

# Introduction au filtrage réseau

Philippe Latu

philippe.latu(at)linux-france.org

<http://www.linux-france.org/prj/inetdoc/>

Historique des versions		
\$Revision: 1112 \$	\$Date: 2007-04-21 16:52:00 +0200 (sam, 21 avr 2007) \$	\$Author: latu \$
Année universitaire 2006-2007 / Draft en cours de rédaction		
Résumé		
<p>L'objectif de ce support de travaux pratiques est d'étudier la mise en oeuvre du filtrage réseau dans le contexte routeur d'agence et client distant. Comme dans les autres supports de travaux pratiques de cette série, on assimile ces configurations types à des interconnexions entre réseaux locaux et réseau étendus. Les questions de ce support sont présentées comme une introduction pas à pas au filtrage réseau. On débute avec les outils, on développe les fonctions de suivi de communication (stateful inspection) sur une interface, puis on applique ce filtrage au routage avec traduction d'adresse (NAT).</p>		

## Table des matières

1. Copyright et Licence .....	2
1.1. Méta-information .....	2
1.2. Conventions typographiques .....	2
2. Architecture réseau étudiée .....	3
2.1. Topologie type .....	3
2.2. Plan d'adressage WAN .....	4
3. Les outils de filtrage réseau .....	4
3.1. Questions sur iptables .....	4
3.2. Questions sur netfilter .....	6
4. Règles de filtrage communes à toutes les configurations .....	7
5. Règles de filtrage sur le poste client distant .....	10
6. Règles de filtrage sur le routeur d'accès .....	12
7. Règles de filtrage avec identification des protocoles .....	14
7.1. Protocole ICMP .....	14
7.2. Règles de filtrage communes à toutes les configurations .....	14
8. Documents de référence .....	16
8.1. IETF & IANA .....	16
8.2. Distribution Debian GNU/Linux .....	16
8.3. Projet [inetdoc.LINUX] .....	16

# 1. Copyright et Licence

Copyright (c) 2000,2007 Philippe Latu.  
 Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 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) 2000,2007 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 1.2 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 aux formats PDF et Postscript : [interco.netfilter.pdf](#)<sup>3</sup> | [interco.netfilter.ps.gz](#)<sup>4</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
- iptables - Administration tools for packet filtering and NAT

## 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.

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

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

<sup>3</sup> <http://www.linux-france.org/prj/inetdoc/telechargement/interco.netfilter.pdf>

<sup>4</sup> <http://www.linux-france.org/prj/inetdoc/telechargement/interco.netfilter.ps.gz>

