

# Filtrage réseau avec netfilter/iptables

Philippe Latu

philippe.latu(at)inetdoc.net

<http://www.inetdoc.net>

Étude du filtrage réseau dans le contexte d'un routeur central (*hub*) connecté à un routeur d'agence (*spoke*) via un lien WAN. Comme dans les autres supports de travaux pratiques de cette série, on assimile ces configurations types à des interconnexions entre réseaux locaux et réseaux étendus. Les questions de ce support sont présentées comme une introduction pas à pas au filtrage réseau. On débute avec les outils, on poursuit avec les fonctions de suivi de communication (*stateful inspection*) sur une interface, puis on ajoute à ce filtrage les fonctions de traduction d'adresse (*NAT*).

## Table des matières

1. Copyright et Licence .....	1
1.1. Méta-information .....	1
1.2. Conventions typographiques .....	2
2. Architecture réseau étudiée .....	3
2.1. Topologie type .....	3
2.2. Plan d'adressage WAN .....	3
3. Les outils de filtrage réseau .....	4
3.1. Questions sur iptables .....	4
3.2. Questions sur netfilter .....	5
4. Règles de filtrage communes à toutes les configurations .....	6
5. Règles de filtrage sur le poste routeur d'agence ( <i>spoke</i> ) .....	9
6. Règles de filtrage sur le routeur central ( <i>hub</i> ) .....	12
7. Règles de filtrage avec identification des protocoles .....	13
7.1. Protocole ICMP .....	13
7.2. Règles de filtrage communes à toutes les configurations .....	14
8. Documents de référence .....	15
8.1. IETF & IANA .....	15
8.2. Distribution Debian GNU/Linux .....	15
8.3. Site inetdoc.net .....	15

## 1. Copyright et Licence

Copyright (c) 2000,2012 Philippe Latu.  
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Copyright (c) 2000,2012 Philippe Latu.  
Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation License), version 1.3 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est incluse dans la section intitulée « Licence de Documentation Libre GNU ».

### 1.1. Méta-information

Cet article est écrit avec *DocBook*<sup>1</sup> XML sur un système *Debian GNU/Linux*<sup>2</sup>. Il est disponible en version imprimable au format PDF : [interco.netfilter.qa.pdf](http://interco.netfilter.qa.pdf)<sup>3</sup>.

Toutes les commandes utilisées dans ce document ne sont pas spécifiques à une version particulière des systèmes UNIX ou GNU/Linux. C'est la distribution *Debian GNU/Linux* qui est utilisée pour les tests présentés. Voici une liste des paquets contenant les commandes :

- *procps* - The /proc file system utilities
- *net-tools* - The NET-3 networking toolkit
- *ifupdown* - High level tools to configure network interfaces
- *iputils-ping* - Tools to test the reachability of network hosts

<sup>1</sup> <http://www.docbook.org>

<sup>2</sup> <http://www.debian.org>

<sup>3</sup> <http://www.inetdoc.net/pdf/interco.netfilter.qa.pdf>

- hping3 - Active Network Smashing Tool
- iptables - Administration tools for packet filtering and NAT
- iptstate - Top-like state for netfilter/iptables

## **1.2. Conventions typographiques**

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou *prompt* spécifique au niveau des droits utilisateurs nécessaires sur le système.

- Toute commande précédée de l'invite \$ ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite # nécessite les privilèges du super utilisateur.

## 2. Architecture réseau étudiée

Les manipulations sur le système de filtrage réseau présentées ici s'appuient sur une maquette type qui illustre une «branche» de la topologie *Hub and Spoke*. Dans cette topologie en étoile, le **routeur central** joue le rôle d'un concentrateur qui fournit les accès réseau à tous les **routeurs d'agence**.

### 2.1. Topologie type

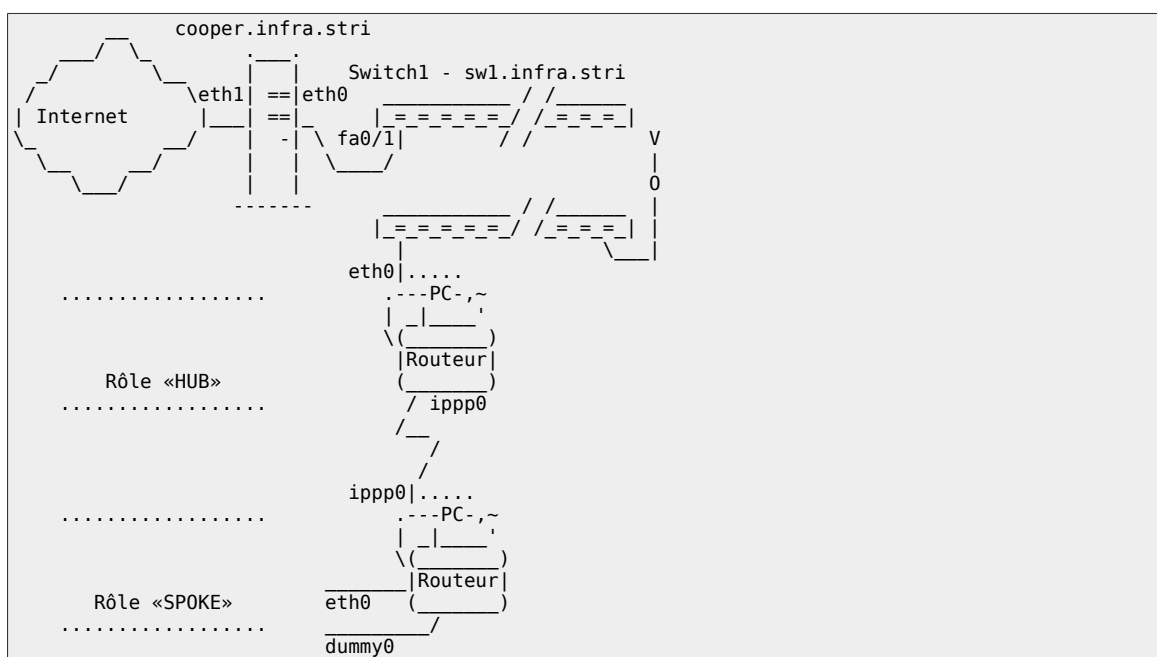
La topologie étudiée associe deux postes de travaux pratiques avec deux rôles distincts.

