Sécurité
TP Firewalls – iptables

Ce sujet a été initialement rédigé par Carlos Aguilar, Éric Asselin et Joris Barrier.

Objectifs

L'objectif de ces séances est de se familiariser avec les principes et la configuration de firewalls dans une architecture réseau.

À lire si vous travaillez sur machines virtuelles. Lancez immédiatement le téléchargement.

```
$ cd
$ wget -c 'http://flash.enseeiht.fr/bemorgan/debian-9.5.0-firewall.tgz'
$ tar xvf debian-9.5.0-firewall.tgz
$ cd debian-9.5.0-firewall
```

Éditez `start.sh`. Remplacez `sakura` par `lxterminal` et remplacez l'option `-x` utilisée plus tard dans le script par `-e`.

Autorisez l'association des interfaces `br[012]` avec vos machines virtuelles : ajouter les bridge br[012] à `/etc/qemu/bridge.conf`.

```
allow br0
allow br1
allow br2
```

Démarrez les machines virtuelles.

```
$ sudo ./start.sh # le script de création des tap n'est pas suid sous Debian
```

1 Introduction

"Iptables est utilisé pour mettre en place, maintenir, et inspecter les tables des règles de filtrage des paquets IP du noyau Linux. Plusieurs tables différentes peuvent être définies. Chaque table contient un nombre de chaînes pré-définies, et peut aussi contenir des chaînes définies par l'utilisateur.

Une chaîne est une liste de règles auxquelles peuvent correspondre un ensemble de paquets. Chaque règle spécifie ce qui doit être fait avec un paquet qui correspond. Cela s'appelle une «cible», qui peut être un saut vers une chaîne définie par l'utilisateur dans la même table"

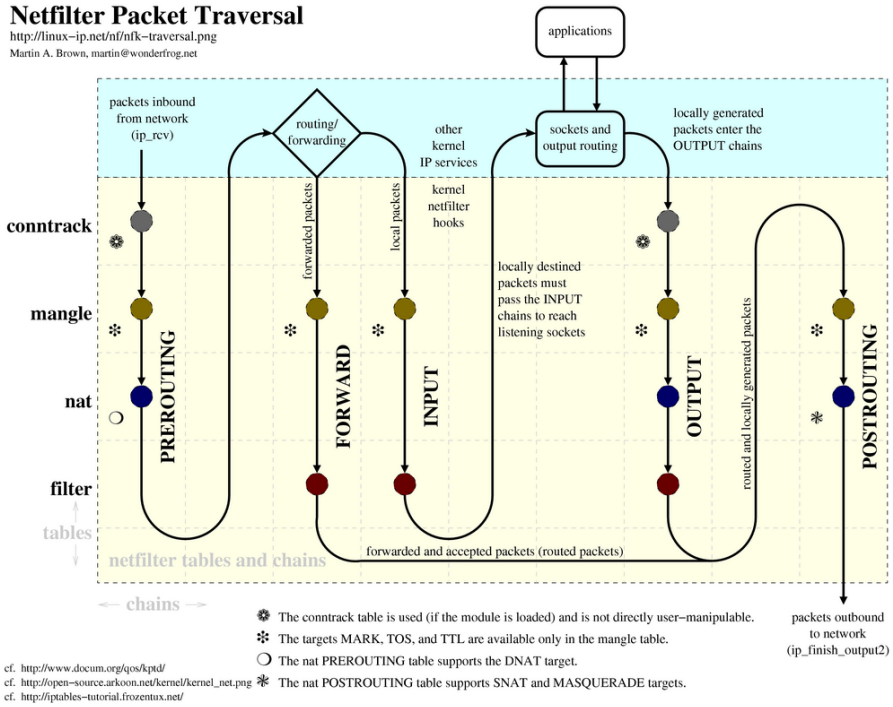
man iptables

Comme son nom l'indique, iptables applique des règles que l'on insère dans des tables. La table par défaut est appelée ***filter*** et elle contient les règles de filtrage. Deux autres tables, ***nat*** et ***mangle***, sont destinées à contenir les règles qui mèneront à des modifications sur les paquets : pour la première ce seront des modifications de type NAT, comme son nom l'indique ; et pour la deuxième toute autre modification. Il existe deux autres tables, ***raw*** et ***security***, mais leur usage est relativement avancé et on les ignorera dans ce TP.

