Chapitre 5

Les iptables

Nous avons vu qu'une des méthodes fondamentales de sécurisation des réseaux informatique est le filtrage des paquets. Maintenant que nous en connaissons le principe, voyons comment le mettre en place.

Le principe est d'écrire un programme qui récupère toutes les trames entrantes et sortantes, les analyse et les fait suivre ou non suivant le filtrage désiré. On a intérêt à utiliser un modèle en couches qui facilite le travail. Le traitement des trames est effectué au niveau noyau (non accessible aux utilisateurs, y compris au super-utilisateur) par le sous-sytème réseau du systèmes d'exploitation. On ajoute donc une fonctionnalité au niveau noyau à ce sous-système avec une interface utilisateur (uniquement le super-utilisateur en général) permettant de paramétrer le filtrage désiré.

Dans le cas du système d'exploitation Linux, plus particulièrement des versions de noyaux 2.4 et 2.6, la fonctionnalité est implémentée par **netfilter**, dont l'adresse Web du projet est :

`http://www.netfilter.org`

dont l'interface utilisateur en « ligne de commande » s'appelle **iptables**. Il s'agit d'un projet lancé en 1998 par Rusty Russell, auteur du programme précédent appelé *ipchains*, sous licence GPL et intégré au noyau Linux 2.3 en mars 2000.

5.1 Mise en place

Avant d'utiliser `iptables` il faut, d'une part, configurer le noyau pour qu'il prenne en charge `Netfilter` et, d'autre part, mettre en place l'interface utilisateur `iptables`.

Heureusement, les distributions de Linux les installent la plupart du temps par défaut. Pour le vérifier, placez-vous en super-utilisateur (soit dans un terminal virtuel texte, soit grâce à la commande `su` dans un pseudo-terminal) et exécuter la commande :

```
# iptables -L
```