Routeur central, *hub*, *Remote Access Server*, RAS

Ce poste réalise une interconnexion LAN/WAN. Il fournit un accès Internet au routeur d'agence client via son interface WAN. Il dispose de son propre accès Internet via son interface LAN Ethernet.

Routeur d'agence, *spoke*, *Customer Premises Equipment*, CPE

Ce poste réalise aussi une interconnexion LAN/WAN. À la différence du routeur central, il obtient l'accès Internet sur son interface WAN et il met cet accès à disposition d'un réseau local d'extrémité via une interface LAN Ethernet (et/ou) une autre «pseudo» interface réseau.



Pour les besoins des questions de travaux pratiques ci-après, on se limite à un scénario simple d'utilisation des fonctions de filtrage réseau.

Le **routeur central** doit s'assurer que le trafic réseau qu'il route vers l'Internet provient bien de son **routeur d'agence** auquel il a attribué une adresse IP via PPP. Ce routeur central doit limiter le volume de trafic ICMP pour éviter les éventuels dénis de services relatifs à ce protocole. Il peut aussi intercepter toutes les requêtes DNS du routeur d'agence en exploitant un service de type *cache only* pour éviter les falsifications des réponses aux questions émises par les postes client du réseau d'agence.

Le **routeur d'agence** doit s'assurer que le trafic réseau entrant sur son interface WAN est bien relatif à une demande qui a été émise via cette même interface. Ce routeur d'agence doit «masquer» son réseau local d'extrémité à l'aide des fonctions de traduction d'adresse source S-NAT. Il doit aussi ouvrir un accès d'administration SSH tout en se protégeant des attaques de type dictionnaire qui consistent à essayer de se connecter au poste avec des milliers de couples *login/password* connus. Enfin, ce même routeur d'agence doit donner accès à un service Web hébergé dans un périmètre dédié au sein de cette agence.

Pour traiter les questions ci-après, il est vivement conseillé de s'appuyer sur la **Section 8**, « Documents de référence ».

### 2.2. Plan d'adressage WAN

Le plan d'adressage des liaisons WAN reprend celui du support de travaux pratiques *Topologie Hub & Spoke avec le protocole PPP*<sup>4</sup>.

Tableau 1. Plan d'adressage liaisons WAN

Poste routeur central	bus	N° tél.	Adresses IP hub:spoke	N° tél.	Poste routeur d'agence
alderaan	S0.1	104	192.168.104.1:192.168.104.2	105	bespin

<sup>4</sup> [http://www.inetdoc.net/travaux\\_pratiques/interco.ppp.q/](http://www.inetdoc.net/travaux_pratiques/interco.ppp.q/)

Poste routeur central	bus	N° tél.	Adresses IP hub:spoke	N° tél.	Poste routeur d'agence
centares	S0.2	106	192.168.106.1:192.168.106.2	107	coruscant
dagobah	S0.3	108	192.168.108.1:192.168.108.2	109	endor
felucia	S0.4	110	192.168.110.1:192.168.110.2	111	geonosis
hoth	S0.5	112	192.168.112.1:192.168.112.2	113	mustafar
naboo	S0.6	114	192.168.114.1:192.168.114.2	115	tatooine

### 3. Les outils de filtrage réseau

Sur un système GNU/Linux, les fonctions de filtrage réseau sont réparties entre les espaces mémoire noyau (*kernel space*) et utilisateur (*userspace*).

Que l'on utilise un noyau fourni par la distribution ou le noyau construit à l'issue des travaux pratiques *Configuration des fonctions réseau & compilation du noyau Linux*<sup>5</sup>, les fonctions de filtrage réseau sont disponibles sous forme de modules que l'on (charge|décharge) de la mémoire système en cours d'exécution. Les outils de filtrage réseau du noyau Linux chargent dynamiquement ces modules en fonction de la syntaxe des règles de filtrage saisies.

#### 3.1. Questions sur iptables

1. Quels paquets contiennent les outils utilisateur principaux de manipulation des fonctions de filtrage réseau ?

Rechercher dans le cache du gestionnaire de paquets de la distribution des mots clés tels que `iptables` ou `firewall` à l'aide des commandes **apt-cache** ou **aptitude**.

Dans le cadre des travaux pratiques, seuls les outils élémentaires sont utilisés pour faciliter la compréhension des mécanismes de suivi de communication du système de filtrage réseau. On ne s'intéresse donc pas aux paquets qui fournissent des solutions de filtrage «clé en main».

On sait que la partie *userspace* des fonctions de filtrage réseau s'appelle `iptables`. On lance donc une recherche avec ce mot clé dans la base de données des paquets Debian et on installe les paquets intéressants.