Quand on crée une règle pour iptables, il faut préciser où l'insérer et à quelle moment elle doit être appliquée. Pour ce faire, il faut préciser la table (si on ne la précise pas, ***filter*** est utilisée par défaut) et une chaîne. La chaîne détermine le moment d'application de la règle. Il y a cinq chaînes par défaut dans iptables : **PREROUTING**, qui concerne les paquets entrants avant les décisions de routage ; **FORWARD**, qui concerne les paquets qui devraient être routés ; INPUT, qui concerne les paquets destinés aux sockets locaux ; **OUTPUT**, qui concerne les paquets générés localement ; et **POSTROUTING** qui concerne les paquets sur le point de partir (qu'ils soient générés localement ou routés).

Le schéma ci-dessous suivante montre comment les paquets sont traités. La table ***conntrack*** contient les règles qui déterminent si plusieurs paquets correspondent à une même connexion ou pas, enregistrent quelles connexions sont établies, etc. Cette table n'est pas modifiable et on l'ignorera tout au long du TP.

Comme on peut le voir dans ce schéma, chaque chaîne n'est accessible que par certaines tables. Par exemple pour appliquer une règle avant le routage, il faudra l'inclure dans la chaîne PREROUTING des tables



mangle ou ***nat***. Ces contraintes correspondent à la logique des tables. Il est normal de modifier un paquet avant les décisions de routage (par exemple pour faire du NAT) mais il est difficile de prendre une décision de filtrage avant la décision de routage.

Principe des chaînes

Quand un paquet passe par une chaîne d'une table donnée dans le traitement, chaque règle de cette chaîne est étudiée. Si le paquet ne correspond pas la règle suivante est examinée. S'il correspond, la suite dépend de la cible de cette règle mais en général les règles suivantes ne sont pas traitées. Comme la citation du manuel en début de TP indique, on appelle cible d'une règle l'action à réaliser si la règle est vérifiée.

Cibles	Description	Options
ACCEPT	Laisse passer le paquet. Passage à la chaîne suivante.	
DROP	Élimine le paquet. Fin de traitement	
REJECT	Élimine le paquet mais contrairement à DROP, envoie un message à l'émetteur en indiquant la raison. Fin de traitement.	--reject-with raison raison pouvant être : icmp-port-unreachable (defaut) tcp-reset ...

D'autres cibles sont possibles, comme par exemple DNAT, SNAT et MASQUERADE pour faire de la traduction d'adresse, ou LOG pour faire de la journalisation. Mais contrairement à ACCEPT, REJECT, et DROP elles n'entraînent pas l'arrêt du traitement de la chaîne.

2 Configuration réseau

À l'aide de trois machines, réalisez le schéma de la figure 1 qui servira tout au long des exercices. Testez que vous pouvez communiquer entre les différents réseaux et avec le routeur en utilisant ping et en faisant diverses connections ssh.

À lire si vous travaillez sur machines virtuelles. Les machines virtuelles proposées pour réaliser le TP configurent déjà automatiquement les interfaces réseaux, le nom d'hôte des machines, ainsi que la résolution de nom statique de toutes les machines du réseau. Les fichiers de configuration sont placés dans `/rw/copy/etc` et sont recopiés dans `/etc` à chaque démarrage des machines. Nous utilisons cette stratégie afin d'utiliser le

même disque monté sur `/` pour les trois machines virtuelles, `/rw` étant un disque dédié à chaque machine. Ceci implique que `/` est en mode *snapshot*, c'est à dire que les modifications apportées au système de fichiers sont perdue à l'extinction de la machine virtuelle. En outre `/rw`, comme son nom l'indique, est persistant, même après instinction de la machine virtuelle. Il restera ensuite les opérations suivantes à effectuer.

