

Les firewalls libres : netfilter, IP Filter et Packet Filter

*Linux Expo Paris
1er Février 2002*

Jean-Baptiste Marchand

Jean-Baptiste.Marchand@hsc.fr

Hervé Schauer Consultants



Firewalls

□ Firewalls

- ★ Logiciel de contrôle d'accès au niveau réseau
- ★ Fonctionnalité principale : filtrage IP
- ★ Fonctionnalité supplémentaire souvent présente : traduction d'adresses et de ports

□ Filtrage IP

- ★ Idée : contrôler les paquets IP autorisés à atteindre un hôte
- ★ Intérêt : sécuriser un hôte de façon globale (au niveau réseau)

□ Traduction d'adresses et de ports

- ★ Idée : récrire les en-têtes des paquets
- ★ Intérêt : faire face à la pénurie d'adresses routables sur l'Internet

Firewalls libres

netfilter

- ★ Filtre de paquets du noyau Linux 2.4
- ★ Successeur d'IPchains (Linux 2.2)
- ★ Conçu par Paul "Rusty" Russell
- ★ Développé par un noyau dur de 5 personnes
- ★ Contributions pour des améliorations ou corrections

IP Filter

- ★ Filtre de paquets fonctionnant sous Unix libres et propriétaires
- ★ Intégré dans FreeBSD et NetBSD
- ★ Conçu et développé par Darren Reed

Packet Filter

- ★ Filtre de paquets dans OpenBSD (à partir de la version 3.0)
- ★ Conçu par Daniel Hartmeier
- ★ Développé par l'équipe d'OpenBSD

Filtrage IP standard

□ Critères

- ★ Interface réseau : entrée ou sortie
- ★ Adresses IP (source et destination) : hôte ou sous-réseau
- ★ Champs de l'en-tête IP
 - ▷ Fragmentation
 - ▷ TOS et TTL
 - ▷ Options IP
- ★ Protocoles de niveau 4 : TCP, UDP et ICMP
 - ▷ Ports source et destination (TCP et UDP)
 - ▷ Drapeaux (TCP)
 - ▷ Types et codes (ICMP)

□ Actions

- ★ Laisser passer
- ★ Bloquer (façon silencieuse)
- ★ Rejeter (message ICMP ou segment TCP avec drapeau RST)

Filtrage à états

□ Principe

- ★ Filtrage dynamique, en conservant des états pour les communications en cours
- ★ Seuls des paquets correspondants à un état pré-existant sont acceptés

□ Intérêt

- ★ Simplifie l'écriture des règles de filtrage
- ★ Améliore la sécurité, en n'autorisant que le trafic effectivement licite

□ Protocoles

- ★ TCP
 - ▷ Segments appartenant à une connexion TCP en cours
- ★ UDP
 - ▷ Datagrammes en réponse à un datagramme UDP émis
 - ▷ Messages ICMP d'erreur
- ★ ICMP
 - ▷ Messages ICMP en réponse à un message ICMP émis

Filtrage à états : mise en oeuvre

□ Mise en oeuvre

- ★ Création d'un état lors de la traversée du premier paquet
- ★ Mémorisation de paramètres identifiant de façon unique une communication
- ★ Validation des paquets par comparaison des états courants
- ★ Expiration des états après un temps paramétrable

□ Paramètres conservés

- ★ Adresses source et destination
- ★ Ports source et destination
- ★ Type, code, identifiant et numéro de séquence (ICMP)

□ TCP

- ★ S'assurer que des segments TCP font partie d'une connexion en cours est complexe
- ★ Real Stateful TCP Packet Filtering in Ipfiler
 - ▷ http://home.iae.nl/users/guido/papers/tcp_filtering.ps.gz

Traduction d'adresses et de ports

□ Traduction d'adresses

★ Uni-directionnelle

- ▷ Traduction en sortie d'adresses (typiquement) privées en adresse(s) publique(s)
- ▷ Possibilité de changer le port source

★ Bi-directionnelle

- ▷ Traduction d'**une** adresse (typiquement) publique en **une** adresse (typiquement) privée et réciproquement