## 2. Architecture réseau étudiée

### 2.1. Topologie type

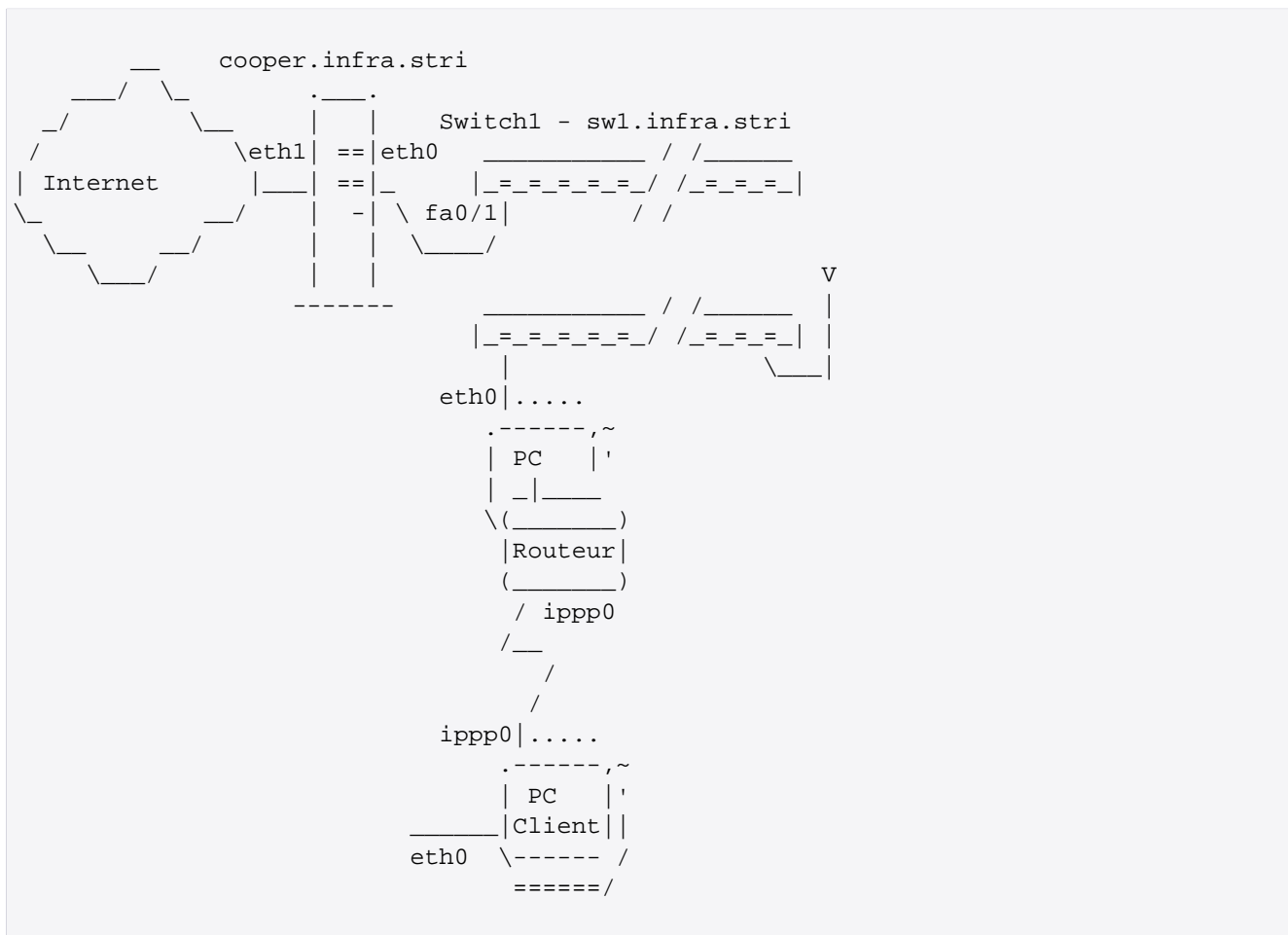
La topologie étudiée associe 2 postes avec 2 rôles distincts.

#### Poste client distant

Ce poste dispose d'un accès Internet sur son interface WAN RNIS. Il offre une connexion sur un réseau local domestique (PAN) via une interface LAN Ethernet.

#### Routeur serveur d'accès

Ce poste réalise l'interconnexion LAN/WAN. Il offre un accès Internet au poste client via son interface WAN RNIS. Il obtient son propre accès Internet via son interface LAN Ethernet.



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

À un premier niveau, le *routeur d'accès* doit s'assurer que le trafic réseau qu'il route vers l'Internet provient bien de son poste «client» auquel il a attribué une adresse IP via PPP. À un second niveau, le routeur serveur d'accès doit limiter le volume de trafic ICMP pour éviter les éventuels dénis de services liés à ce protocole. Il peut aussi intercepter toutes les requêtes DNS du poste client en exploitant un service de type *cache only* pour éviter les falsifications de réponses.

À un premier niveau, le *poste client distant* 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. À un second niveau, le poste client doit «masquer» son réseau local PAN 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.

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 *Configurations routeur d'agence & serveur d'accès*<sup>5</sup>.

