Netfilter & Iptables

Master 2 I2L

Année 2009/2010

D'après la version de Jean-Christophe Soulié (2007-2008)

D. Duvivier
LIL – Université du Littoral Côte d'Opale
duvivier@lil.univ-littoral.fr

1 Vérification

Vérifier que vous avez bien la commande `iptable` avec l'une des commandes suivantes :

```
which iptables
whereis iptables
apropos iptables
dpkg –l iptables
dpkg-query --search iptables | grep '/iptables$'
aptitude search iptables
aptitude show iptables
…
```

Si ce n'est pas installé, utilisez par exemple la commande suivante :

```
aptitude install iptables
```

Premier contact avec iptables :

```
bench:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt  source               destination

Chain FORWARD (policy ACCEPT)
Target     prot opt  source               destination

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

Des erreurs peuvent se produire au lancement de la commande si vous utilisez une ancienne version du noyau Linux ou si ce dernier a été mal configuré (D. Duvivier : la suite est reprise intégralement sans mise à jour pour 2009/2010 ☹). Sinon, il faut recompiler le noyau en mettant les options suivantes :

- Enable loadable module support (CONFIG_MODULES) : Y
- Packet socket (CONFIG_PACKET) : Y
- Network packet filtering (replaces ipchains) (CONFIG_NETFILTER) : Y
- Unix domain sockets (CONFIG_UNIX) : Y
- TCP/IP networking (CONFIG_INET) : Y
- IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) : Y
- Connection tracking (required for masq/NAT) (CONFIG_IP_NF_CONNTRACK) : M
- FTP protocol support (CONFIG_IP_NF_FTP) : M
- IRC protocol support (CONFIG_IP_NF_IRC) : M
- IP tables support (required for filtering/masq/NAT) (CONFIG_IP_NF_IPTABLES) : M
- Packet filtering (CONFIG_IP_NF_FILTER) : M
- Full NAT (CONFIG_IP_NF_NAT) : M
- Limit match support (CONFIG_IP_NF_MATCH_LIMIT) : M
- Connection state match support (CONFIG_IP_NF_MATCH_STATE) : M
- MASQUERADE target support (CONFIG_IP_NF_TARGET_MASQUERADE) : M
- REDIRECT target support (CONFIG_IP_NF_TARGET_REDIRECT) : M
- LOG target support (CONFIG_IP_NF_TARGET_LOG) : M
- ULOG target support (CONFIG_IP_NF_TARGET_ULOG) : M

Pour recompiler le noyau, voyez votre merveilleux TP de SRS/Kernel ☺ !

2 Commandes

Voici les options les plus intéressantes de la commande « iptables » :

Option	Paramètre	Option	Explication	Exemple
-t (table)	"filter", "nat", "mangle", "raw"	Oui	Indique sur quelle table nous travaillons. Si aucun paramètre n'est fourni, c'est la table filter qui est sélectionnée par défaut	"iptables -t nat ..." permet de travailler sur la table "NAT"
-F (Flush)	"filter", "nat", "mangle", "raw"	Oui	Supprime toutes les règles d'une chaîne prédéfinie (tel POSTROUTING, INPUT, OUTPUT, FORWARD et PREROUTING). Si aucun paramètre n'est donné, toutes les chaînes prédéfinies sont supprimées	"iptables -F" supprime toutes les chaînes prédéfinies de la table "filter"
-X (eXclude)	[Chaîne utilisateur]	Oui	Supprime une chaîne utilisateur. S'il n'y a aucun paramètre, toutes les chaînes utilisateurs sont supprimées	"iptables -X perso_3" supprime la chaîne utilisateur "perso_3"
-P (Policy)	"filter", "nat", "mangle", "raw"	Non	Définit la politique (ou cible) par défaut d'une chaîne. Seules les chaînes prédéfinies peuvent avoir un comportement par défaut. Cette cible ne sera appliquée qu'après l'exécution de la dernière règle de la chaîne	"iptables -P INPUT DROP" supprime par défaut toutes les trames dans la chaîne "INPUT". Si aucune règle plus "souple" n'est définie, aucun paquet réseau n'arrive aux processus utilisateurs de notre machine ▪ efficace mais peu fonctionnel
-N (New)	[Chaîne utilisateur]	Non	Cette option crée une nouvelle chaîne utilisateur. Par la suite, des règles doivent êtres créées, afin de remplir cette chaîne	"iptables -N LogAndDrop" crée une chaîne utilisateur dont le nom laisse supposer que l'on va logger les paquets, puis les supprimer
