de trois sortes de **chaîne** :

- **INPUT** : Permet d'analyser les paquets entrants. Si le paquet est adressé au poste, il est confronté au filtre INPUT.
- **FORWARD** : Permet d'analyser et d'autoriser les trames à passer d'une interface à une autre, seulement dans le cadre d'une interface réseau servant de passerelle.
- **OUTPUT** : Permet d'analyser les paquets sortants. Si le paquet sort du poste, il passera par la chaîne OUTPUT.

À cette table, peuvent être affectées des politiques (policy) : **DROP**, **LOG**, **ACCEPT** et **REJECT**.

C. Configuration du pare-feu

Nous allons configurer notre pare-feu de la manière suivante :

- On bloque tout le trafic entrant par défaut.
- On autorise au cas par cas : le trafic appartenant ou lié à des connexions déjà établies et le trafic à destination des serveurs (web, ssh, etc.) que nous souhaitons mettre à disposition.

Afin de ne pas avoir de problème au moment où on crée ces règles, nous allons d'abord créer les autorisations, puis nous enverrons le reste en enfer.

En tapant « `sudo iptables -L` », une liste de vos règles actuelles est affichée. Si vous (ou un logiciel) n'avez encore jamais touché à *iptables*, les chaines sont vides, et vous devriez voir :

```
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
```

Pour l'instant, tout passe dans toutes les directions (*policy ACCEPT*). Pour cette configuration basique, seul le trafic entrant (chaine *input*) nous intéresse.

Par défaut, « `sudo iptables -L` » n'affiche que la table "filter". Pour consulter les autres tables, vous devez ajouter l'option -t suivie de "nat", "mangle" ou "raw". Pour la configuration d'un pare-feu la table "filter" est toutefois la seule nécessaire.

ATTENTION, si vous avez modifié la règle par défaut pour le blocage (iptables -P INPUT DROP voir plus bas) et que vous tapez iptables -F vous bloquerez tous les accès … y compris celui en cours. Ceci est particulièrement problématique sur une machine sur laquelle vous accédez à distance (serveur etc.).

Si vous avez déjà modifié la configuration et que vous voulez la réinitialiser, tapez :

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

1°) ***Autoriser le trafic entrant d'une connexion déjà établie***

Pour permettre à une connexion déjà ouverte de recevoir du trafic :

```
# iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Si vous utilisez une ancienne version de iptables la commande ci-dessus peut ne pas fonctionner, dans ce cas utilisez celle-ci :

```
# iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

Une ancienne configuration avec l'état "–state RELATED" est toujours sur internet, or cette option peut permettre l'ouverture de port non désirée sur votre machine par un attaquant. L'option "RELATED" est à utiliser avec prudence. Pour plus d'information :

https://gist.github.com/azlux/6a70bd38bb7c525ab26efe7e3a7ea8ac

a) ***Permettre le trafic entrant sur un port spécifique***

Pour permettre le trafic entrant sur le port 22 (traditionnellement utilisé par SSH, vous devrez indiquer à iptables tout le trafic TCP sur le port 22 de votre adaptateur réseau.

```
# iptables -A INPUT -p tcp -i eth0 --dport ssh -j ACCEPT
```

Cette commande ajoute une règle (-A) à la chaine contrôlant le trafic entrant *INPUT*, pour autoriser le trafic (-j ACCEPT), vers l'interface (-i) *eth0* et à destination du port (--dport) *SSH* (on aurait pu mettre 22).

Maintenant vous pouvez vérifier vos règles iptables :

```
# iptables -L
Chain INPUT (policy ACCEPT)
target   prot opt source       destination
ACCEPT    all --  anywhere       anywhere            state RELATED,ESTABLISHED
ACCEPT   tcp --  anywhere     anywhere            tcp dpt:ssh
```

Maintenant, acceptons tout le trafic web (www) entrant :

```
# iptables -A INPUT -p tcp -i eth0 --dport 80 -j ACCEPT
```

En regardant nos règles, nous avons :

```
# iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source     destination
ACCEPT     all  --  anywhere    anywhere          state RELATED,ESTABLISHED
ACCEPT    tcp  --  anywhere    anywhere           tcp dpt:ssh
ACCEPT    tcp  --  anywhere    anywhere           tcp dpt:www
```