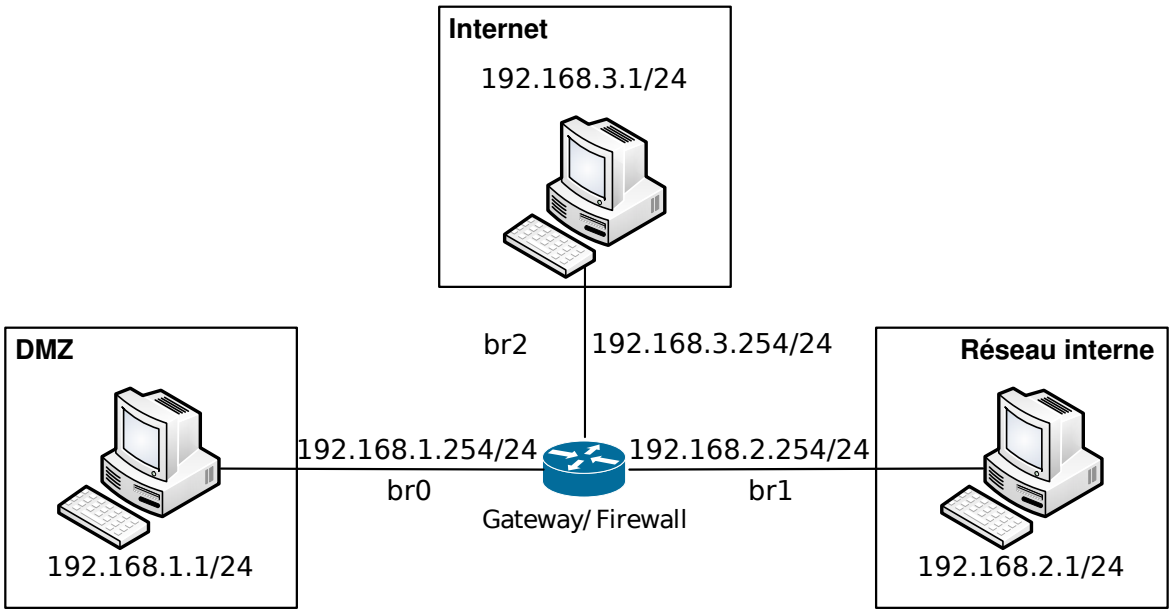


FIGURE 1 – Schéma complet du TP.

Activez le relayage réseau sur le pare-feu avec la commande

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

ou

```
sudo net.ipv4.ip_forward = 1
```

N'oubliez pas d'ajouter les routes sur les machines Réseau interne et DMZ.

3 Mise en place des services

Pour tester les règles du firewall il nous faudrait avoir divers serveurs (HTTP, FTP, SMTP, DNS, etc.) qu'il serait fastidieux de configurer et mettre en route. On va plutôt simuler ces serveurs avec netcat. La commande

```
netcat -v -l [-u -n] -s <ip> -p <port>
```

permet de lancer un serveur qui écoute (option -l de listen) sur un port en TCP (ou UDP is les options -u -n sont utilisées). L'option -v (verbose) indique d'afficher plus de choses que normalement. Pensez à utiliser l'adresse IP de l'interface qui vous intéresse, si vous tapez 127.0.0.1 l'écoute ne se fera que sur l'interface lo (on dit en loopback).

Pour se connecter en au serveur créé en utilisant netctat, utilisez la commande

```
netcat -v [-u -n] <ip> <port>
```

Pour éviter de relancer systématiquement les serveurs une fois que le client se déconnecte. Il est possible de les lancer dans une boucle infinie.

```
while true; do <command>; done
```

Cette boucle pourra être terminée à l'aide du signal `SIGSTOP` (Ctrl+z) suivi de l'envoi d'un signal `SIGINT` avec la commande `kill %1`. `%1` référence le premier job dans la liste des jobs du shell courant.