Si tout est installé et que vous n'avez encore jamais touché à `iptables`, les chaînes sont vides et vous 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
```

Si ce n'est pas le cas, on pourra se référer, par exemple, au chapitre 5 de [AND-06] pour l'installation.

5.2 Tables et chaînes

Au début, comme le montre l'option `-L` de la commande `iptables`, tout passe dans toutes les directions (*policy ACCEPT*), c'est-à-dire pour chacune des trois chaînes par défaut (`INPUT`, `OUTPUT` et `FORWARD`) de la table `filter` (celle par défaut).

Notion de chaîne.- Nous allons donner un ensemble de règles de traitement des trames. Par commodité, cet ensemble de règles est décomposé en plusieurs sous-ensembles. Dans le vocabulaire de `netfilter` une **table** spécifie un tel sous-ensemble. Un sous-ensemble est lui-même décomposé en sous-ensembles, spécifié par une **chaîne** dans le vocabulaire de `netfilter`. Il existe des chaînes dont le nom est prédéfini et des chaînes créées par l'utilisateur, quelquefois appelées **sous-chaînes**.

Parcours des tables et des chaînes.- Le noyau de Linux récupère en tâche de fond les trames dans le tampon de chaque interface réseau ainsi évidemment que les trames crées sur l'ordinateur et `Netfilter` les traite une à une (voir figure 5.1) en les faisant passer, dans un certain ordre, dans un ensemble de chaînes.

Une trame destinée à l'ordinateur passe par les chaînes suivantes dans cet ordre :

table	Chaîne
raw	PREROUTING
mangle	PREROUTING
nat	PREROUTING
mangle	INPUT
filter	INPUT

Remarquons tout d'abord que plusieurs chaînes portent le même nom mais on les différencie

par la table à laquelle ils appartiennent. De plus la même table est impliquée dans plusieurs parties de la route.

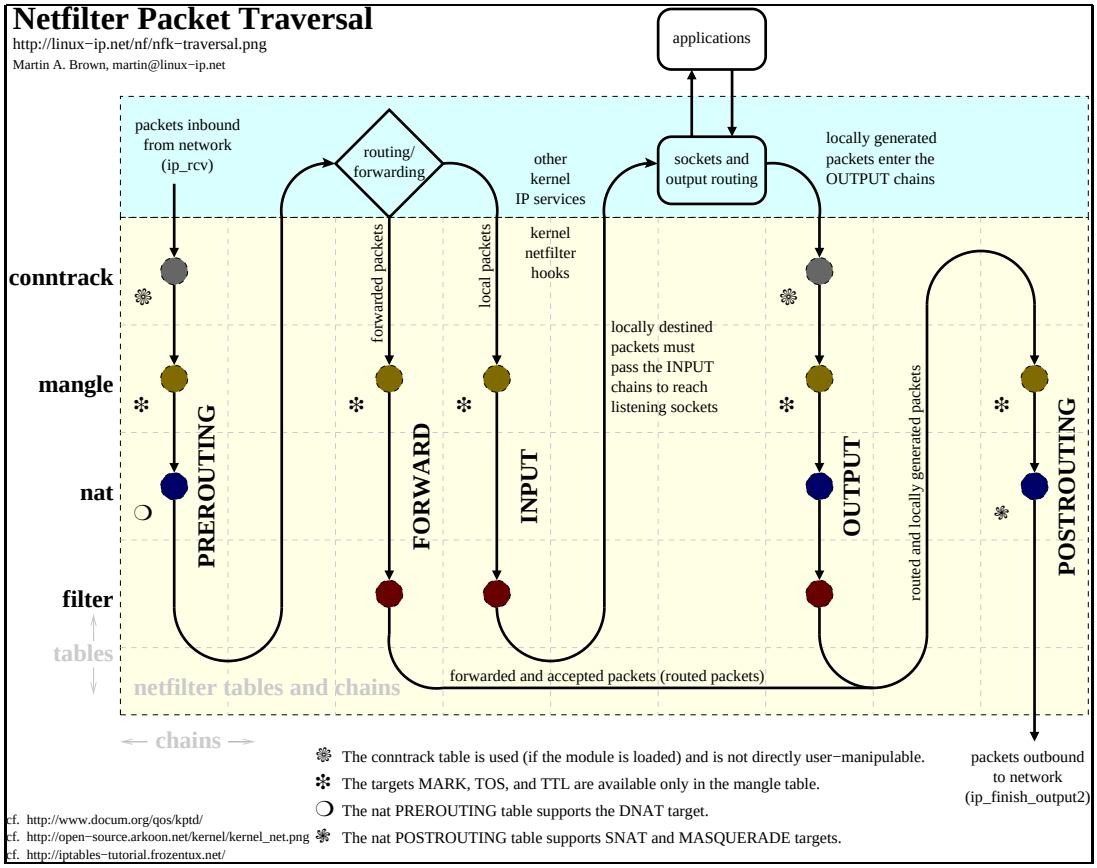

FIG. 5.1 – Parcours des tables et des chaînes prédéfinies

Nous rencontrons quatre tables :

- la table **raw** (brute) ou **conntrack** ;
- la table **mangle** (estropieuse), utilisée en général pour changer certains champs en-tête de la trame (par exemple le TOS de l'en-tête IP) ;
- la table **nat**, utilisée pour la traduction d'adresses ;
- la table **filter**, utilisée surtout pour le filtrage d'adresses.

Les chaînes `PREROUTING` sont utilisées avant toute décision de routage par Linux et `INPUT` après celles-ci.

De même, le noyau de Linux fait passer les trames crées par l'ordinateur et destinées à un autre hôte par les chaînes suivantes dans cet ordre avant de les envoyer à l'interface réseau :

table	Chaîne
raw	OUTPUT
mangle	OUTPUT
nat	OUTPUT
filter	OUTPUT
mangle	POSTROUTING
nat	POSTROUTING

Comme ci-dessus les chaînes `OUTPUT` sont traitées avant toute décision de routage et les chaînes `POSTROUTING` après celles-ci.

Enfin une trame récupérée sur une interface réseau mais destinée à un autre hôte passe par les chaînes suivantes dans cet ordre avant d'être envoyées une autre interface réseau :

table	Chaîne
raw	PREROUTING
mangle	PREROUTING
nat	PREROUTING
mangle	FORWARD
filter	FORWARD
mangle	POSTROUTING
nat	POSTROUTING

5.3 Les commandes iptables

Syntaxe générale d'une commande.- Une commande de l'interface de `netfilter` se présente sous la forme suivante :