-A (Append)	[Chaîne]	Non	Ajoute une ou plusieurs règle(s) à une chaîne prédéfinie ou utilisateur. La règle est ajoutée en fin de chaîne. Nous allons utiliser majoritairement cette commande	"iptables -A INPUT ..." ajoute une règle à la table des paquets entrant dans l'espace des programmes
-I (Insert)	[Chaîne] Numéro de règle]	Non	Insère une ou plusieurs règle(s) à une position passée en argument dans une chaîne	"iptables -I INPUT 1 ..." ajoute une règle à la table des paquets entrant dans l'espace des programmes en position 1 (i.e. en tête de chaîne)
-D (Delete)	[Chaîne] [Numéro de règle]	Non	Supprime une règle dans une chaîne particulière. Le numéro de la règle peut être retrouvé avec la commande "iptables -L" ou "iptables -L -n"	"iptables -D OUTPUT 1 -t mangle" supprime la 1ère règle de la chaîne "OUTPUT" de la table "Mangle"
-L (Liste)	[Chaîne]	Oui	Liste les règles pour la chaîne indiquée. Ou, si aucune chaîne n'est indiquée, affiche les règles pour toutes les chaînes de la table indiquée. Note : Rajoutées à "-L" les options "-n" et "-v" sont très utiles	"iptables -L -n -v" affiche le maximum d'informations sur les règles de la table "filter"
-j (jump)	[Cible]	Non	Définit l'action à prendre si un paquet répond aux critères de cette règle. Les principales valeurs sont : ACCEPT, DROP, REJECT, LOG	"iptables -A OUPUT -j DROP" supprime tous les paquets sortant des processus locaux
-i (input)	[Interface réseau]	Non	Critère sur l'interface réseau dont provient le paquet	"iptables –A INPUT -i eth0 ..." filtre les paquets arrivant aux programmes et venant de l'interface réseau eth0
-o (output)	[Interface réseau]	Non	Critère sur l'interface réseau par où les paquets vont sortir	"iptables –A OUTPUT -o ppp0" filtre les paquets créés par les programmes et sortant sur Internet
-s (source)	[!] [Adresse IP ou réseau]	Non	Critère sur l'adresse IP source du paquet	"iptables -t nat -A PREROUTING -i 192.168.0.0/24 ..." travaille sur le routage des paquets du réseau interne network.net
-d (destination)	[!] [Adresse IP ou réseau]	Non	Critère sur l'adresse IP de destination du paquet	"iptables -A OUTPUT -d 192.168.0.0/24 ..." filtre les paquets sortant de l'espace utilisateur, à destination du réseau network.net
-p (protocole)	[!] "all", "tcp", "udp", "icmp", ou un numéro	Non	Critère sur le type de trames utilisé dans le paquet. D'autres numéros de protocoles peuvent être utilisés. Voir votre fichier "/etc/procoles"	"iptables -A INPUT -p udp ..." filtre uniquement les paquets de type UDP.
--sport	[!] [Port ou service]	Non	Critère sur le port source des paquets IP	"iptables -A INPUT -sport 80 ... " filtre tout ce qui vient du port HTTP (80)
--dport	[!] [Port ou service]	Non	Critère sur le port de destination des paquets IP	"iptables -O OUPUT -dport ! 21 ... " filtre tout ce qui n'est pas destiné à FTP (21)
-m (module)	[Nom d'un module]	Non	Demande d'utiliser un module particulier	"iptables ... -m state ..." utilise le module de suivi de connexion
--state	[!] "NEW", "ESTABLISHED", "RELATED", "INVALID"	Non	Cette option ne s'utilise que cumulée avec l'option "-m state", afin d'être utilisée pour le suivi de connexion	"iptables ... -state NEW,ESTABLISHED ..." filtre les paquets de nouvelles connexions, ou de connexions déjà établies via une "poignée de main"
--log-prefix --ulog-prefix	[Un mot]	Non	Rajoute un commentaire pour les cibles LOG et ULOG	"iptables ... -j LOG --log-prefix toto" rajoutera le mot "toto" au log de cette règle

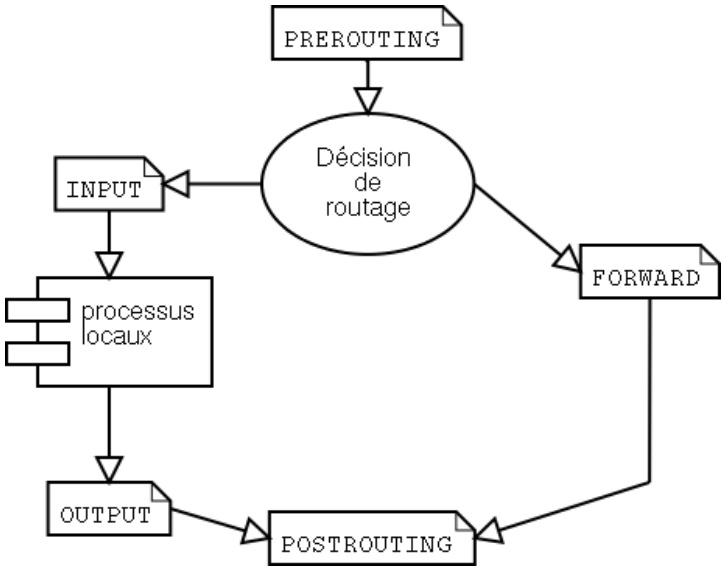
La commande « `iptables` » possède de nombreux paramètres, vous pouvez vérifier par vous-même :