Une recherche simple dans le catalogue des paquets donne le résultat suivant.

```
$ aptitude search iptables
p arno-iptables-firewall - single- and multi-homed firewall script with DSL/ADSL support
i iptables - Outils d'administration pour le filtrage de paquets et le NAT
p iptables-dev - iptables development files
p iptables-persistent - Simple package to set up iptables on boot
p libiptables-chainmgr-perl - Perl extension for manipulating iptables policies
p libiptables-parse-perl - Perl extension for parsing iptables firewall rulesets
```

Une recherche plus précise dans les descriptions de paquets donne une liste plus étoffée.

```
# aptitude search '~diptables' | grep ipt
p apf-firewall - easy iptables based firewall system
p arno-iptables-firewall - single- and multi-homed firewall script wi
p firehol - An easy to use but powerful iptables state
p fwsnort - Snort-to-iptables rule translator
p ipkungfu - iptables-based Linux firewall
i iptables - Outils d'administration pour le filtrage d
p iptables-dev - iptables development files
p iptables-persistent - Simple package to set up iptables on boot
i iptstate - Top-like state for netfilter/iptables
p libiptables-chainmgr-perl - Perl extension for manipulating iptables p
p libiptables-parse-perl - Perl extension for parsing iptables firewa
p mxallowd - Anti-Spam-Daemon using nolistening/iptables
p netscript-2.4 - Linux 2.4.x (and 2.6.x) router/firewall ne
p uif - Advanced iptables-firewall script
p uruk - Very small firewall script, for configurin
i xtables-addons-common - Extensions targets and matches for iptable
p xtables-addons-source - Extensions targets and matches for iptable
```

Comme on doit rester à un faible niveau d'intégration des règles de filtrage de manière à bien illustrer les mécanismes de suivi de communication. Les deux paquets retenus sont `iptables` et `iptstate`. On utilise les commandes usuelles d'installation et de consultation des informations sur ces deux paquets.

```
# aptitude install iptables iptstate
<snipped/>
# apt-cache show iptables
<snipped/>
# apt-cache show iptstate
<snipped/>
```

<sup>5</sup> [http://www.inetdoc.net/travaux\\_pratiques/interco.kernel.q/](http://www.inetdoc.net/travaux_pratiques/interco.kernel.q/)

2. Quelles sont les options de la commande iptables qui permettent de visualiser les règles de filtrage réseau actives ainsi que les compteurs correspondants ? Quelle option faut-il préciser pour spécifier la table consultée : netfilter ou nat.

Consulter les pages de manuels via `# man iptables`.

La consultation des pages de manuels permet de relever le jeu d'options `-vL`.

```
# iptables -vL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
```

La table netfilter est utilisée de façon implicite alors que la table nat de traduction d'adresses doit être appelée explicitement.

```
# iptables -vL -t nat
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
```

3. Comment visualiser les modules chargés dynamiquement en fonction de l'utilisation des règles de filtrage réseau ?

Utiliser la commande qui sert à lister les modules chargés en mémoire avant et après avoir consulté les tables de filtrage réseau pour la première fois.

La commande **lsmod** sert à lister les modules chargés en mémoire. En exécutant cette commande avant et après avoir utilisé **iptables**, on visualise par différence les nouveaux modules chargés par chaque appel. Par exemple, l'exécution des deux commandes de consultation de la question précédente provoque le chargement des modules suivants.

- Après consultation de la table netfilter :

```
# lsmod |less
Module                Size Used by
iptables_filter        2258 0
ip_tables              13899 1 iptables_filter
x_tables               12845 1 ip_tables
<snipped/>
```

- Après consultation de la table nat :

```
# lsmod |less
iptables_nat           4299 0
nf_nat                 13388 1 iptables_nat
nf_conntrack_ipv4     9833 3 iptables_nat,nf_nat
nf_conntrack          46535 3 iptables_nat,nf_nat,nf_conntrack_ipv4
nf_defrag_ipv4        1139 1 nf_conntrack_ipv4
iptables_filter        2258 0
ip_tables              13899 2 iptables_nat,iptables_filter
x_tables               12845 2 iptables_nat,ip_tables
<snipped/>
```

4. Quels sont les outils de sauvegarde et de restauration des jeux de règles de filtrage réseau fournis avec le paquet iptables ?

Consulter la liste des fichiers du paquet iptables.

La liste des fichiers du paquet contient les outils recherchés : `# dpkg -L iptables |grep bin`. Les deux programmes **iptables-save** et **iptables-restore** permettent respectivement de sauvegarder et de restaurer l'ensemble des règles des tables netfilter et nat.

Ces programmes sont indispensables pour éditer, insérer ou retirer des règles sans avoir à se préoccuper de l'ordre de saisie. De plus, le programme de restauration se charge de l'effacement des règles précédentes.

### 3.2. Questions sur netfilter

1. Comment identifier la version du noyau utilisée et la disponibilité des fonctions de filtrage réseau de cette version ?

Après avoir utilisé la commande «historique» d'identification de la version du noyau, faire une recherche dans l'arborescence des modules de ce noyau avec la commande **find**. Trouver les fichiers dont le nom comprend la chaîne netfilter.

On utilise la commande **uname** que l'on associe à une recherche dans l'arborescence des modules du noyau en cours d'exécution.

```
$ uname -r
2.6.32-5-amd64
$ find /lib/modules/`uname -r` -type d -name netfilter
/lib/modules/2.6.32-5-amd64/kernel/net/bridge/netfilter ❶
/lib/modules/2.6.32-5-amd64/kernel/net/ipv4/netfilter ❷
/lib/modules/2.6.32-5-amd64/kernel/net/decnet/netfilter
/lib/modules/2.6.32-5-amd64/kernel/net/ipv6/netfilter ❸
/lib/modules/2.6.32-5-amd64/kernel/net/netfilter ❹
```

- ❶ Fonctions de filtrage au niveau liaison. Ces fonctions ne sont pas utilisées ici.
- ❷ Fonctions de filtrage au niveau réseau utilisant le protocole IPv4. C'est dans ce répertoire que se trouvent les modules utilisés dans ces travaux pratiques.
- ❸ Fonctions de filtrage au niveau réseau utilisant le protocole IPv6. Ces fonctions ne sont pas utilisées ici.
- ❹ Fonctions communes de filtrage réseau indépendantes des niveaux et des protocoles utilisés. C'est notamment à ce niveau que l'on trouve les modules de la machine d'état de suivi de communications.

**2. Quels sont les objets du système de fichiers virtuel /proc relatifs aux fonctions de filtrage réseau du noyau Linux ?**

Avec le chargement des modules en mémoire système, de nouvelles entrées apparaissent dans l'arborescence /proc. Même si l'arborescence /proc n'est pas un véritable système de fichiers, il est possible d'effectuer des recherches toujours à l'aide de la commande **find**.