```
iptables [-t table] command [match] [target/jump]
```

où le nom de la table est `raw`, `mangle`, `nat` ou `filter`. Par défaut il s'agit de `filter`, comme nous l'avons vu lors de notre premier test.

Si tous les critères (spécifiés par le paramètre `match`) sont rencontrés alors l'instruction (spécifiée par le paramètre `target/jump`) est exécutée.

Ensemble des commandes.- La description de l'ensemble des commandes disponibles se retrouve, de façon usuelle, grâce à :

```
man iptables
```

avec l'introduction [AND-06], devenue classique. Nous allons cependant décrire les commandes les plus usuelles, en commençant par la commande d'affichage déjà utilisée.

La commande d'affichage.- La commande `-L` ou `--list` affiche toutes les entrées de la chaîne spécifiée (ou de toutes les chaînes si aucune chaîne n'est spécifiée).

Par exemple, au début, la commande suivante devrait renvoyer ce que l'on voit après la commande :

```
# iptables -t nat -L PREROUTING
 Chain PREROUTING (policy ACCEPT)
 target  prot opt source  destination
```

5.4 Gestion des chaînes

5.4.1 Politique de filtrage

Commençons par une commande qui ne concerne pas réellement la gestion générale des chaînes.

Notion.- Comme nous l'avons vu ci-dessus, par défaut il n'y a aucun filtrage, c'est-à-dire que toutes les trames sont acceptées, reroutées et envoyées.

On peut filtrer finement, soit par l'adresse IP, soit par le port, mais il serait beaucoup trop long de donner une liste exhaustive de tout ce qui existe. Une première étape est de choisir ce qui est appelé la **politique de filtrage** :

- Si on rejette peu d'éléments, le mieux est de commencer par tout accepter, puis d'indiquer individuellement ce que l'on n'accepte pas.
- Si on est très précautionneux et que l'on rejette beaucoup, le mieux est de commencer par tout rejeter, puis d'indiquer individuellement ce que l'on accepte.
- Dans un cas médian, on choisit l'une ou l'autre de ces politiques, plus ou moins au hasard.

Syntaxe.- Pour choisir la politique de filtrage, on utilise la commande :

`iptables -P` *chain policy*

où *chain* est le nom d'une chaîne et *policy* l'une des deux valeurs `ACCEPT` (pour tout accepter sauf ce qui sera explicitement rejeté) ou `DROP` (pour tout rejeter sauf ce qui sera explicitement accepté).

Bien entendu il faut passer en paramètre le nom de la table s'il ne s'agit pas de la table par défaut. On peut remplacer `-P` par `--policy`.

Exemple.- Rejetons de façon un peu dure toutes les trames entrantes pour notre ordinateur :

```
# iptables -P INPUT DROP
```

Vérifions que la table est bien changée :

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

 Chain FORWARD (policy ACCEPT)
 target  prot opt source  destination

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

Vérifions surtout que l'on ne peut plus rien recevoir, par exemple en essayant dans le navigateur web quelque chose de nouveau (n'oublions pas que le cache pourrait nous afficher une page web sans aller la chercher effectivement sur le réseau).

Remarque.- L'exemple précédent ne fonctionnera que si l'ordinateur sur lequel on se trouve peut fonctionner sans aucun service réseau, ce qui n'est pas le cas si on utilise, par exemple, le système de fichiers en réseau `nfs`. Il faudra dans ce denier cas permettre l'accès aux fichiers par une règle que nous verrons ci-après.

5.4.2 Ajouter une règle

Une fois la politique choisie, on peut ajouter des règles aux chaînes pour affiner notre filtrage, et il en a évidemment bien besoin.

Syntaxe.- On peut ajouter une règle grâce à la commande :

`iptables -A` *chain* `[match][target/jump]`

avec `A` pour *append*, qui ajoute la règle à la fin de la liste.

Il faut faire attention que l'ordre dans la liste peut être important.

On peut utiliser `--append` au lieu de `-A`.

Exemple.- Nous avons tout rejeter en entrée. Permettons cependant ce qui provient du serveur web `wwww.u-pec.fr` de notre université.

En permettant à nouveau toutes les entrées, cherchons l'adresse de son serveur grâce à un `ping`, ce qui donne 193.48.143.10.

N'acceptons en entrée que ce qui provient de ce serveur :