```
man iptables
info iptables
→ Allez jeter un œil du côté du répertoire /usr/share/doc/iptables/…
```

Avec iptables/netfilter, il est possible de :

- créer un firewall filtrant basé sur les paquets mais aussi sur le statut des connexions engendrés par les paquets (suivi de connexion) ;
- utiliser NAT(Network Address Translation) et le masquerading afin de partager un accès internet à plusieurs machines ;
- utiliser NAT pour faire du proxy transparent (évite d'avoir à paramétrer le proxy sur les clients/navigateurs web) ;
- mettre en place (notamment grace au marquage des paquets via la table mangle) la possibilité d'obtenir un routeur sophistiqué permettant le QoS (Quality of Service, i.e. privilégier certains services, mettre en place des limites d'utilisation de bande passante sur un utilisateur, sur un groupe etc.) ;
- manipuler des paquets pour par exemple altérer un champ d'un datagramme IP

Lorsqu'un paquet arrive, il est orienté (en fonction d'un ensemble de paramètres) dans l'une des différentes chaînes disponibles.



Un paquet entre dans la machine via la chaîne PREROUTING et sort de la machine via la chaîne POSTROUTING (chaînes servant notamment à certaines opérations de routage entre réseaux).

Les chaînes INPUT et OUTPUT servent respectivement à ajouter des règles pour les paquets destinés à la machine et ceux émis par la machine ; un paquet destiné à ma machine arrive dans la chaîne INPUT.

Par exemple, pour visualiser une page web depuis la machine, j'émets une requête qui sort par la chaîne OUTPUT et la réponse arrive sur ma machine par la chaîne INPUT .

Au moment où le paquet entre dans la chaîne, les règles « constituant » cette chaîne sont appliquées dans l'ordre dans lequel elles sont stockées.

3 Outils complémentaires

iptstate → consultez la documentation
netstat-nat → consultez la documentation
aptitude install nmap knmap nmapsi4 → consultez la documentation

Consultez de la documentation sur Internet et/ou jetez un œil sur /etc/services pour vous [re]mettre en mémoire les ports des principaux services à contrôler. Pour une configuration classique, nous obtenons la liste (non exhaustive) suivante :

- 20 et 21 pour ftp
- 22 pour ssh
- 25 pour smtp
- 53 pour DNS
- 67 et 68 pour DHCP
- 80 pour http
- 109 pour pop2
- 110 pour pop3
- 115 pour sftp
- 143 pour imap2
- 443 pour https
- 631 pour CUPS
- 3128 (et éventuellement 3130 si plusieurs proxy/caches communiquent) pour les proxy
- …

4 Iptables basique

Nous allons commencer par travailler exclusivement sur la table `filter`.

Bon, petit rappel sur LA règle de filtrage universellement reconnue :

- **Premièrement**, vider toutes les chaînes de toutes les tables de Netfilter, afin de savoir exactement ce que l'on a dans notre firewall. Mais il ne faudra pas rester trop longtemps dans cette situation, car la machine sera sans aucune protection.
- **Deuxièmement**, interdire par défaut tous les paquets. Pour cela, nous allons utiliser l'option `"-P"` ("Politique par défaut") des chaînes `INPUT`, `FORWARD` et `OUTPUT` de la table `filter`.
- **Dans un dernier temps**, nous n'allons autoriser que certains flux bien particuliers.
- **Concernant l'ordre des règles :** Lorsque nous appelons la commande `"iptables"` pour ajouter/supprimer des règles, l'ordre d'insertion de celles-ci a une certaine importance. Si une première règle supprime un certain type de paquets, une seconde règle écrite un peu plus loin dans votre script ne verra jamais ces paquets. Donc inutile de les accepter ou de les supprimer de nouveau. Mais en général, comme on écrit uniquement des règles pour accepter des paquets (`"-j ACCEPT"`), il n'y pas d'importance dans l'ordre des règles.
- **Suppression de toutes les chaînes :** C'est facile, il faut supprimer les tables prédéfinies (option `"-F"`) et toutes les tables utilisateurs (option `"-X"`). Que nous en ayons déjà ou pas écrite n'a aucune importance car nous supprimons tout (règle via –F et chaînes utilisateur via –X) :