Nous avons exceptionnellement autorisé le trafic tcp pour ssh et les ports web, mais comme nous n'avons rien bloqué, tout le trafic passe quand même.

b) Bloquer le trafic

Maintenant que nous avons fini avec les autorisations, il faut maintenant bloquer le reste. Nous allons en fait modifier la « politique par défaut » (*policy*) de la chaine *INPUT* : cette décision (*DROP*) s'applique lorsqu'aucune règle n'a été appliquée à un paquet. Donc, si la tentative de connexion n'est permise par aucune des règles précédentes, elle sera rejetée.

```
# iptables -P INPUT DROP #warning : à ne pas utiliser sur un serveur distant !
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source       destination
ACCEPT     all  --  anywhere         anywhere          state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere     anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere     anywhere            tcp dpt:www
```

Un autre moyen de procéder est l'ajout en fin de chaine d'une règle supprimant les paquets (les paquets autorisés par les règles précédentes n'atteindraient pas celle-ci), *via* `iptables -A INPUT -j DROP`, ***mais il faudrait alors faire attention à la position des futures règles.***

c) Autoriser le trafic local

Un petit problème de notre configuration est que même l'interface locale *(loopback)* est bloquée. Nous pourrions avoir écrit les règles de rejet seulement pour *eth0* en spécifiant -i eth0, mais nous pouvons aussi ajouter une règle pour *loopback*. Par exemple, nous pourrions l'insérer en 2ème position :

```
# iptables -I INPUT 2 -i lo -j ACCEPT
```

Pour lister les règles plus en détail.

```
# iptables -L -v -n
```

d) Autoriser les requêtes ICMP (ping)

Il peut être utile de valider les réponses aux requêtes "ping", ne serait-ce que pour s'assurer que le poste est toujours en activité.

```
# On autorise le PC à faire des pings sur des IP externes et à répondre aux
requêtes "ping"
iptables -A OUTPUT -p icmp -m conntrack --ctstate NEW,ESTABLISHED,RELATED -j
ACCEPT

# Si vous utilisez une ancienne version de iptables la commande ci-dessus peut ne
pas fonctionner, dans ce cas entrez la commande suivante :
iptables -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# On autorise les pings
iptables -A INPUT -p icmp -j ACCEPT
```

e) Supprimer une règle

Si vous vous êtes trompé dans la création d'une règle et que cela vous bloque une connexion, vous pouvez supprimer une seule entrée plutôt que de tout réinitialiser.

Tout d'abord vous listez l'ensemble de vos règles avec l'affichage des lignes :

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

Ce qui me retourne par exemple:

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    DROP       icmp --  anywhere             anywhere
2    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
3    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:www
4    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:webmin

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

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

```
1    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:www
2    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:12345
```

Je souhaite supprimer la ligne 2 de la chaîne OUTPUT

Syntaxe : iptables -D chaine numéro_de_ligne

```
iptables -D OUTPUT 2
```

f) Sauvegarder vos règles

Passer en mode superutilisateur

```
sudo -s iptables-save -c
```

2°) Appliquer les règles au démarrage

Vous avez testé vos règles, ça marche au poil, alors il reste à les appliquer au démarrage.

* Depuis au moins Ubuntu 12.04, le paquet **iptables-persistent** gère les règles au démarrage. Il propose de sauvegarder les règles dans le dossier **/etc/iptables**, fichier **rules.v4** pour les règles IPv4 et **rules.v6** pour les règles IPv6. Le script peut s'appeler via :

```
service iptables-persistent
```

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

* Si vous avez une distribution plus ancienne.

Commencez par éditer un fichier en root, que vous enregistrerez sous **/etc/init.d/monIptables** (variable selon la distribution utilisée). La première ligne de ce fichier doit être :

```
#!/bin/bash
```

Cette ligne indique que le fichier doit être enregistré en tant que script bash.

Le reste du fichier doit contenir les commandes *iptables* que vous avez générées.