```
# iptables -P INPUT DROP
# iptables -A INPUT -s 193.48.143.10 -j ACCEPT
```

sans avoir détaillé, pour l'instant, les règles de syntaxe mais celles utilisées sont suffisamment explicites.

Vérifions que la table est bien changée :

```
# iptables -L
 Chain INPUT (policy DROP)
 target  prot opt source       destination
 ACCEPT  all -- 193.48.143.10  anywhere

 Chain FORWARD (policy ACCEPT)
 target  prot opt source       destination

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

Vérifions que l'on ne peut rien utiliser (pas de réception de courrier ou de `ping`), pas même le navigateur web, ni même `http://www.u-pec.fr` puisque le serveur DNS est bloqué. Par contre `http://193.48.143.10` donne accès à la page d'accueil du serveur web de notre université et permet de naviguer à partir de celle-ci tant que l'on ne quitte pas cette adresse.

5.4.3 Supprimer des règles

Syntaxe.- On peut supprimer des règles d'une chaîne une à une grâce à la commande `-D` (ou `--delete`), avec les mêmes paramètres que pour - A, ou vider la chaîne complète grâce à la commande :

```
# iptables -F
```

ou `--flush`.

Exemple.- Vérifier que :

```
# iptables -F
```

nous ramène à l'état antérieur :

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

 Chain FORWARD (policy ACCEPT)
 target  prot opt source  destination

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

5.4.4 Utilisation d'un script

On n'a pas écrit grand chose pour l'instant. Tout sera perdu lors de la prochaine session mais on peut le réécrire facilement. Lorsqu'on aura écrit quelque chose de plus conséquent, cela deviendra gênant. Heureusement, comme pour toutes commandes en ligne, on peut utiliser un script. Écrivons donc :

```
#!/bin/bash
iptables -P INPUT DROP
iptables -A INPUT -s 193.48.143.10 -j ACCEPT
```

dans un fichier texte appelé, par exemple, `moniptables.sh`.

Revenons à la situation par défaut où l'on accepte tout, rendons le script exécutable (on ne sait jamais) :

```
# chmod +x ./moniptables.sh
```

puis exécutons-le script en tant que super-utilisateur :

```
# sh ./moniptables.sh
```

On revient à la situation où toute entrée est bloquée sauf l'accès au serveur de notre université.

5.5 Règles de filtrage

Maintenant que nous avons vu le principe d'utilisation de `iptables`, détaillons les règles de filtrage que celui-ci permet.

5.5.1 Filtrages généraux

Syntaxe.- Un filtrage général permet d'accepter (ou de rejeter) :

- tous les paquets d'une interface réseau donnée (en entrée ou en sortie) :

  `iptables -A` *chain* `-i` *interface* `[jump]`
  `iptables -A` *chain* `-o` *interface* `[jump]`

  où `-i` peut être remplacé par `--in-interface` et `-o` par `--out-interface`. Pour la première règle, la chaîne ne peut être que INPUT, FORWARD et PREROUTING et pour la seconde que OUTPUT, FORWARD et POSTROUTING.
- une adresse IP source ou destination donnée :

  `iptables -A` *chain* `-s` *interface* `[jump]`
  `iptables -A` *chain* `-d` *interface* `[jump]`

  où `-s` peut être remplacé par `--src` ou `--source` et `-d` par `--dst` ou `--destination`. L'adresse peut également utiliser un masque CIDR, par exemple `192.168.0.0/24` ou un masque, par exemple `192.168.0.0/255.255.255.0` pour obtenir le même effet que précédemment.
- un protocole donné :

  `iptables -A` *chain* `-p` *protocol* `[jump]`

  où `-p` peut être remplacé par `--protocol`. Le protocole est TCP, UDP ou ICMP ou l'un de ceux spécifiés dans `/etc/protocols`. On peut utiliser une valeur entière, par exemple 1 pour ICMP.