Vous pouvez vérifier l'état d'écoute réseau de la commande netcat avec le programme `lsof`. Il faudra être root car les fichiers réseau ouvert sont ici privilégiés.

```
sudo lsof -a -i4 -i6 -itcp -iudp -c netcat
```

1. Lancez sur la machine de la DMZ des serveurs en écoute en TCP sur 80, 443, 25, 53 (il vous faudra être root) et en UDP sur 53.
2. Testez que vous pouvez les contacter avec netcat [-u -n] ip-destination port. Ce que vous tapez dans le terminal du client devrait s'afficher dans le terminal du serveur et réciproquement. Attention : si vous lancez un serveur et client TCP et que vous coupez l'un des deux, l'autre recevra un TCP RST et s'en rendra compte. Si vous êtes en UDP par contre ceci ne sera pas le cas et le serveur donnera des problèmes. On vous conseille donc de, pour chaque test avec un serveur UDP, couper le serveur avec Ctrl-C et le relancer.

4 Quelques commandes

Commandes de base	**Description**
-t table	Si la table n'est pas spécifiée, alors la table par défaut est ***filter***.
-j cible	Exécute l'action définie par cible.
-A chaîne	Append. Ajoute la règle à la fin de la chaîne indiquée.
-I chaîne index	Insert. Ajoute la règle à la place indiquée par index. Les indices commencent à un.
-D chaîne index	Delete. Élimine une règle de la chaîne indiquée.
-F [chaîne]	Flush. Efface toutes les règles de la chaîne et table sélectionnée. Si chaîne n'est pas précisé ça efface toutes les chaînes de la table.
-L [-v]	List. Liste les règles de toutes les chaînes de la table demandée (avec plus de détail si -v est précisé).
-s adresse-ip[/masque]	Adresse source du paquet. Peut être une plage si masque est précisé.
-d adresse-ip[/masque]	Adresse destination du paquet. Peut être une plage si masque est précisé.
-i interface	Interface d'entrée du paquet.
-o interface	Interface de sortie du paquet.

Protocoles	**Description**
-p protocole	Protocole du paquet : icmp, udp, tcp, etc.
-p tcp (ou udp) --sport ports	Ports source. Valeur unique ou plage au format portdede-part :portdefin.
-p tcp (ou udp) --dport ports	Ports destination. Valeur unique ou plage au format portdede-part :portdefin.
-p tcp --syn	Flag SYN. Comme pour beaucoup d'autres options on peut l'inverser. "! --syn" c'est tout paquet avec le flag SYN à zéro.
--icmp-type type	Type ICMP. Généralement echo-request ou echo-reply.

Voici quelques exemples [1] d'utilisation :

```
iptables -t filter -A FORWARD -i eth1 -s 192.168.1.0/24 -d 192.168.2.0/24
-p tcp --dport 23 -j ACCEPT
```

Ici, on ajoute (-A) une règle dans la chaîne FORWARD de la table ***filter***. Cette règle stipule que les paquets TCP (-p tcp) qui transitent par le firewall (FORWARD) en arrivant par l'interface ethernet eth1, dont l'adresse source est une adresse du réseau 192.168.1.0/24, dont l'adresse destination est une adresse du réseau 192.168.2.0/24 et dont le port destination est le port 23 sont acceptés.

1. Tirés du TP firewalls de Vincent Nicomette et Eric Alata de l'INSA Toulouse

`iptables -t filter -L -v`

Cette commande vous permet de visualiser toutes les règles que vous avez configurées dans la table ***filter***.

5 Configuration de base du firewall

Dans toutes les questions on vous demande d'utiliser la page de manuel d'iptables (man iptables) pour savoir quelle syntaxe utiliser.