□ Redirection de ports

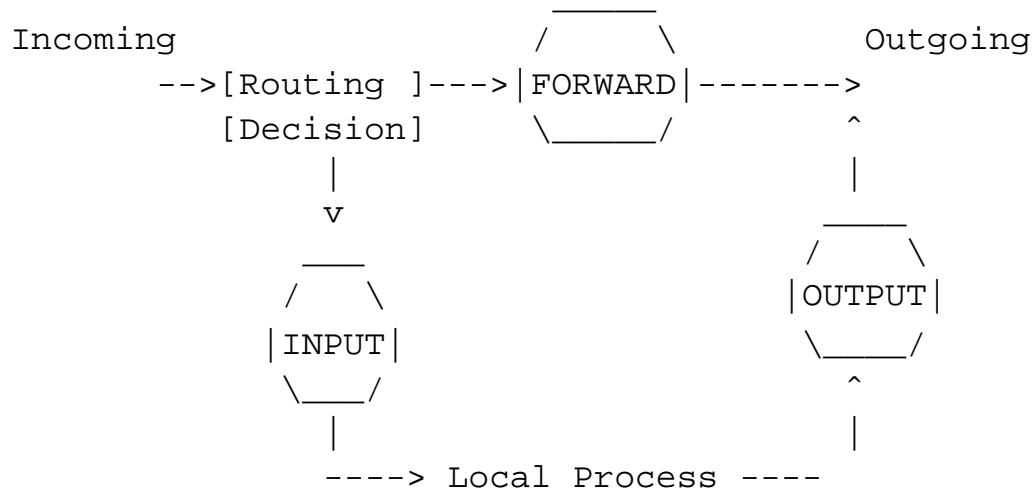
- ★ Redirection d'un port en entrée vers un autre, en modifiant l'adresse de destination ou non

□ Mise en oeuvre

- ★ Fonctionnalités présentes dans netfilter, IP Filter et Packet Filter
- ★ Ne sont pas détaillées ici

netfilter : chaînes

- Règles de filtrage placées dans des chaînes
 - ★ Chaîne : sous-programme
 - ★ Règles de filtrage : instructions
- Une des 3 chaînes standards est obligatoirement traversée :
 - ★ INPUT : paquets à destination de la machine locale
 - ★ OUTPUT : paquets émis par la machine locale
 - ★ FORWARD : paquets transitant par la machine locale



netfilter : règles de filtrage

Règles de filtrage

- ★ Selection du trafic
- ★ Cible à atteindre

Cibles

★ Classiques

- ▷ ACCEPT : paquet accepté
- ▷ DROP : paquet rejeté de façon silencieuse
- ▷ REJECT : paquet rejeté en informant l'émetteur
- ▷ RETURN
 - fin du parcours d'une chaîne utilisateur
 - cible par défaut dans les chaînes INPUT, OUTPUT ou FORWARD

★ Journalisation

- ▷ LOG : paquet journalisé

★ Extension du filtrage

- ▷ QUEUE
 - redirection du paquet vers une application en mode utilisateur
 - la décision de filtrage est prise par l'application

netfilter : critères de filtrage

- Critères de filtrage classiques
 - ★ IP, TCP, UDP, ICMP
- Spécification des interfaces réseau
 - ★ Chaîne INPUT : interface d'entrée
 - ★ Chaîne OUTPUT : interface de sortie
 - ★ Chaîne FORWARD : interfaces d'entrée et de sortie
- Critères de filtrages moins classiques
 - ★ Filtrage par adresses MAC
 - ★ Filtrage sur l'origine d'un paquet local
 - ▷ uid, gid, pid, sid du processus émetteur

netfilter : filtrage à états

- Protocoles TCP, UDP et ICMP
- Options permettant de comparer le trafic aux états
 - ★ NEW : création d'un nouvel état
 - ★ ESTABLISHED : paquet appartenant à une communication en cours
 - ★ RELATED : paquet lié à une communication en cours
 - ▷ Message ICMP d'erreur
 - ▷ Trafic lié au fonctionnement d'un protocole applicatif
 - ★ INVALID : paquet non-identifié parmi les communications en cours
- Stratégie
 - ★ Créer un état au début de communication avec NEW
 - ★ Accepter les paquets liés aux communications en cours
 - ▷ ESTABLISHED
 - ▷ RELATED