Exemples.- 1^o) Le problème en bloquant tout par défaut est que même l'interface locale (*loopback*, par exemple utilisée pour l'impression) est bloquée. On a donc intérêt à insérer au tout début des règles :

```
#iptables -A INPUT -i lo -j ACCEPT
```

- 2^o) Nous avons déjà rencontré la règle :

```
#iptables -A INPUT -s 194.214.24.116 -j ACCEPT
```

- 3^o) 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 :

```
#iptables -A INPUT -p ICMP -j ACCEPT
```

5.5.2 Filtrages UDP

Syntaxe.- Un filtrage UDP permet d'accepter (ou de rejeter) suivant le port source ou le port de destination :

`iptables -A` *chain* `-p udp --sport` *port* `[jump]`
`iptables -A` *chain* `-p udp --dport` *port* `[jump]`

où `--sport` peut être remplacé par `--source-port` et `--dport` par `--destination-port`.

Le numéro de port peut être remplacé par le nom du service tel que spécifié dans le fichier `/etc/services`, par exemple `ssh` au lieu de 22.

On peut utiliser des intervalles de port, par exemple `22 :80`.

Exemple.- On peut permettre le trafic entrant sur un port spécifique, par exemple 22 (traditionnellement utilisé par `ssh`) :

```
#iptables -A INPUT -p udp --dport 22 -j ACCEPT
```

5.5.3 Filtrages TCP

Syntaxe.- Un filtrage TCP permet d'accepter (ou de rejeter) :

- suivant le port source ou le port de destination :

  `iptables -A` *chain* `-p tcp --sport` *port* `[jump]`
  `iptables -A` *chain* `-p tcp --dport` *port* `[jump]`

- suivant l'état des drapeaux :

  `iptables -A` *chain* `-p tcp --tcp-flags` *drapeaux-à-comparer drapeaux* `[jump]`

  les drapeaux étant `SYN`, `FIN`, `ACK`, `RST`, `URG` et `PSH` séparés par des virgules si on en spécifie plusieurs. On peut aussi utiliser les valeurs `ALL` et `NONE`.

Exemples.- 1^o) On peut permettre le trafic entrant sur un port spécifique, par exemple 22 (traditionnellement utilisé par `ssh`) :

```
#iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

- 2^o) On peut permettre le trafic entrant des segments qui essaient d'établir une connexion :

```
#iptables -A INPUT -p tcp --tcp-flags SYN,FIN,ACK SYN -j ACCEPT
```

même si on ne voit pas à quoi cela peut servir pour l'instant.

5.6 Fichiers de messages

Introduction.- Nous avons vu que l'appariement permet d'écarter ou d'accepter certains paquets grâce à un saut (*jump* en anglais). On peut également repérer certains paquets et les signaler dans le fichier de messages du noyau (généralement connu sous le nom de fichier *log*) engendrés par le démon *syslogd* (ou *rsyslogd*), que l'on peut lire grâce à la commande en ligne *dmesg*.

Vous pouvez consulter :

```
man dmesg
```

pour une introduction au fichier des messages du noyau.

Syntaxe.- On peut signaler le transférer certaines informations d'un paquet sélectionné grâce à un appariement dans le fichier *log* grâce à la commande :

`iptables -A` *chain match* `-j LOG --log-level` *level*

où *level* peut prendre l'une des valeurs `debug`, `info`, `warning`, `error`, entre autres.

Exemple.- Signalons tous les paquets entrants au fichier *log* :