Exercice 1

1. Pour éviter de devoir retaper de nombreuses commandes à chaque fois que vous faites une erreur et enregistrer votre travail au fur et à mesure vous allez créer un fichier contenant les commandes et exécuter les commandes d'un bloc en tapant `source nomdufichier`. Ce fichier vous permettra aussi d'accéder aux adresses IP par des variables. Créez un fichier `firewallrc` définissant des variables de type EXTNET, DMZ, INTNET (pour les réseaux), et EXTIF, DMZIF, INTIF (pour les interfaces). Pour les utilisateurs de machine virtuelle, placez ce fichier dans `/rw`.
2. Ajoutez à ce fichier la commande `iptables -X` qui efface les chaînes autres que celles par défaut (vous aurez plus de détails sur les chaînes utilisateur par la suite). Ajoutez également des commandes pour vider toutes les tables standard. Testez ce fichier en ajoutant quelques règles à la main et en l'appelant avec `source firewallrc`. Après cela `iptables -L -t` table devrait montrer des tables vides.

Dans la suite pour chaque question ajoutez les nouvelles commandes au fichier `firewallrc` et testez qu'en l'exécutant vous arrivez à une configuration en accord avec l'ensemble des questions antérieures. Ne partez pas sans avoir sauvegardé votre fichier de configuration, on en aura besoin pour les autres TPs.

Exercice 2

1. L'option -P (politique, ou policy) permet de dire quelle est la cible (i.e. action) par défaut pour une chaîne et une table donnée si un paquet n'a été accepté ni éliminé en fin de chaîne. Changez la cible par défaut de ACCEPT à DROP pour la chaine FORWARD de la table ***filter***. Testez à nouveau les communications par ping et ssh. Vous ne devriez plus pouvoir communiquer entre les réseaux mais tous les ordinateurs devraient pouvoir échanger avec le routeur. Changez aussi la politique à DROP pour les chaînes INPUT et OUTPUT de la table ***filter***. Testez à nouveau. Tout devrait être coupé.
2. Ajoutez une règle à la table ***filter*** pour que tous les paquets en loopback soient acceptés dans les deux sens. Testez. Ping et ssh devraient passer en local et pas passer depuis les autres ordinateurs vers le routeur. Il ne devrait pas être possible d'envoyer des paquets depuis le routeur vers l'extérieur.

On veut autoriser les connexions HTTP vers la DMZ. On va suivre une approche simpliste et pas très bonne pour introduire après la bonne manière de faire ceci.

Exercice 3

1. Ajoutez des règles à la table ***filter*** pour autoriser les paquets uniquement allant vers la machine de la DMZ (192.168.1.1) en HTTP (port destination 80). Essayez d'initier une connexion depuis un autre réseau. Ça ne devrait pas marcher (les paquets sortants ne sont pas autorisés).
2. Ajoutez des règles à la table ***filter*** pour autoriser les paquets dans l'autre sens (i.e. venant de la DMZ et ayant pour port source le port 80). Testez à nouveau, ça devrait marcher maintenant.
3. Essayez maintenant, depuis la machine de la DMZ, de démarrer une connexion sortante avec pour port source 80 et pour port destination 22 vers la machine de l'Intranet (`sudo nc -p 80 192.168.2.1 22`). Pouvez vous dire pourquoi ceci est extrêmement problématique ? Si ce n'est pas le cas faites appel à l'enseignant de TP.