Le chargement des trois premiers modules entraîne la création des entrées relatives aux noms chaînes, aux correspondances et aux prises de décisions.

```
# find /proc/net/ -name "*tables*"
/proc/net/ip_tables_targets
/proc/net/ip_tables_matches
/proc/net/ip_tables_names
```

La consultation des règles de la table nat entraîne la création de toutes les entrées nécessaires à la machine d'état de suivi de communication.

```
# find /proc/net/ -name "*conntrack*"
/proc/net/ip_conntrack_expect
/proc/net/ip_conntrack
/proc/net/nf_conntrack
/proc/net/nf_conntrack_expect
/proc/net/stat/ip_conntrack
/proc/net/stat/nf_conntrack
```

**3. Comment visualiser les informations de la machine d'état de suivi de communication ?**

Rechercher les entrées de l'arborescence /proc dont le nom comprend la chaîne conntrack.

La section «7.2 Les entrées de conntrack» du *Tutoriel iptables*<sup>6</sup> décrit précisément les différents champs du suivi de communication.

Le programme iptstate affiche les entrées de la table de suivi de communication sur le même mode que la commande **top**.

Les états sont directement consultables à partir du fichier virtuel /proc/net/ip\_conntrack. Par exemple :

```
# cat /proc/net/ip_conntrack
<snipped/>
udp 17 27 src=192.0.2.3 dst=192.0.2.1 sport=54932 dport=53 \
  packets=1 bytes=59 src=192.0.2.1 dst=192.0.2.3 sport=53 dport=54932 \
  packets=1 bytes=259 mark=0 secmark=0 use=2
```

L'exemple ci-dessus donne l'état du suivi de communication d'une requête DNS entre un poste client avec l'adresse 192.0.2.3 et le port source 54932 et un serveur avec l'adresse 192.0.2.1.

**4. Règles de filtrage communes à toutes les configurations**

La mise en place du filtrage réseau sur les équipements doit répondre à deux principes.

- Comme les équipements d'interconnexion mis en œuvre dans ces travaux pratiques délimitent des périmètres de faible dimension, on a une connaissance exhaustive des flux réseaux sur le système. On adopte donc la règle : *tout trafic réseau non autorisé est interdit*.
- Pour exploiter au mieux les fonctionnalités offertes par le noyau Linux, on s'appuie sur le suivi de communication (*stateful inspection*) pour obtenir un filtrage réseau le plus efficace possible. On cherche ~~donc à suivre~~ la règle d'or d'écriture des règles de filtrage qui consiste à *décrire le plus précisément*

<sup>6</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

possible le premier paquet qui doit être enregistré dans la table de suivi de communication. Cette règle de description du premier paquet doit être placée après celle(s) qui laisse(nt) passer les flux réseau déjà enregistrés dans la machine d'état de suivi de communication.

1. Quelle est la syntaxe de la commande **iptables** sur la politique par défaut à appliquer sur les chaînes de la table netfilter pour respecter le premier principe de filtrage énoncé ci-dessus ?

De façon très classique, on consulte les pages de manuels de la commande **iptables** et on recherche le mot clé **policy**. La stratégie retenue suppose que l'on implante règles d'autorisation des flux réseaux valides et que tout autre trafic soit éliminé. La politique par défaut à appliquer sur les trois chaînes est donc : DROP.

La section «9.3. Commandes» du *Tutoriel iptables*<sup>7</sup> donne aussi la syntaxe de configuration de *cible par défaut* pour les chaînes élémentaires : INPUT, FORWARD et OUTPUT.

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
# iptables -vL
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination

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

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

2. Quelle est la syntaxe de la commande **iptables** qui autorise le trafic réseau déjà enregistré dans la machine d'état de suivi de communication sur les chaînes INPUT et OUTPUT ?

La recherche de la correspondance **state** dans les pages de manuel de la commande **iptables** permet de sélectionner les états ESTABLISHED et RELATED à appliquer sur les chaînes.

La section «7.3. États de l'espace utilisateur» du *Tutoriel iptables*<sup>8</sup> décrit les correspondances entre les états et les flux réseau.

```
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -vL
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 ACCEPT 0 -- any any anywhere anywhere state RELATED,ESTABLISHED

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

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 ACCEPT 0 -- any any anywhere anywhere state RELATED,ESTABLISHED
```

À partir de cette étape, on utilise les programmes **iptables-save** et **iptables-restore** pour optimiser les manipulations. Ces programmes présentent un grand intérêt dans la mesure où l'affichage des règles de filtrage est plus condensé.

```
# iptables-save >/var/lib/iptables/active
# vim /var/lib/iptables/active
# iptables-restore </var/lib/iptables/active
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
COMMIT
```

La commande **iptables-restore** doit être utilisée après chaque édition du fichier `/var/lib/iptables/active`.

3. À partir des règles de filtrage précédentes, est-il possible d'émettre ou de recevoir du trafic réseau non enregistré dans la machine d'état de suivi de communication ?

~~Faire des tests ICMP, DNS et HTTP. Conclure et justifier.~~

<sup>7</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

<sup>8</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

La réponse est non. La politique par défaut sur les chaînes INPUT et OUTPUT étant positionnée à DROP, tout nouveau paquet entrant ou sortant est rejeté. Pour qu'une communication soit possible, il faudrait avoir enregistré un flux réseau dans la machine d'état avant d'appliquer ce jeu de règles de filtrage.

4. Quelle est la syntaxe de la commande **iptables** qui autorise le trafic réseau depuis (chaîne OUTPUT) et vers (chaîne INPUT) l'interface de boucle locale de l'équipement ?

Pour que les processus locaux au système puissent communiquer entre eux via la pile de protocole TCP/IP, il est *essentiel* d'autoriser le trafic sur l'interface de boucle locale `lo`. La recherche de la correspondance `state` dans les pages de manuel de la commande **iptables** permet de sélectionner l'état `NEW` pour autoriser le premier paquet depuis et vers cette interface. Tous les autres paquets devront correspondre aux règles déjà écrites ci-dessus.