```
#iptables -A INPUT -j LOG --log-level debug
```

Effectuons quelques manipulations faisant intervenir le réseau et vérifions le fichier `log` qui aura alors la forme suivante :

[7992.488140] IN=eth0 OUT= MAC=00 :dd :10 :01 :04 :c8 :00 :1e :74 :47 :d6 :88 :08 :00
SRC=63.118.7.207 DST=192.168.1.11 LEN=64 TOS=0x00 PREC=0x00 TTL=113 ID=12872
PROTO=TCP SPT=443 DPT=52085 WINDOW=16384 RES=0x00 ACK SYN URGP=0
[7992.622664] IN=eth0 OUT= MAC=00 :dd :10 :01 :04 :c8 :00 :1e :74 :47 :d6 :88 :08 :00
SRC=63.118.7.207 DST=192.168.1.11 LEN=1492 TOS=0x00 PREC=0x00 TTL=113 ID=12971
DF PROTO=TCP SPT=443 DPT=52085 WINDOW=65368 RES=0x00 ACK URGP=0

---

dont nous n'avons retenu que deux descriptions de trames (qui constituent deux lignes du fichier *log*, chacune étant présentée ici sur trois lignes pour une question de mise en forme).

Marquage des lignes.- Le fichier *log* comprend un très grand nombre de lignes. On peut faire précéder les lignes engendrées par *netfilter* par une chaîne de caractères, en utilisant l'option `--log-prefix`, ce qui en facilitera le reperérage et permettra d'utiliser *grep* pour les extraire. .

Exemple.- Signalons tous les paquets entrants au fichier *log* :

```
#iptables -A INPUT -j LOG --log-level debug --log-prefix "PAQUET ENTRANT "
```

Effectuons quelques manipulations faisant intervenir le réseau et vérifions le fichier `log` qui aura alors la forme suivante :

[9743.721270] PAQUET ENTRANT IN=eth0
OUT= MAC=00 :dd :10 :01 :04 :c8 :00 :1e :74 :47 :d6 :88 :08 :00 SRC=192.168.1.1
DST=192.168.1.11 LEN=66 TOS=0x00 PREC=0x00 TTL=64 ID=28418 DF
PROTO=UDP SPT=53 DPT=50886 LEN=46

Remarque.- Les paquets dont une copie est envoyée au démon *syslog* poursuivent par ailleurs leur cheminement et traversent la règle suivante de la table. Pour signaler le paquet et le rejeter, on utilisera en général deux règles consécutives analogues mais avec des sauts différents.

5.7 Nouvelles chaînes

Nous avons dit que l'utilisateur peut créer de nouvelles chaînes. Voyons comment et quelle peut en être l'utilité.

Syntaxe.- On utilise la commande `-N` pour créer une nouvelle table :

`iptables -N` *chain*

le nom de la nouvelle chaîne ne pouvant pas être l'un des noms de chaînes prédéfinies.

Exemple.- Créons une nouvelle chaîne chargée de placer dans le fichier *log* les paquets acceptés.

```
iptables -N LOGACCEPT
iptables -A LOGACCEPT -j LOG --log-prefix "LOGACCEPT : "
iptables -A LOGACCEPT -j ACCEPT
```

Exercice.- Créer de même une nouvelle chaîne `LOGDROP` chargée de placer dans le fichier *log* les paquets à rejeter et qui rejette ces paquets.

Exemple.- Écrivons maintenant un script créant les tables LOGACCEPT et LOGDROP et qui utilise celles-ci :

```
#!/bin/bash

iptables -N LOGACCEPT
iptables -A LOGACCEPT -j LOG --log-prefix "LOGACCEPT: "
iptables -A LOGACCEPT -j ACCEPT

iptables -N LOGDROP
iptables -A LOGDROP -j LOG --log-prefix "LOGDROP: "
iptables -A LOGDROP -j DROP