6 Suivi des connexions

Le suivi des connexions est la capacité de créer et maintenir des informations sur les connexions, comme les adresses, ports, types de protocole, l'état et la durée de la connexion, etc. Le suivi de connexion est bien sûr propre aux pare-feux à états (stateful). Pour utiliser les fonctionnalités de suivi de connexion il faut s'assurer que les bons modules sont chargés dans le noyau avec les commandes :

```
modprobe ip_conntrack
modprobe ip_conntrack_ftp
```

Pour créer une règle qui prend en compte l'état de la connexion on peut utiliser les options :

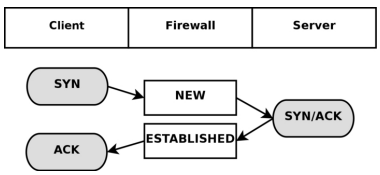
```
-m state --state état1,...,étatN
```

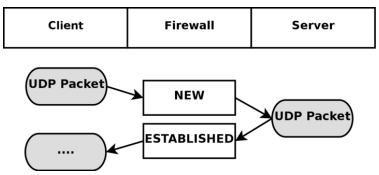
Cette syntaxe particulière veut dire que nous souhaitons utiliser l'extention `state` (similaire à `conntrack` qui est plus récent), voir nota bene) de netfilter apportée par les modules précédents. L'option `--state` permet de passer un état à ce module. les différents états possibles sont :

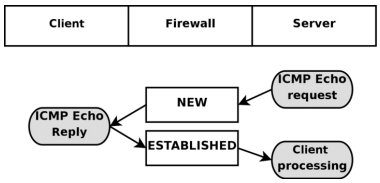
États	**Description**
NEW	le paquet est le premier d'une nouvelle connexion
ESTABLISHED	Le paquet appartient à une connexion déjà établie
RELATED	Le paquet n'appartient pas à une connexion établie mais a une relation avec une autre connexion déjà établie (ICMP, connexion de données FTP pour une connexion de contrôle déjà établie, etc.).

D'autres états sont possibles (INVALID, UNTRACKED) mais nous ne les utiliserons pas dans ce TP. La notion de connexion dans iptables est assez large et existe pour tout protocole où il y a une réponse. C'est au moment de recevoir la première réponse à un paquet NEW que la connexion passe à ESTABLISHED. Ces images (de www.iptables.info) illustrent quel est le moment exact du changement.

Nota bene : historiquement le suivi des connexions était apporté par l'extension `state`. Il est aujourd'hui remplacé par l'extension `conntrack` que nous vous avons demandé de charger précédemment. L'extension `conntrack` est invoquée à l'aide de `-m conntrack`, l'état cette fois-ci est choisi à l'aide de l'option `--ctstate` à la place de l'option `--state` de l'extension state. Bien que les états supportés soient les mêmes, il se peut que `conntrack` arrive à suivre plus de protocoles que `state`. Vous le verrez lors de ce TP.







Comme le module ***conntrack*** agit avant toute chaîne (comme le montre la première figure de ce TP), les paquets sont traités comme ESTABLISHED à partir du premier paquet de réponse inclus.

De l'utilisation du suivi pour autoriser les connexions

Le suivi de connexions nous permet d'autoriser des connexions dans un sens donné de façon beaucoup plus simple et sûre qu'avec des règles dites stateless comme on a fait dans la section précédente. L'idée est qu'on autorise par défaut tout paquet RELATED ou ESTABLISHED quelque soit le réseau d'entrée et de destination. Ensuite, quand on veut autoriser des connexions dans un sens entre deux points on ajoute une règle autorisant pour les paramètres souhaités (ip et ports source et destination, protocole, etc.) les paquets dans l'état NEW. Cette règle plus ou moins précise nous permettra de dire quelles initialisations on autorise, et la règle générale assurera que ce qui découle de notre autorisation passe et le reste pas.

Exercice 4

Ajoutez les lignes de type `modprobe ...` à votre fichier de configuration pour vous assurer que vous pouvez utiliser le suivi. Ajoutez également une règle à la table ***filter*** pour que tous les paquets dans l'état RELATED ou ESTABLISHED soient acceptés.

6.1 Flux vers la DMZ

Exercice 5

1. Effacez les règles stateless (section 5, exercice 3) et ajoutez une règle à la table ***filter*** pour autoriser les connexions vers la machine de la DMZ (192.168.1.1) en HTTP/HTTPS (ports 80 et 443). Testez.