On ajoute deux nouvelles règles sur les chaînes INPUT et OUTPUT qui admettent respectivement tous les paquets entrant et sortant par l'interface de `lo` dans la table de suivi des communications.

```
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
COMMIT
```

À partir de ce jeu de règles, on peut lancer un test ICMP : `# ping -c 4 127.0.0.1`.

5. Quelle est la syntaxe de la commande **iptables** qui autorise le trafic sortant sur l'interface LAN en sortie du système ?

Comme pour la question précédente, les processus locaux au système doivent pouvoir émettre du trafic via la pile de protocole TCP/IP. Il est donc préférable d'autoriser le trafic sortant sur l'interface LAN. On utilise à nouveau l'état `NEW` pour autoriser le premier paquet depuis l'interface. Tous les autres paquets devront correspondre aux communications déjà enregistrées dans les tables de suivi.

On ajoute une règle sur la chaîne OUTPUT qui admet, dans la table de suivi des communications, les paquets sortant par l'interface `eth0`.

```
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
```

6. Quelle est la syntaxe de la commande **iptables** qui autorise le trafic ICMP en entrée du système en limitant le nombre des nouvelles requêtes à 5 par minute ?

Interdire tout trafic ICMP est une très mauvaise idée du point de vue administration réseau. Pour autant, il est très facile de se prémunir contre les tentatives de saturation du trafic sur les interfaces en limitant le nombre de requêtes simultanées en entrée sur toutes les interfaces. Dans un premier temps on se contente de cette règle unique très simple même s'il est judicieux de valider les types et les codes des messages ICMP. Dans un deuxième temps, on distinguera les types de messages ; voir [Section 7.1, « Protocole ICMP »](#).

On peut qualifier le fonctionnement de la limitation de trafic à l'aide des commandes **ping** et **hping3** à partir d'un hôte distant.

La recherche de la correspondance `limit` dans les pages de manuel de la commande **iptables** permet de compléter la syntaxe de la règle d'autorisation du trafic ICMP avec l'état `NEW` pour le suivi de communication.

```
# grep -v '^#' /var/lib/iptables/active
```

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -m limit --limit 5/min -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
```

Les tests de qualification de la nouvelle règle utilisent la commande usuelle **ping** puis un outil beaucoup moins classique qui offre de nombreuses «possibilités», **hping3**.

```
# ping -n -c 3 192.0.2.3
PING 192.0.2.3 (192.0.2.3) 56(84) bytes of data.
64 bytes from 192.0.2.3: icmp_req=1 ttl=64 time=0.958 ms
64 bytes from 192.0.2.3: icmp_req=2 ttl=64 time=0.746 ms
64 bytes from 192.0.2.3: icmp_req=3 ttl=64 time=0.803 ms

--- 192.0.2.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.746/0.835/0.958/0.095 ms

# hping3 -l --rand-source --fast -c 10 192.0.2.3
HPING 192.0.2.3 (tap0 192.0.2.3): icmp mode set, 28 headers + 0 data bytes
len=28 ip=192.0.2.3 ttl=64 xml:id=5204 icmp_seq=0 rtt=1.0 ms
len=28 ip=192.0.2.3 ttl=64 xml:id=9948 icmp_seq=1 rtt=0.5 ms
len=28 ip=192.0.2.3 ttl=64 xml:id=31892 icmp_seq=2 rtt=0.6 ms
len=28 ip=192.0.2.3 ttl=64 xml:id=48870 icmp_seq=3 rtt=0.6 ms
len=28 ip=192.0.2.3 ttl=64 xml:id=40501 icmp_seq=4 rtt=0.7 ms

--- 192.0.2.3 hping statistic ---
10 packets transmitted, 5 packets received, 50% packet loss
```

On constate que le premier test ne produit aucune erreur alors que la tentative de *spoofing* rapide des adresses IP source entraîne des pertes de paquets ICMP dès que la limite fixée dans la règle de filtrage est atteinte.

Une fois ces règles basiques en place, on peut aborder les filtrages réseau spécifiques à la topologie de travaux pratiques.

## 5. Règles de filtrage sur le poste routeur d'agence (*spoke*)

Suivant le cahier des charges fixé, la première contrainte imposée au poste routeur d'agence est de s'assurer que le trafic entrant sur son interface WAN est bien relatif à une demande émise via cette même interface. Le seconde contrainte étant le routage pour le réseau local d'extrémité, on doit compléter le filtrage avec une traduction d'adresse source liée à cette même interface WAN.

Il est conseillé de travailler à partir du fichier de règles de filtrage établi dans la section précédente. Après chaque édition de ce fichier, la commande **iptables-restore** permet d'appliquer le nouveau jeu de règles après avoir effacé les règles précédentes et remis les compteurs de paquets à zéro.

### 1. Quelle est la syntaxe de la commande **iptables** qui autorise le trafic sortant sur l'interface WAN ?

Relativement à la configuration commune présentée précédemment, il suffit d'ajouter une règle d'autorisation sur la chaîne OUTPUT tout en enregistrant le premier paquet sortant avec l'état NEW.

Avec ce jeu de règles actif, on peut lancer la séquence habituelle des tests ICMP et caractériser l'utilisation des règles en visualisant l'évolution des compteurs avec la commande **iptables -vL**.

Comme dans le cas de l'interface LAN, on ajoute une règle qui admet les nouveaux paquets sortant par l'interface `ipp0` dans la table de suivi des communications via la chaîne OUTPUT.