iptables -A INPUT -s 193.48.143.10 -j LOGACCEPT
iptables -A INPUT -j LOGDROP
```

5.8 Politique de retransmission

Intérêt.- Supposons que notre réseau d'entreprise soit formé de deux réseaux locaux d'adresses privées 192.168.1.0/24 et 192.168.2.0/24 reliés par une passerelle (avec une interface réseau eth0 vers le premier sous-réseau et eth1 vers le second sous-réseau). Une machine d'un des sous-réseaux peut évidemment communiquer avec toute machine de ce même sous-réseau. Par contre on peut, dans ces conditions, décider si une machine du premier sous-réseau peut communiquer ou non avec une machine du second sous-réseau. La politique **FORWARD** permet à l'administrateur de contrôler où les paquets peuvent être routés au sein du réseau d'entreprise.

Mise en place.- Pour autoriser la retransmission du sous-réseau local 192.168.2.0/24 dans tout le réseau d'entreprise, on utilise les règles suivantes :

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

Cette règle permet à une machine d'un sous-réseau de communiquer avec une machine de l'autre sous-réseau en utilisant des adresses IP privées mais pas à une machine sur Internet : si les machines du réseau d'entreprise ci-dessus envoyent directement des paquets sur Internet, les réponses ne reviendront jamais, parce que l'IANA (institut de régulation des adresses) a considéré ce réseau (avec d'autres) comme privé, et a restreint son usage à des réseaux locaux isolés d'Internet.

Bien entendu, pour interdire de façon explicite la communication entre les machines d'un sous-réseau à un autre, on utilisera :

```
# iptables -A FORWARD -i eth1 -j DROP
# iptables -A FORWARD -o eth1 -j DROP
```

5.9 Traduction d'adresse

Nous avons expliqué antérieurement le principe et l'intérêt de la traduction d'adresse : un **serveur NAT** (pour *Network Address Translation*) traduit les adresses IP source et/ou destination des paquets en adresses différentes en jouant sur les numéros de port. Le serveur NAT reçoit le paquet, change l'en-tête IP (il réécrit l'adresse source ou destination et recalcule la somme de contrôle du paquet) puis change l'en-tête de transport (il donne un numéro de port non utilisé et recalcule la somme de contrôle si celle-ci a un sens).

La partie *netfilter* de Linux permet de mettre en place facilement ces traductions d'adresse : on n'a pas en particulier à se préoccupper explicitement de la gestion des numéros de port ou du calcul des sommes de contrôle, effectués pour nous par le noyau Linux. On se sert pour cela de la table `nat` et de l'une des cinq cibles suivantes :

- La cible principale est **SNAT**, employée pour changer l'adresse source des paquets. Avec cette cible, le serveur NAT effectuera automatiquement sur les paquets de la traduction d'adresse source dans un sens et de la traduction d'adresse source inverse dans l'autre sens.
- La cible **DNAT** est employée de même pour changer l'adresse de destination.
- La cible **MASQUERADE** (*camouflage* en français) est l'analogue de la cible SNAT mais s'utilise lorsque les adresses sont données par DHCP. On spécifiera donc l'interface réseau au lieu d'une adresse IP.
- La dernière cible s'appelle **REDIRECT**.

5.9.1 Traduction d'adresse source

Intérêt.- La traduction d'adresse réseau source est utilisée le plus fréquemment lorsque nous n'avons pas les moyens ou ne voyons pas l'intérêt d'avoir une adresse IP publique pour chacun des postes d'un réseau local. Disons, par exemple, que nous avons un réseau local 192.168.1.0/24. L'une des machines de ce réseau local sera transformée en **passerelle** (*gateway* en anglais) possédant deux interfaces réseau : l'une d'adresse réseau privée, disons 192.168.1.0, reliée au réseau local et l'autre d'adresse réseau publique, disons 217.115.95.34, reliée à Internet. Cette passerelle jouera le rôle de serveur SNAT, traduisant toutes les adresses 192.168.1.0/24 en l'adresse publique. De cette façon, il y aura 5-10 clients ou beaucoup plus qui utiliseront la même adresse IP publique partagée.

Mise en place.- Pour *netfilter*, la cible SNAT n'est valide que dans la table `nat`, à l'intérieur de la chaîne POSTROUTING. C'est, en d'autres termes, la seule chaîne dans laquelle on peut utiliser SNAT, c'est-à-dire juste avant que le paquet ne soit définitivement envoyé à l'extérieur : ceci signifie que toute autre fonction sur la machine sous Linux (routage, filtrage de paquets) verra le paquet non modifié par la traduction d'adresse. Cela signifie aussi que l'option '-o' (interface de sortie) peut être utilisée.

La traduction d'adresse source est spécifiée en utilisant l'option '-j SNAT', suivie de '–to-source' qui spécifie une adresse IP, une plage d'adresses IP et éventuellement un port ou une plage de ports (pour les protocoles UDP et TCP seulement) :

- pour changer l'adresse source en 1.2.3.4 :