```
bench:~# iptables -t filter -F
bench:~# iptables -t filter -X
```

- **Définition de la politique (cibles) par défaut :** La table `filter` possède 3 chaînes, donc elles sont toutes les trois à initialiser. Par défaut, on décide donc de détruire tous les paquets (`DROP`) :

```
bench:~# iptables -t filter -P INPUT DROP
bench:~# iptables -t filter -P OUTPUT DROP
bench:~# iptables -t filter -P FORWARD DROP
```

Pour voir le résultat, il suffit de taper :

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

Chain FORWARD (policy DROP)
Target     prot opt   source               destination

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

A ce stade, nous avons court-circuité tout le système de réseau. Toutes nos connexions réseaux sont hors service. Aucun logiciel ne peut accéder au réseau ou à ses propres serveurs.

Le réseau sur notre machine est complètement hors service !

Même la commande "`ping localhost`" ou "`ping 127.0.0.1`" ne fonctionne plus !!!

```
bench:~# ping localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
--- localhost. localdomain ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2011ms
```

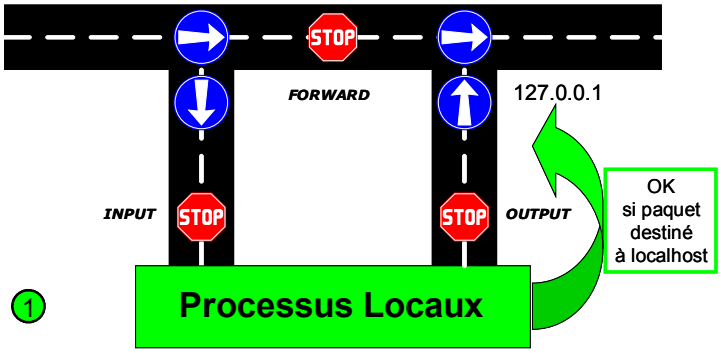
Mais cela tombe bien, car c'est exactement ce que nous voulions faire ! D'un autre coté, c'est la protection absolue du réseau, qui rivalise presque avec la déconnexion physique des câbles…

Nous allons pouvoir maintenant reconstruire des autorisations au fur et à mesure.

`localhost` (réseau virtuel local) : C'est le plus facile. Nous pouvons avoir toute confiance en ce réseau, car il est interne à la mémoire de notre machine.

(1) ➔ Nous allons donc autoriser ("`-j ACCEPT`") toutes les connexions sortantes des processus locaux ("`-A OUTPUT`") par cette interface virtuelle ("`-d localhost`") :

```
bench:~# iptables -t filter -A OUTPUT -d localhost -j ACCEPT
```



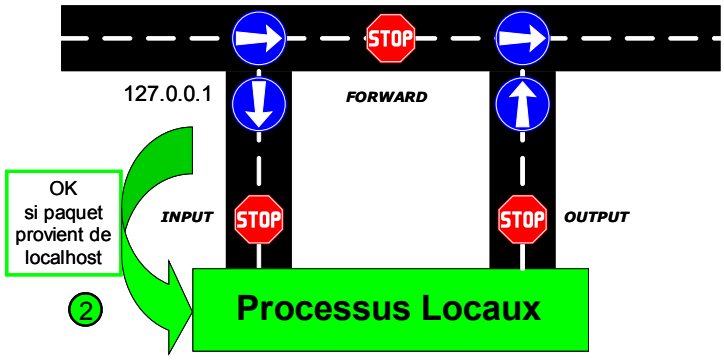
Cette fois les paquets peuvent sortir à destination de localhost (mais pas encore entrer en provenance de 127.0.0.1), il n'y a plus d'erreur (operation not permitted) mais le ping reste figé et doit être interrompu par un « Ctrl C » car les paquets ne sont pas autorisés à entrer :