```
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/min -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
```

```
#-----:: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
```

2. Quelle est la syntaxe de la commande **iptables** qui active la traduction d'adresse source pour le trafic sortant sur l'interface WAN ?

Le but est de configurer le routeur d'agence pour qu'il ne soit visible de l'Internet qu'à travers l'adresse IP délivrée par le routeur central (ou le fournisseur d'accès). Cette opération utilise la table `nat`. On ajoute dans cette table une règle de traduction d'adresse source dynamique liée à l'interface WAN. Il faut rechercher les informations sur la cible `MASQUERADE` dans les pages de manuels de la commande **iptables**.

On ajoute une règle dans la chaîne `POSTROUTING` qui traduit l'adresse IP source de tous les paquets sortant par l'interface `ipp0` avec l'adresse attribuée dynamiquement via le protocole PPP à cette interface.

```
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#-----:: POSTROUTING
-A POSTROUTING -o ipp0 -j MASQUERADE
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----:: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/min -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----:: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
COMMIT
```

3. Ce jeu de règles est-il suffisant pour que le poste se comporte comme un routeur avec une fonction de traduction d'adresses IP sources ?

Cette question comprend deux parties. Il faut s'intéresser dans un premier temps à la fonction de routage proprement dite et consulter l'indicateur d'état du noyau Linux qui correspond à la capacité à transmettre un paquet d'une interface vers une autre. Dans un second temps, il faut identifier dans le système de filtrage la chaîne dédiée au transit des paquets.

La réponse est non. Il manque au moins deux conditions pour que le routage et la transmission des paquets entre les interfaces soient actifs.

- Pour qu'un paquet soit transmis d'une interface réseau vers une autre, il faut s'assurer que le routage est actif au niveau du noyau. Cette fonction est paramétrée par la variable d'état `ip_forward` du système de fichiers virtuel `/proc`. La valeur 1 indique que la fonction routage est active dans le noyau :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Comme la politique par défaut sur la chaîne `FORWARD` est `DROP`, aucun paquet ne peut traverser les règles de filtrage et transiter d'une interface vers l'autre. Sans règle supplémentaire, les tests ICMP doivent incrémenter le compteur `DROP` de la chaîne `FORWARD`.

4. Quelle est la syntaxe de la commande **iptables** qui autorise le transfert des paquets entrant par l'interface LAN vers l'interface WAN ?

Rechercher la syntaxe des règles correspondant à ce qui a déjà été vu dans la mise au point du jeu de règles communes pour les chaînes `INPUT` et `OUTPUT`. Il faut que tout trafic relatif à une demande enregistrée dans la table de suivi des communications soit accepté. Il faut aussi que les nouveaux paquets entrant par l'interface LAN soient admis et enregistré dans cette table.

On implante deux règles dans la chaîne `FORWARD`. Une première règle pour le trafic relatif à une demande déjà enregistrée et une seconde pour les paquets entrant par l'interface LAN.

```
# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#-----:: POSTROUTING
-A POSTROUTING -o ipp0 -j MASQUERADE
COMMIT
*filter
:INPUT DROP [0:0]
```

```

:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/min -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----: FORWARD
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -i eth0 -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
COMMIT
    
```

Cette configuration ne peut être qualifiée qu'avec un trafic devant transiter entre deux interfaces. Avec la configuration de travaux pratiques proposée, il faut connecter un poste supplémentaire sur le même réseau local que celui de l'interface LAN du routeur d'agence. C'est le trafic réseau initié par ce nouvel hôte réseau qui utilise les deux règles de la chaîne FORWARD implantées dans le script ci-dessus.

L'instruction `# iptables -vL FORWARD` affiche les compteurs relatifs à la chaîne FORWARD. Ces compteurs évoluent lorsqu'un nouveau trafic à destination d'un autre réseau apparaît sur une interface.

- Quelle est la syntaxe de la commande **iptables** qui permet l'administration à distance du routeur d'agence (*Spoke*) via son interface WAN en utilisant le protocole SSH ? Proposer une configuration qui offre une protection contre les attaques de type «dictionnaire».

Un premier niveau de réponse consiste à admettre les nouvelles demandes de connexions TCP sur le port numéro 22 sur l'interface WAN. On obtient alors le jeu de règles suivant.

```

# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#-----: POSTROUTING
-A POSTROUTING -o ipp0 -j MASQUERADE
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/min -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
-A INPUT -i ipp0 -p tcp --syn --dport 22 -m state --state NEW -j ACCEPT
#-----: FORWARD
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -i eth0 -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
    
```

Du point de vue sécurité, cette configuration n'est pas très satisfaisante. Sachant que toutes les nouvelles demandes de connexion TCP sont acceptées, on ouvre la porte à toutes les attaques de type «dictionnaire».

La section «10.3.19. Correspondance Recent» du *Tutoriel iptables*<sup>9</sup> décrit précisément les différentes possibilités du module recent. En utilisant cette fonctionnalité, on peut remplacer la solution donnée ci-dessus par le jeu de règles suivant qui limite le nombre de tentatives de connexions à 4 par minute.

```

# grep -v '^#' /var/lib/iptables/active
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#-----: POSTROUTING
-A POSTROUTING -o ipp0 -j MASQUERADE
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----: INPUT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/min \
-m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
-A INPUT -i ipp0 -p tcp --dport 22 -m recent --set --name SSH \
-m state --state NEW -j ACCEPT
-A INPUT -i ipp0 -p tcp --dport 22 -m recent --update --seconds 60 --hitcount 4 \
    
```

<sup>9</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

```

--rttl --name SSH -m limit --limit 5/min -j LOG --log-prefix "SSH_brute_force "
-A INPUT -i ipp0 -p tcp --dport 22 \
  -m recent --update --seconds 60 --hitcount 4 --rttl --name SSH -j DROP
#-----: FORWARD
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -i eth0 -m state --state NEW -j ACCEPT
#-----: OUTPUT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
    