**Tableau 1. Plan d'adressage liaisons WAN**

Poste routeur NAS	bus	N° tél.	Adresses IP serveur:client	N° tél.	Poste Client distant
asterix	S0.1	104	192.168.104.1:192.168.104.2	105	obelix
tintin2	S0.2	106	192.168.106.1:192.168.106.2	107	haddock
dupond	S0.3	108	192.168.108.1:192.168.108.2	109	tif
hochet	S0.4	110	192.168.110.1:192.168.110.2	111	blake
jourdan	S0.5	112	192.168.112.1:192.168.112.2	113	mortimer
dupont	S0.6	114	192.168.114.1:192.168.114.2	115	danny

## 3. Les outils de filtrage réseau

Sur un système Debian 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*, 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 basiques de manipulation des fonctions de filtrage réseau ?

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.

```
# apt-cache search iptables |grep iptables
firehol - An easy to use but powerful iptables stateful firewall
ipac-ng - IP Accounting for iptables (kernel >=2.4)
ipkungfu - iptables-based Linux firewall
iptables - administration tools for packet filtering and NAT
iptables-dev - development files for iptable's libipq and libiptc
iptstate - Top-like state for netfilter/iptables
kmyfirewall - iptables based firewall configuration tool for KDE
libiptables-ipv4-ipqueue-perl - Perl extension for libipq
reaim - Enable AIM and MSN file transfer on Linux iptables based NAT
uif - Advanced iptables-firewall script
uruk - Very small firewall script, for configuring iptables
```

Pour ces travaux pratiques, 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.

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

<sup>5</sup> <http://www.linux-france.org/prj/inetdoc/cours/interco.ppp.tp/>

```
<snipped/>
```

2. Quelles sont les options de la commande `iptables` qui permettent de visualiser les règles de filtrage réseau actives ainsi que les compteurs correspondants ?

En consultant les pages de manuels via `# man iptables`, on relève les 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'adresse 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 ?

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.

- Consultation de la table `netfilter` :

```
# lsmod |less
Module                Size  Used by
iptables_filter       3200  0
ip_tables              14536  1 iptables_filter
x_tables              14852  1 ip_tables
```

- Consultation de la table `nat` :

```
# lsmod |less
Module                Size  Used by
iptables_nat          7940  0
ip_nat                17836  1 iptables_nat
ip_contrack           53684  2 iptables_nat,ip_nat
nfnetlink              6808  2 ip_nat,ip_contrack
iptables_filter       3200  0
ip_tables              14536  2 iptables_nat,iptables_filter
x_tables              14852  2 iptables_nat,ip_tables
```

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

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

```
# uname -r
2.6.18.3
$ find /lib/modules/`uname -r` -type d -name netfilter
/lib/modules/2.6.18.3/kernel/net/bridge/netfilter ❶
/lib/modules/2.6.18.3/kernel/net/ipv4/netfilter ❷
/lib/modules/2.6.18.3/kernel/net/ipv6/netfilter ❸
/lib/modules/2.6.18.3/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`. Le chargement des trois premiers modules entraîne la création des entrées relatives aux noms chaînes, aux correspondances et aux prises de décisions.

```
# find /proc -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 -name "*contrack*"
/proc/sys/net/ipv4/ip_contrack_max
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_max_retrans
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_be_liberal
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_loose
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_max_retrans
/proc/sys/net/ipv4/netfilter/ip_contrack_log_invalid
/proc/sys/net/ipv4/netfilter/ip_contrack_generic_timeout
/proc/sys/net/ipv4/netfilter/ip_contrack_icmp_timeout
/proc/sys/net/ipv4/netfilter/ip_contrack_udp_timeout_stream
/proc/sys/net/ipv4/netfilter/ip_contrack_udp_timeout
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_close
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_time_wait
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_last_ack
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_close_wait
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_fin_wait
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_established
/proc/sys/net/ipv4/netfilter/ip_contrack_tcp_timeout_syn_recv
```

```

/proc/sys/net/ipv4/netfilter/ip_conntrack_tcp_timeout_syn_sent
/proc/sys/net/ipv4/netfilter/ip_conntrack_checksum
/proc/sys/net/ipv4/netfilter/ip_conntrack_buckets
/proc/sys/net/ipv4/netfilter/ip_conntrack_count
/proc/sys/net/ipv4/netfilter/ip_conntrack_max
/proc/net/ip_conntrack_expect
/proc/net/ip_conntrack
/proc/net/stat/ip_conntrack

```

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

Les états sont directement consultables à partir du fichier virtuel `/proc/net/ip_conntrack`. Par exemple :