Déplacez le script iptables dans /etc/init.d

```
sudo mv /emplacement/du/script/iptables /etc/init.d
```

Rendez ce script exécutable :

```
sudo chmod +x /etc/init.d/monIptables
```

Pour indiquer à votre ordinateur de l'utiliser au démarrage:

```
sudo update-rc.d monIptables defaults
```

Ça devrait être bon. Au prochain redémarrage, vous pouvez vérifier que vos règles sont bien utilisées, en effectuant :

```
sudo iptables -L
```

3°) Script iptables

Ce script est un exemple, il est à adapter à vos besoins. Il peut toutefois être utilisé pour une utilisation courante, il offre une plutôt bonne "protection" pour un usage Desktop.

iptables :

```
#!/bin/bash

iptables-restore < /etc/iptables.test.rules

## Script iptables by BeAvEr.

## Règles iptables.

## On flush iptables.

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

```
iptables -t mangle -X

## On drop les requêtes ICMP (votre machine ne répondra plus aux requêtes ping
sur votre réseau local).

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j DROP

## On récupère notre adresse internet. À utiliser seulement si vous êtes derrière
un réseau local.

export ip=$(/sbin/ip addr | grep 'state UP' -A2 | tail -n1 | awk '{print $2}' |
cut -f1  -d'/')

## Allow Samba.

iptables -t raw -A OUTPUT -p udp -m udp --dport 137 -j CT --helper netbios-ns

## Allow Avahi-daemon (seulement notre LAN. Si vous n'avez pas de LAN, supprimer
la partie --source $ip/24).

iptables -A INPUT -p udp -m udp --source $ip/24  --dport 5353 -j ACCEPT

iptables -A INPUT -p udp -m udp --source $ip/24  --dport 427 -j ACCEPT

## On accepte le Multicast.

iptables -A INPUT -m pkttype --pkt-type multicast -j ACCEPT

## On drop tout le trafic entrant.

iptables -P INPUT DROP

## On drop tout le trafic sortant.

iptables -P OUTPUT DROP

## On drop le forward.

iptables -P FORWARD DROP

## On drop les scans XMAS et NULL.

iptables -A INPUT -p tcp --tcp-flags FIN,URG,PSH FIN,URG,PSH -j DROP

iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

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

## Dropper silencieusement tous les paquets broadcastés.

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

## Permettre à une connexion ouverte de recevoir du trafic en entrée.

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

## Permettre à une connexion ouverte de recevoir du trafic en sortie.

iptables -A OUTPUT -m conntrack ! --ctstate INVALID -j ACCEPT

## On accepte la boucle locale en entrée.

iptables -I INPUT -i lo -j ACCEPT
```

```
## On log les paquets en entrée.

iptables -A INPUT -j LOG

## On log les paquets forward.

iptables -A FORWARD -j LOG

exit 0
```

4°) ***Documentations supplémentaires***

a) ***En anglais :***

- *Firewall Builder : Un outil très performant pour configurer iptables (entre autres)*
- *How To Iptables*
- *Documentation Multilingue de Netfilter et Iptables*
- *Rusty's Remarkably Unreliable Guides*

b) ***En Français :***

- *Présentation des différentes tables et chaînes d'Iptables*
- *Bible française pour apprendre les bases de la sécurité et d'Iptables sous Linux*
- *Bible française d'introduction aux réseaux et à Internet ou ici (même auteur et contenu trés proche)*
- *Tutoriel de Alexis Delatre (avec fichier iptables tout fait)*
- *"Mémoire Grise Libérée" : IpTables HowTo*
- *iptables-tutorial de Oskar Andreasson traducteur Marc Blanc et publié par Philippe Latu*
- *Supprimer une règle précise sous Iptables sur IT-Connect.fr*

5°) ***Sources***

- Basé sur https://wiki.ubuntu.com/IptablesHowTo
- Merci à Rusty Russell et son How-To, il est la base de cette page.
- Et merci surtout pour son travail au sein de l'équipe de développement de Netfilter.

Contributeurs : Kmeleon, eks,BeAvEr (Création du script iptables et modification majeure de la documentation), maverick62, mydjey (mise à jour et refonte).