```
bench:~# ping localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
^C
--- localhost. localdomain ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2011ms
```

A noter que si je tente avec une autre adresse IP, y compris l'adresse de la machine (par exemple 192.168.0.13), j'ai toujours l'erreur en boucle « Operation not permitted » et c'est normal car seul les paquets à destination de 127.0.0.1 sont autorisés pour l'instant !

(2) ➔ Nous allons faire l'inverse, c'est à dire autoriser (`"-j ACCEPT"`) toutes les connexions entrantes dans les processus locaux (`"-A INPUT"`) :

```
bench:~# iptables -t filter -A INPUT  -s localhost -j ACCEPT
```



Note : dans le schéma ci-dessus je ne fais apparaître que la nouvelle règle, mais en réalité les règles représentées par les schémas **(1)** et **(2)** sont actives.

Cette fois la commande « ping 127.0.0.1 » fonctionne de nouveau, mais « ping 192.168.0.13 » génère toujours une erreur « Operation not permitted » en boucle et c'est normal.

Avec `iptables -L`, on obtient quelque chose dans le style :

```
bench:~# iptables -L
Chain INPUT (policy DROP)
target     prot  opt   source               destination
ACCEPT     all   --    loalhost             anywhere

Chain FORWARD (policy DROP)
Target     prot  opt   source               destination

Chain OUTPUT (policy DROP)
Target     prot  opt   source               destination
ACCEPT     all   --    anywhere             localhost
```

Note : si vous n'obtenez pas cela, vous avez probablement inversé les options « -d » et « -s » ou « INPUT » et « OUTPUT ». Il faut tout recommencer (depuis les commandes iptables -t filter -F; iptables -t filter -X).

La commande `"ping localhost" fonctionne correctement`, mais pas possible de faire un ping sur une adresse externe (www.google.fr par exemple).

Normal, … Pourquoi ? Concernant une adresse externe comme « www.google.fr », les paquets qui vont et viennent vers le DNS sont filtrés ! Il nous faut donc mettre en place une règle permettant de dialoguer avec le DNS. On va donc autoriser notre machine à communiquer via le port 53 qui est le port du DNS (*cf.* /etc/services pour vérifier) :

```
bench:~# iptables -t filter -A OUTPUT -p UDP --dport 53 -j ACCEPT
bench:~# iptables -t filter -A INPUT -p UDP --sport 53 -m state --state
RELATED,ESTABLISHED -j ACCEPT
```

ATTENTION : pas d'espace dans la liste « RELATED,ESTABLISHED »

Et là, normalement, ça doit être bon ! Donc, on retente :

```
bench:~# ping www.google.fr
PING www.google.fr (209.85.135.103) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
--- www.google.fr ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3010ms
```

Ca ne marche pas encore… De plus, la commande "ping localhost" fonctionne correctement, mais toujours pas possible de faire un ping sur une adresse locale (192.168.0.13). Là encore, c'est normal, les trames ping (icmp) sont filtrées (je vous rappelle que la politique par défaut est de tout mettre à la poubelle !). Il nous faut donc autoriser le ping sur notre machine.

```
bench:~# iptables -t filter -A INPUT -p icmp -j ACCEPT
bench:~# iptables -t filter -A OUTPUT -p icmp -j ACCEPT
```

Et donc, maintenant, on doit être bon !