```

# cat /proc/net/ip_conntrack
<snipped/>
udp      17 118 src=172.16.80.4 dst=172.16.80.1 sport=34100 dport=53 \
        packets=2 bytes=124 src=172.16.80.1 dst=172.16.80.4 sport=53 \
        dport=34100 packets=2 bytes=300 [ASSURED] mark=0 secmark=0 use=1

```

L'exemple ci-dessus donne l'état du suivi de communication d'une requête DNS entre un poste client avec l'adresse `172.16.80.4` et le port source `34100` et un serveur avec l'adresse `172.16.80.1`.

La section 7.2 *Les entrées de conntrack* du *Didacticiel sur Iptables* 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`.

## 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 oeuvre 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 possible le premier paquet qui doit être enregistré dans la table de suivi de communication*. Cette règle de description du premier paquet doit toujours 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 les 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`. Du point de vue syntaxique, on applique les commandes suivantes :

```

# 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. Du point de vue syntaxique on applique :

```
# 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. Autre intérêt, l'affichage est plus condensé.

```
# iptables-save > iptables.common
# cat iptables.common
*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
#~~~~~
# O U T P U T
#~~~~~
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
```

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 ?

Non. La politique par défaut sur les chaînes INPUT et OUTPUT étant positionnée à DROP, tout nouveau paquet arrivant ou sortant est jeté. 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, il est préférable 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 l'interface. Tous les autres paquets devront correspondre aux règles déjà écrites ci-dessus. Le fichier de règles devient :

```
# cat iptables.common
*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 -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 -i lo -m state --state NEW -j ACCEPT
COMMIT

# iptables-restore <iptables.common

```

À partir de ce jeu de règles, il est possible de communiquer avec l'interface de boucle locale `lo`. 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 ICMP en entrée et en sortie du système en limitant le nombre des nouvelles requêtes à 5 par seconde ?

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.

```

# cat iptables.common
*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/s -m state --state NEW -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 -p ICMP -m limit --limit 5/s -m state --state NEW -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
COMMIT

# iptables-restore <iptables.common

```

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 de la commande **ping** sur un hôte distant.

```

# ping -n -c 3 192.168.1.7
PING 192.168.1.7: 56 data bytes
64 bytes from 192.168.1.7: icmp_seq=0 ttl=64 time=1.8 ms
64 bytes from 192.168.1.7: icmp_seq=1 ttl=64 time=1.5 ms
64 bytes from 192.168.1.7: icmp_seq=2 ttl=64 time=1.6 ms

--- topaze.linux.home ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.5/1.6/1.8 ms
# ping -f -n -c 10 192.168.1.7

```

```
PING 192.168.1.7: 56 data bytes
.....
--- topaze.linux.home ping statistics ---
106 packets transmitted, 10 packets received, 90% packet loss
round-trip min/avg/max = 1.3/1.7/2.7 ms
```

On constate que le premier test ne produit aucune erreur (ie. l'administration réseau est correcte) alors que la tentative élémentaire de saturation engendre des pertes de paquets ICMP.

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 client distant

Suivant le cahier des charges fixé, la seule contrainte imposée au poste client distant est de s'assurer que le trafic entrant sur son interface WAN est bien relatif à une demande émise via cette même interface. De plus, si ce poste doit jouer le rôle de routeur domestique, on doit compléter le filtrage avec une traduction d'adresse source aussi liée à l'interface WAN.

### 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. On obtient donc :

```
# cat iptables.client
#-----
# 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
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p icmp -m limit --limit 5/sec -m state --state NEW -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ipp0 -m state --state NEW -j ACCEPT
COMMIT
```

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

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

Relativement à la question précédente, on prépare la transformation du poste client distant en «routeur» SOHO qui doit servir de passerelle à un petit réseau domestique. On ajoute 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**.