  ```
  # iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
  ```

- pour changer l'adresse source en 1.2.3.4, 1.2.3.5 ou 1.2.3.6 (*netfilter* choisira l'adresse à utiliser pour une meilleure répartition des charges) :

  ```
  # iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.6
  ```

– pour changer l'adresse source en 1.2.3.4, port 1-1023 :

```
# iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT \
--to 1.2.3.4 :1-1023
```

Exemple.- En reprenant notre exemple de début et en supposant que la passerelle a pour interface `eth1` vers l'extérieur (d'adresse 217.115.95.34) et `eth0` vers le réseau local, nous réaliserons la traduction d'adresse voulue grâce à la règle :

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 -j SNAT \
--to 217.115.95.34
```

5.9.2 Traduction d'adresse de destination

Intérêt.- La traduction d'adresse de destination est utile lorsque, dans la configuration ci-dessus, on veut qu'un serveur visible de l'extérieur par l'adresse 217.115.95.34 se trouve en fait physiquement sur une machine du réseau local qui n'est pas la passerelle. Par exemple, nous pouvons faire tourner un serveur web et ftp sur la même machine, tandis qu'il existe une machine physiquement distincte contenant différents services d'IRC que les employés travaillant à domicile ou étant sur la route peuvent utiliser en relation avec le personnel sur le site de l'entreprise. Nous pouvons alors faire fonctionner tous ces services sur la même IP depuis l'extérieur *via* DNAT.

Mise en place.- Pour *netfilter*, la cible DNAT n'est valide que dans la table `nat`, à l'intérieur de la chaîne PREROUTING, au moment où le paquet arrive : cela signifie que toute autre fonction de la machine sous Linux (routage, filtrage de paquets) verra le paquet aller vers sa destination 'réelle'. Cela signifie aussi que l'option '-i' (interface d'entrée) peut être utilisée.

La traduction d'adresse de destination est spécifiée en utilisant '-j DNAT', suivie de '–to-destination' qui spécifie une adresse IP, une plage d'adresses IP et éventuellement un port ou une plage de ports (pour les protocoles UDP et TCP seulement) :

– pour changer l'adresse de destination en 5.6.7.8 :

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8
```

– pour changer l'adresse de destination en 5.6.7.8, 5.6.7.9 ou 5.6.7.10 :

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8-5.6.7.10
```

– pour changer l'adresse de destination du trafic web en 5.6.7.8, port 8080 :

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 \
-j DNAT --to 5.6.7.8 :8080
```

Exemple.- Le dernier cas nous montre comment faire dans le cas d'un serveur déporté.

Remarque.- Si la politique par défaut est DROP dans la chaîne FORWARD, on doit ajouter une règle autorisant la retransmission de requêtes HTTP entrantes afin que le routage NAT de destination soit possible :

```
# iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 5.6.7.8 -j ACCEPT
```

Cette règle autorise la retransmission de requêtes HTTP entrantes depuis la passerelle vers la destination souhaitée sur le serveur HTTP derrière la passerelle.

5.9.3 Camouflage

Intérêt.- Le camouflage d'adresse IP est un cas particulier de traduction d'adresse source à utiliser pour des adresses IP allouées dynamiquement, comme pour la liaison téléphonique avec les fournisseurs d'accès. Dans ce cas on ne spécifie pas explicitement l'adresse source publique mais l'interface : *netfilter* utilisera l'adresse source de l'interface par laquelle le paquet sort. De plus, si le lien est rompu, les connexions (qui sont de toute façon perdues) sont oubliées, ce qui évite des problèmes éventuels quand la connexion est rétablie avec une nouvelle adresse IP.

Mise en place.- Pour camoufler tout ce qui sort par l'interface ppp0, on utilise la règle suivante :

```
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```