```
bench:~# ping www.google.fr
PING www.google.fr (209.85.135.103) 56(84) bytes of data.
64 bytes from 209.85.135.103: icmp_seq=1 ttl=64 time=0.479 ms

--- www.google.fr ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.479/0.479/0.479/0.000 ms
```

Super, ça marche ! Faisons le point ; vérifions la configuration en utilisant la commande « iptables -L » :

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  localhost            anywhere
ACCEPT     udp  --  anywhere              anywhere            udp spt:domain
state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             localhost
ACCEPT     udp  --  anywhere             anywhere            udp dpt:domain
ACCEPT     icmp --  anywhere             anywhere
```

Il est possible d'afficher les détails de la configuration en utilisant la commande « iptables -L -n -v » :

```
Chain INPUT (policy DROP 160 packets, 44554 bytes)
 pkts bytes target     prot opt in     out     source               destination
   92 10762 ACCEPT     all  --  *      *       127.0.0.1            0.0.0.0/0
  112 33121 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp spt:53 state
RELATED,ESTABLISHED
    5   420 ACCEPT     icmp --  *      *       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 490 packets, 36875 bytes)
 pkts bytes target     prot opt in     out     source               destination
   92 10762 ACCEPT     all  --  *      *       0.0.0.0/0            127.0.0.1
  220 15645 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    8   879 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
```

Ouvrons les connections HTTP (port 80) et HTTPS (port 443) avec :

```
bench:~# iptables -t filter -A OUTPUT -p TCP --dport 80 -j ACCEPT
bench:~# iptables -t filter -A INPUT -p TCP --sport 80 -m state --state RELATED,ESTABLISHED -j ACCEPT
bench:~# iptables -t filter -A OUTPUT -p TCP --dport 443 -j ACCEPT
bench:~# iptables -t filter -A INPUT -p TCP --sport 443 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

On peut tester maintenant avec la commande nmap (attention à cette commande quand même ☹) :

```
bench:~# nmap www-lil.univ-littoral.fr -p 80,443

Starting Nmap 5.00 ( http://nmap.org ) at 2009-10-07 01:42 CEST
Interesting ports on lil.univ-littoral.fr (193.49.192.193):
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.467 seconds
```

Voilà c'est fait ! Enfin nous allons loguer dans le fichier « /var/log/debug » tous les paquets que nous allons bloquer avant de les bloquer :

```
bench:~# iptables -t filter -A INPUT -i eth0 -j LOG --log-prefix "Iptables INPUT dropped : " --log-level debug
bench:~# iptables -t filter -A OUTPUT -o eth0 -j LOG --log-prefix "Iptables OUPUT dropped : " --log-level debug
```

Maintenant voyons quelques commandes utiles…

Sauver votre configuration :

```
bench:~# iptables-save > /root/iptables-sauvegarde
```

Restaurer votre configuration :

```
bench:~# iptables-restore /root/iptables-sauvegarde
```

Voir votre configuration complète :

```
bench:~# iptables -L -v --line-numbers
```

Vérifier le log en temps réel (paquets rejetés) : (tapez « [CTRL + C] » pour sortir)

```
bench:~# tail -f /var/log/debug
```

Au prochain redémarrage vos règles « `iptables` » seront perdues, il faut donc automatiser la mise en place de ces règles. Il est possible d'utiliser le paquet « iptables-persistent », mais c'est tellement plus drôle de le faire à la main ☺ ! Allez dans « `/etc/network` » et créez le fichier « `iptables` » :

```
iptables -t filter -F
iptables -t filter -X
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
iptables -t filter -P FORWARD DROP
iptables -t filter -A OUTPUT -d localhost -j ACCEPT
iptables -t filter -A INPUT  -s localhost -j ACCEPT
iptables -t filter -A OUTPUT -p UDP --dport 53 -j ACCEPT
iptables -t filter -A INPUT -p UDP --sport 53 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -p icmp -j ACCEPT
iptables -t filter -A OUTPUT -p icmp -j ACCEPT
iptables -t filter -A OUTPUT -p TCP --dport 80 -j ACCEPT
iptables -t filter -A INPUT -p TCP --sport 80 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t filter -A OUTPUT -p TCP --dport 443 -j ACCEPT
iptables -t filter -A INPUT -p TCP --sport 443 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -i eth0 -j LOG --log-prefix "Iptables INPUT dropped : " --log-level debug
iptables -t filter -A OUTPUT -o eth0 -j LOG --log-prefix "Iptables OUPUT dropped : " --log-level debug
```

Créez le fichier de démarrage à l'aide des commandes suivantes ou basez-vous sur /etc/init.d/skeleton :

```
bench:~# cd /etc/init.d/
bench:~# emacs iptables-conf
```

Entrez ceci :