```
# cat iptables.client
#-----
# T a b l e   N A T
#-----
```

```

*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o ipp0 -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
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p icmp -m limit --limit 5/sec -m state --state NEW -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 une passerelle de traduction d'adresses IP sources ?

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`.

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

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 communication dans la même machine d'état. On obtient le jeu de règles suivant :

```

# cat iptables.client
#-----
# T a b l e   N A T
#-----
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o ipp0 -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 eth0 -m state --state NEW -j ACCEPT
#-----
# O U T P U T
#-----
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p icmp -m limit --limit 5/sec -m state --state NEW -j ACCEPT
-A OUTPUT -o lo -m state --state NEW -j ACCEPT
-A OUTPUT -o ippp0 -m state --state NEW -j ACCEPT
COMMIT

```

## 6. Règles de filtrage sur le routeur d'accès

Suivant le cahier des charges fixé, le routeur d'accès 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 poste client distant, 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 poste client distant, le routeur d'accès 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 ?

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 communication 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 ippp0 -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 -p icmp -m limit --limit 5/sec -m state --state NEW -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 poste client distant avec l'outil iptstate.
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 qui publie la route vers le réseau étendu sur l'Internet, quelle est la solution technique pour que les postes clients distants puissent accéder au réseau public ?

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. On obtient le jeu de règles suivant :

```

# 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 ippp0 -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 -p icmp -m limit --limit 5/sec -m state --state NEW -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.

Pour traiter les questions suivantes, on s'appuie sur les publication du groupe de travail de l'IETF : *TCP Maintenance and Minor Extensions*.

### 7.1. Protocole ICMP

Le protocole ICMP décrit dans le document standard *RFC792 Internet Control Message Protocol*<sup>6</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.

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

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) : *Types de messages ICMP*.
- Le *Didacticiel sur Iptables* 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 ?

<sup>6</sup> <http://www.faqs.org/rfcs/rfc792.html>

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
-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>7</sup>.

#### TCP Maintenance and Minor Extensions

[TCP Maintenance and Minor Extensions](#)<sup>8</sup> : groupe de travail à l'origine de nombreux *drafts* de préconisations de sécurité et d'évolutions autour de la pile de protocoles TCP/IP.

### 8.2. Distribution Debian GNU/Linux

#### Debian Reference Chapter 10 - Network configuration

[Debian Reference Chapter 10 - Network configuration](#)<sup>9</sup> : chapitre du manuel de référence *Debian* consacré à l'administration réseau.

### 8.3. Projet [inetdoc.LINUX]

#### Configuration d'une interface de réseau local

[Configuration d'une interface de réseau local](#)<sup>10</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

[Fonctions réseau du noyau Linux](#)<sup>11</sup> : présentation et configuration des fonctions réseau du noyau LINUX

<sup>7</sup> <http://www.iana.org/assignments/icmp-parameters>

<sup>8</sup> <http://tools.ietf.org/wg/tcpm/>

<sup>9</sup> <http://www.debian.org/doc/manuals/reference/ch-gateway.en.html>

<sup>10</sup> <http://www.linux-france.org/prj/inetdoc/cours/config.interface.lan/>

<sup>11</sup> <http://www.linux-france.org/prj/inetdoc/cours/interco.noyau/>

*Configuration des fonctions réseau & compilation du noyau LINUX*

*Configuration des fonctions réseau & compilation du noyau LINUX*<sup>12</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*

*Didacticiel sur Iptables*<sup>13</sup> : guide très complet sur le fonctionnement du filtrage réseau avec les noyaux Linux.

*Guide Pratique du NAT sous Linux 2.4*

*Guide Pratique du NAT sous Linux 2.4*<sup>14</sup> : Ce document décrit comment réaliser du camouflage 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>12</sup> <http://www.linux-france.org/prj/inetdoc/cours/interco.noyau.tp/>

<sup>13</sup> <http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/>

<sup>14</sup> <http://www.linux-france.org/prj/inetdoc/guides/NAT-HOWTO/>