```



### Avertissement

Dans la copie d'écran ci-dessus, des lignes ont été coupées avec des caractères '\ ' dans le but d'optimiser l'affichage. Pour rétablir la syntaxe correcte des règles de filtrage, il est possible d'utiliser sed avec une instruction du type `$ sed '/^[ \-].*\$/N;s/\\n *//' dump.iptables` où le fichier `dump.iptables` contient la copie d'écran ci-dessus.

Non seulement la solution présentée ci-dessus s'est montrée très efficace ces dernières années, mais elle a le mérite de ne pas faire intervenir un outil tiers ; ce qui diminue le coût d'administration.

## 6. Règles de filtrage sur le routeur central (*hub*)

Suivant le cahier des charges fixé, le routeur central doit autoriser le trafic issu du poste client sur son interface WAN et le router sur l'interface LAN.

Tout comme dans le cas du routeur d'agence, on utilise le jeu de règles communes que l'on complète avec les besoins spécifiques à la configuration d'un routeur qui doit faire transiter le trafic d'une interface sur l'autre.

À la différence du routeur d'agence, le routeur central maîtrise l'attribution des adresses IP. On peut donc inclure le contrôle des adresses IP sources dans les règles de filtrage réseau.

### 1. Le jeu de règles communes est-il suffisant pour que le poste se comporte comme un routeur ?

Identifier les conditions nécessaires pour que la fonction routage du noyau soit active et que le filtrage réseau autorise le transit de l'interface WAN vers l'interface LAN.

Non. Il manque au moins 2 conditions pour que le routage et la traduction d'adresses sources soient actifs.

- Pour qu'un paquet soit transmis d'une interface réseau vers une autre, il faut s'assurer que le routage est actif au niveau du noyau. Cette fonction est paramétrée par la variable d'état `ip_forward` du système de fichiers virtuel `/proc`. La valeur 1 indique que la fonction routage est active dans le noyau :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Comme la politique par défaut sur la chaîne `FORWARD` est `DROP`, aucun paquet ne peut traverser les règles de filtrage et transiter d'une interface vers l'autre. Sans règle supplémentaire, les tests ICMP doivent incrémenter le compteur `DROP` de la chaîne `FORWARD`.

### 2. Quelle est la syntaxe de la commande `iptables` qui autorise le transfert des paquets entrant par l'interface WAN vers l'interface LAN ?

Il faut implanter deux règles dans la chaîne `FORWARD`. Une première règle qui correspond à ce qui a déjà été vu dans la mise au point du jeu de règles communes pour les chaînes `INPUT` et `OUTPUT` : tout trafic relatif à une demande enregistrée dans la machine d'état de suivi de communication est accepté. Une seconde règle qui accepte les paquets entrants par l'interface LAN en enregistrant les nouvelles communications dans la même machine d'état. On obtient le jeu de règles suivant :

```

# cat iptables.router
#-----
# T a b l e   F I L T E R
#-----
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----
# I N P U T
#-----
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -m limit --limit 5/sec -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----
# F O R W A R D
#-----
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i ipp0 -s 192.168.96.0/20 -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
    
```

```
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
```

3. Après avoir initié des communications avec les différents protocoles usuels (ICMP, UDP et TCP), relever l'état des communications du routeur d'agence distant avec l'outil `iptables`.

4. Est-il possible de visualiser à l'aide de l'analyseur réseau `Wireshark` le trafic retour relatif aux requêtes émises par le client WAN ?

Non. Pour que le trafic retour aboutisse sur l'interface du client WAN, il faudrait que la route vers le réseau étendu soit connue du reste de l'Internet.

5. Sans protocole de routage dynamique assurant la publication de la route vers le réseau étendu sur l'Internet, quelle est la solution technique à utiliser pour que les postes clients distants puissent accéder aux autres réseaux ?

C'est la traduction d'adresse source qui permet d'utiliser l'adresse IP de l'interface LAN du routeur comme la seule interface visible de l'Internet.

6. Quelle est la syntaxe de la règle d'implantation de la traduction d'adresses IP source en sortie de l'interface LAN du routeur central ?

```
# cat iptables.router
#-----
# T a b l e  N A T
#-----
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
#-----
# T a b l e  F I L T E R
#-----
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----
# I N P U T
#-----
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -m limit --limit 5/sec -m state --state NEW -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----
# F O R W A R D
#-----
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i ipp0 -s 192.168.96.0/20 -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -m state --state NEW -j ACCEPT
COMMIT
```

## 7. Règles de filtrage avec identification des protocoles

Pour les deux configurations étudiées ci-avant, aucune distinction de protocole n'a été effectuée. Pour affiner le processus d'enregistrement et de suivi des communications réseau, il est possible de distinguer les caractéristiques de chacun des protocoles autorisés.

### 7.1. Protocole ICMP

Le protocole ICMP décrit dans le document standard *RFC792 Internet Control Message Protocol*<sup>10</sup> est une pièce essentielle du modèle TCP/IP. Il est principalement utilisé pour rapporter les conditions d'erreurs sur les réseaux. Cependant, les caractéristiques actuelles du protocole ne recommandent aucun contrôle de validation sur les messages d'erreur reçus. Ce protocole laisse donc la porte ouverte à une grande variété d'attaques qui peuvent être effectuées contre TCP à l'aide de messages ICMP. Ces attaques comprennent la réinitialisation de connexion, la réduction du débit de sortie, les dégradations de performances. Toutes ces attaques peuvent être réalisées depuis des réseaux distants, sans la nécessité d'analyser les paquets qui correspondent à la connexion TCP attaquée.

Alors que les implications sur la sécurité du protocole ICMP sont connues depuis longtemps, tous les systèmes n'ont pas mis en application des contrôles de validation sur les messages d'erreur reçus pour réduire au minimum l'impact de ces attaques.

<sup>10</sup> <http://www.faqs.org/rfcs/rfc792.html>

Au niveau du noyau Linux, les responsables du sous-système réseau ont décidé de ne plus traiter les messages de type 4 source-querch.

On dispose des ressources suivantes pour débiter l'étude du protocole ICMP.

- La liste des types de messages ICMP est enregistrée par l'*Internet Assigned Numbers Authority* (IANA) : [ICMP parameters](#)<sup>11</sup>.
- Le [Tutoriel iptables](#)<sup>12</sup> contient une section complète de présentation des caractéristiques du protocole ICMP.

## 7.2. Règles de filtrage communes à toutes les configurations

- Avec le protocole TCP, il est possible d'identifier les phases d'établissement, de maintien et de libération de connexion.
- Avec le protocole UDP, il n'y a pas grand chose à identifier puisque ce protocole n'est pas orienté connexion et que le nombre des champs de l'en-tête est très limité.

1. Quelle est la syntaxe d'appel de la commande **iptables** qui permet d'afficher la liste des messages ICMP et leurs types connus du système de filtrage réseau ?

Après avoir recherché le mot clé `icmp` dans les pages de manuels de la commande **iptables**, on obtient l'instruction suivante : `# iptables -p icmp -h`.

2. Quelles sont les modifications à apporter sur le jeu de règles communes pour distinguer les messages ICMP les plus importants ?

On considère quatre types de messages ICMP :

- Type de message 8 : `echo-request` : on autorise les nouvelles requêtes *ping* à raison de 5 par seconde.
- Type de message 0 `echo-reply` : on autorise les réponses *pong* aux requêtes *ping* enregistrées dans la machine d'état de suivi de communication.
- Type de message 3 : `destination-unreachable` : on autorise toutes les notifications d'erreur sur la destination relatives à une demande émise à partir de ce système.
- Type de message 11 : `time-exceeded` : on autorise toutes les notification de débordement de temps relatives au trafic émis à partir de ce système.