```
#! /bin/sh
set -e

iptables_start() {
          if [ -f /etc/network/iptables ]; then
                    . /etc/network/iptables
          fi
}

iptables_stop() {
          iptables -t filter -F INPUT
          iptables -t filter -F OUTPUT
          iptables -t filter -P INPUT ACCEPT
          iptables -t filter -P OUTPUT ACCEPT
}

case "$1" in
  start)
          echo -n "Apply Iptables configuration"
          iptables_start
          echo "."
          ;;
  stop)
          echo -n "Clear Iptables configuration"
          iptables_stop
          echo "."
          ;;
  restart)
          echo -n "Reloading Iptables configuration"
          iptables_stop
          iptables_start
          echo "."
          ;;
  *)
          echo "Usage: /etc/init.d/iptables-conf {start|stop|restart}"
          exit 1
esac

exit 0
```

Donnons les bons droits à notre fichier :

```
bench:~# chmod +x iptables-conf
```

Enregistrez et quittez. Rajoutez le script de démarrage dans le mode voulu avec la priorité voulue :

```
bench:~# update-rc.d iptables-conf start 99 2 3 4 5 . stop 20 0 1 6 .
```

Maintenant vous pouvez jouer avec les « start » et les « stop » sur le fichier de configuration.

Bien voilà pour les bases… Nous pourrions y rester des heures et des heures…

5 Quelques exercices

5.1 Opérations sur une seule chaîne et sur la table filter

Créer les règles suivantes :

1. Interdire tout paquet entrant, effacer la règle → ✎ *Paramètre protocole*
2. Interdire le protocole ICMP entrant, effacer la règle → ✎ *Paramètre source*
3. Interdire le protocole ICMP provenant de localhost, effacer la règle
   → ✎ *Chaîne OUTPUT paramètre destination*
4. Interdire tout paquet sortant à destination de localhost, effacer la règle → ✎ *Paramètre inversion*
5. Interdire un paquet s'il ne provient pas de localhost, effacer la règle ✎ *Paramètre interface d'entrée*
6. Interdire tout paquet entrant s'il provient de lo (à ne surtout jamais faire sur une machine si l'on ne sait pas EXACTEMENT ce que l'on fait), effacer la règle → ✎ *Paramètre interface de sortie*
7. Interdire tout paquet sortant par eth0, effacer la règle → ✎ *Paramètre destination port*
8. Interdire tout paquet sortant à destination du port ftp, effacer la règle → ✎ *Paramètre source port*
9. Interdire tout paquet sortant par eth0 dont le N° de port destination est < à 1025 (Les services associés aux ports < 1025 tournent sous le compte root, il sont donc à protéger prioritairement), effacer la règle
   → ✎ *Paramètre flag TCP*
10. Interdire toute tentative d'initialisation de connexion TCP provenant de eth0, effacer la règle
    → ✎ *Paramètre flag icmp*
11. Interdire tout paquet entrant correspondant à un ping, effacer la règle
12. Interdire toute réponse à un ping, effacer la règle → ✎ *Paramètre extension:*

- extension mac : attention on ne peut l'utiliser que sur les tables INPUT, PREROUTING et FORWARD

13. Interdire tout paquet entrant par eth0 dont l'adresse mac n'est pas celle du voisin, effacer la règle

- extension limit

14. Positionner la police par défaut à DROP pour la chaîne INPUT, écrire une règle qui laisse entrer 5 tentatives de connexion TCP puis qui n'en laisse plus passer que 2 par minute, faire de même avec les pings, on suppose que le burst a été consommé, combien de temps (sans tentative de connexion ou d'echo-request) faudra-t'il pour qu'on puisse à nouveau avoir 5 de ces paquets qui passent à la suite ? Effacer la règle.
    → ✎ *Le suivi de connexion (ip_conntrack)*
15. Positionnez les règles par défaut à DROP pour les chaînes INPUT, OUTPUT, FORWARD, autoriser tout paquet relatif à une connexion déjà établie ou en rapport avec une connexion déjà établie en entrée, interdire tout paquet relatif à une connexion de type INVALID, autoriser tout paquet créant une nouvelle connexion en sortie à destination du port 80, que faut-il modifier ici pour que l'on puisse naviguer sur le net ? Effacer la règle

5.2 Opérations sur plusieurs chaînes et sur la table filter

16. Créer une nouvelle chaîne qui logue les paquets entrants en ajoutant le préfixe [INPUT DROP] et qui les drop
17. Renvoyer sur cette nouvelle chaîne tout paquet engendrant une nouvelle connexion en entrée