2. Ajoutez des règles à la table ***filter*** pour autoriser les connexions vers la machine de la DMZ en SMTP (port 25) et pour que la machine de la DMZ puisse se connecter vers n'importe quel ordinateur du réseau interne en SMTP. Testez.
3. Ajoutez des règle à la table ***filter*** pour autoriser les connexions vers la machine de la DMZ en DNS (en TCP et UDP). Testez.
4. Autorisez le ping vers les serveurs de la DMZ. Testez

6.2 Flux d'administration du routeur

Exercice 6
Autorisez les connexions SSH entrantes au niveau du routeur si elle ont pour origine une machine donnée de l'Intranet (par exemple 192.168.2.1). Testez que vous pouvez vous connecter depuis cette machine et pas d'une autre de l'Intranet ou d'un autre réseau.

6.3 Filtrage ingress/egress (règles anti-spoofing)

Exercice 7
Ajoutez des règles pour que tout paquet qui a une IP source hors de son réseau d'origine soit éliminé.

6.4 Protection contre des attaques classiques

Exercice 8
1. Protection contre les scans de ports. Utilisez le module limit (c.f. page du manuel) pour que le nombre de paquets tcp avec un flag SYN, ACK, FIN, RST soient limités à 5 par seconde (utilisez l'option --tcp-flags). Testez.
2. Protection contre les inondations. Utilisez le module limit (c.f. page du manuel) pour que le nombre de pings par seconde acceptés soit limité à 5 par seconde. Testez.

7 Chaînes utilisateur

7.1 Préambule : un peu de LOG

La cible LOG permet d'envoyer des messages au démon syslogd via le noyau. Les messages reçus par syslogd depuis le noyau sont disponibles dans le fichier /var/log/kern.log. Vous pouvez voir les derniers messages de ce fichier en tapant tail [-f] /var/log/kern.log. L'option -f vous permet de suivre (follow) les modifications du fichier : au lieu de vous redonner la main, tail vous montre les dernières lignes du fichier et se met en attente de modifications. Dès que quelque chose s'ajoute à la fin du fichier tail vous affiche les nouvelles lignes.

Exercice 9
1. Tapez tail -f /var/log/kern.log puis insérez une clé USB en regardant au niveau du terminal. Vous devriez pouvoir suivre en direct les modifications apportées. Maintenant vous pouvez vérifier en laissant cette commande tourner quels sont les logs que vos règles ajouteront.
2. On souhaite enregistrer dans les logs tout événement de type DROP. Ajoutez devant chaque règle ayant pour cible DROP une règle identique avec pour cible LOG. N'oubliez pas qu'il y a une règle par défaut en fin de chaîne ayant pour cible DROP !
3. On souhaite limiter le nombre de logs enregistrés, pour éviter des attaques de type déni de service qui satureraient le disque avec des logs inutiles. Utilisez le module limit pour limiter le nombre de logs à 10 par seconde.
4. Par défaut LOG envoie des messages à syslogd avec un niveau d'alerte nommé warning. Ce niveau d'alerte requiert par convention une action rapide d'un administrateur or ce n'est pas ce que l'on veut ici. Nous souhaitons juste conserver une information dans les logs pour pouvoir la consulter plus tard si besoin est. Le bon niveau d'alerte est celui qui est nommé info. Modifier les règles de log en ajoutant l'option --log-level info.

Ici on a souhaité combiner deux actions. Dans d'autres cas on voudra combiner plus d'actions. C'est relativement embêtant de devoir reproduire ces ensembles d'actions dans le code, et de devoir tous les modifier

à chaque fois que l'on souhaite faire un changement sur le principe de fonctionnement (comme ici avec le niveau d'alerte). En réalité l'action faire un log et un drop c'est un enchaînement qu'on pourrait appeler log-and-drop. On voudrait bien définir et pouvoir appeler autant de fois que l'on veut comme on appelle une fonction dans un langage de programmation. C'est une des choses que nous permettent de faire les chaînes utilisateur.