```
# cat iptables.common
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----
# I N P U T
#-----
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/s -m state --state NEW -j ACCEPT
-A INPUT -p icmp --icmp-type echo-reply -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p icmp --icmp-type destination-unreachable -m state --state RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type time-exceeded -m state --state RELATED -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
COMMIT
```

3. Quelles sont les modifications à apporter sur le jeu de règles communes pour distinguer les conditions sur les connexions TCP ?

On distingue les demandes d'ouverture de connexion avec l'option `--syn` des connexions déjà établies avec l'option inverse ! `--syn`.

```
# cat iptables.common
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----
# I N P U T
#-----
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/s -m state --state NEW -j ACCEPT
```

<sup>11</sup> <http://www.iana.org/assignments/icmp-parameters>

<sup>12</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

```

-A INPUT -p icmp --icmp-type echo-reply -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p icmp --icmp-type destination-unreachable -m state --state RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type time-exceeded -m state --state RELATED -j ACCEPT
-A INPUT -p tcp ! --syn -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p tcp --syn -m state --state RELATED -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
COMMIT
    
```

4. Quelles sont les modifications à apporter sur le jeu de règles communes pour distinguer le protocole UDP ?

```

# cat iptables.common
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
#-----
# I N P U T
#-----
-A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/s -m state --state NEW -j ACCEPT
-A INPUT -p icmp --icmp-type echo-reply -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p icmp --icmp-type destination-unreachable -m state --state RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type time-exceeded -m state --state RELATED -j ACCEPT
-A INPUT -p tcp ! --syn -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p tcp --syn -m state --state RELATED -j ACCEPT
-A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
COMMIT
    
```

## 8. Documents de référence

### 8.1. IETF & IANA

*Types de messages ICMP*

L'Internet Assigned Numbers Authority a enregistré les types de messages ICMP à la page [ICMP parameters](#)<sup>13</sup>.

### 8.2. Distribution Debian GNU/Linux

*Manuel de référence Debian*

[Manuel de référence Debian : configuration du réseau](#)<sup>14</sup> : chapitre du manuel de référence Debian consacré à la configuration réseau.

### 8.3. Site inetdoc.net

*Configuration d'une interface de réseau local*

[Configuration d'une interface de réseau local](#)<sup>15</sup> : identification du type d'interface, de ses caractéristiques et manipulations des paramètres. Ce support fournit une méthodologie de dépannage simple d'une connexion réseau.

*Fonctions réseau du noyau Linux*

[Configuration des fonctions réseau & compilation du noyau Linux](#)<sup>16</sup> : présentation et configuration des fonctions réseau du noyau LINUX

*Configuration des fonctions réseau & compilation du noyau LINUX*

[Configuration des fonctions réseau & compilation du noyau Linux](#)<sup>17</sup> : travaux pratiques sur la préparation d'un système routeur GNU/Linux. Compilation d'un noyau LINUX à partir de ses sources après avoir passé en revue ses fonctions réseau et sélectionné les pilotes de périphériques nécessaires.

*Didacticiel sur Iptables*

[Tutoriel iptables](#)<sup>18</sup> : guide très complet sur le fonctionnement du filtrage réseau avec les noyaux Linux.

<sup>13</sup> <http://www.iana.org/assignments/icmp-parameters>

<sup>14</sup> <http://www.debian.org/doc/manuals/debian-reference/ch05.fr.html>

<sup>15</sup> [http://www.inetdoc.net/travaux\\_pratiques/config.interface.lan/](http://www.inetdoc.net/travaux_pratiques/config.interface.lan/)

<sup>16</sup> [http://www.inetdoc.net/travaux\\_pratiques/interco.kernel.q/](http://www.inetdoc.net/travaux_pratiques/interco.kernel.q/)

<sup>17</sup> [http://www.inetdoc.net/travaux\\_pratiques/interco.kernel.q/](http://www.inetdoc.net/travaux_pratiques/interco.kernel.q/)

<sup>18</sup> <http://www.inetdoc.net/guides/iptables-tutorial/>

*Guide Pratique du NAT*

*Guide Pratique du NAT*<sup>19</sup> : Ce document décrit comment réaliser du camouflagement d'adresse IP, un serveur mandataire transparent, de la redirection de ports ou d'autres formes de Traduction d'adresse réseau (*Network Address Translation* ou NAT) avec le noyau Linux 2.4.

---

<sup>19</sup> <http://www.netfilter.org/documentation/HOWTO/fr/NAT-HOWTO.html>