7.2 Présentation des chaînes utilisateur

Il est possible de construire d'autres chaînes que celles par défaut et de mettre dedans des règles comme on l'a fait jusqu'à maintenant. On peut utiliser toute chaîne utilisateur comme cible d'une règle. Quand on fait cela, le traitement dans la chaîne actuelle s'arrête, on saute à la chaîne cible et on traite toutes les règles, si aucune de ces règles ne provoque un arrêt du traitement, on retourne à la chaîne initiale et on continue le traitement.

Ces chaînes sont généralement utilisées pour deux choses. Premièrement ça permet de définir des enchaînements d'actions comme une chaîne et quand on souhaite réaliser cet enchaînement d'actions on appelle la chaîne comme cible. Il y a une cible supplémentaire dont on ne vous a pas parlé jusqu'à maintenant mais qui est particulièrement utile dans les chaînes utilisateur. La cible RETURN fait que le traitement dans la chaîne utilisateur actuelle s'arrête et qu'on retourne immédiatement à la chaîne qui nous a appelés (comme en programmation avec les fonctions), même si la chaîne utilisateur n'est pas complètement traitée. L'utilisation des chaînes utilisateurs et de RETURN permet de faire des fichiers de configuration modulaires, comme dans n'importe quel langage de programmation.

Le deuxième cas d'usage est l'amélioration des performances. Si vous avez cent règles dans la chaîne FORWARD, chaque paquet qui passe par le routeur va subir cent tests, ce qui va le ralentir considérablement. Les chaînes utilisateurs vous permettent de rendre le traitement arborescent et de réduire le nombre de règles traitées. Supposons que dans forward vous avez des règles du type "Si le protocole est tcp aller à la cible regles-pour-tcp" et dix autres règles comme celle-ci envoyant vers dix chaînes utilisateur qui regroupent par paquets de dix (dans un cas idéal) les cent règles initiales. Pour chaque paquet on ne vérifiera que onze règles au total et vous aurez considérablement amélioré les performances.

7.3 Utilisation pratique

Pour créer une chaîne on utilise la commande : iptables -N nouvelle-chaine.Pour effacer une chaîne il suffit d'utiliser la commande iptables -X [chaine-a-effacer]. Si aucune chaîne n'est précisée, la commande essaye d'effacer toutes les chaînes utilisateur. Attention : une chaîne utilisateur ne peut être effacée que si elle est vide (utilisez -F), et aucune règle ne l'utilise.

Exercice 10

1. Créez une chaîne utilisateur log-and-drop pour factoriser tous les cas de LOG puis DROP que vous pouvez dans votre fichier de configuration. Remplacez ces enchaînements par des appels à cette chaîne. Attention : il faudra modifier le début de votre fichier pour qu'il vide cette chaîne avant de faire iptables -X.
2. Essayez de rendre votre traitement un peu plus arborescend pour que quand vous ajouterez des nouvelles règles tout ne se fasse pas de façon linéaire.

8 Annexes

8.1 Rappels de commandes réseaux

8.1.1 Probe and set ethernet port

```
ethtool -p eth0
ip link set eth0 up
ip address add 192.168.0.1/24 dev eth0
```

8.1.2 Utilitaires

```
netcat -v -l [-u -n] ip port
ping -i 0.005 ip
```

8.1.3 Transferer un fichier

```
Dest: netcat -l ip_addr port > nom_du_ficher
Src: cat nom_du_fichier | netcat ip_addr port
```

8.1.4 Routeur

```
sysctl net.ipv4.ip_forward=1
```

8.1.5 Hôtes

```
ip route add default via "addr routeur" dev "interface"
```

8.2 Flush complet des règles iptables

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

8.3 Conflit avec connlimit

Attention aux conflits avec d'autres règles, par exemple :

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -p ! icmp -j ACCEPT
```