netfilter : écriture de règles

□ Règles

- ★ Options du programme iptables
- ★ Evaluation
 - ▷ Parcourues depuis le début de la chaîne
 - ▷ Arrêt du parcours en cas de correspondance d'une règle
 - ▷ Politique par défaut en fin des chaînes standards
 - ▷ Retour à la chaîne parente en fin des chaînes utilisateurs
- ★ Chaînes utilisateurs
 - ▷ Possibilité de créer de nouvelles chaînes
 - ▷ Améliore l'efficacité du filtrage
 - ▷ Regroupe des règles liées à un même type de trafic

□ Chaînes

- ★ Placer une politique par défaut pour les chaînes standards
- ★ Ajouter les règles en fin de chaîne

netfilter : exemple

```
IFACE=eth0
MYLAN=192.168.1.0
MYWKS=192.168.1.22
MYDNS=192.168.1.53
MYADDR=192.168.1.42

# vide les chaînes standards
iptables --flush

# chaînes standards bloquent par défaut
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Filtrage à états
iptables -A OUTPUT -i $IFACE -m state --state ESTABLISHED, RELATED -j ACCEPT
iptables -A INPUT -i $IFACE -m state --state ESTABLISHED, RELATED -j ACCEPT

# Administration par SSH
iptables -A INPUT -i $IFACE -p tcp -s $MYWKS --sport 1024:65535 -d $MYADDR --dport 22 \
-m state --state NEW -j ACCEPT

# Résolution DNS
iptables -A OUTPUT -i $IFACE -p udp -s $MYADDR --sport 1024:65535 -d $MYDNS --dport 53 \
-m state --state NEW -j ACCEPT

# Surveillance avec ping(1)
iptables -A OUTPUT -i $IFACE -p icmp -s $MYADDR --icmp-type echo-request -d $MYLAN \
-m state --state NEW -j ACCEPT
```

IP Filter : généralités

□ Fonctionnement

★ Ordre

1. Filtrage en entrée
2. Traitement TCP/IP du noyau
3. Filtrage en sortie

★ Schéma

- ▷ <http://coombs.anu.edu.au/~avalon/ipfil-flow.html>

□ Règles

★ Spécifiées dans un fichier de configuration

★ Syntaxe aisément compréhensible

★ Possibilité de créer des groupes de règles

- ▷ Regroupement de règles portant sur un même type de trafic
- ▷ Mot clés head et group

★ Deux jeux de règles possibles

- ▷ Actif
- ▷ Inactif

IP Filter : règles de filtrage

□ Évaluation des règles

- ★ Lues dans l'ordre du fichier
- ★ Celle qui s'applique est la **dernière**
- ★ Mot clé `quick` permet d'arrêter le parcours des règles

□ Filtrage à états

- ★ `keep frags`
 - ▷ autorise les fragments d'un paquet en cours de réassemblage
- ★ `keep state`
 - ▷ autorise les paquets appartenant à une communication en cours
- ★ Règles ne sont pas évaluées si le filtrage à états valide un datagramme

□ Journalisation

- ★ Option `log` au niveau d'une règle
- ★ Programme `ipmon(8)`, journalisation via `syslogd(8)`

IP Filter : exemple

ipf.conf

```
# Administration par SSH
pass in quick on xl0 proto tcp from 192.168.1.22 port > 1023 to 192.168.1.42 \
    port = 22 flags S/SA keep state

# Résolution DNS
pass out quick on xl0 proto udp from 192.168.1.42 port > 1023 to 192.168.1.53 \
    port = 53 keep state

# Administraton avec ping(1)
pass out quick on xl0 proto icmp from 192.168.1.42 icmp-type echo to 192.168.1.0/24 \
    keep state

block in on xl0 all
block out on xl0 all
```

Chargement des règles

★ `ipf -Fa -f /etc/ipf.conf`

Statistiques

★ `ipfstat`

Packet Filter : généralités

□ Historique

- ★ Nouveau filtre de paquets pour OpenBSD 3.0
- ★ Mêmes fonctionnalités qu'IP Filter
- ★ Maintenu par les développeurs de l'équipe d'OpenBSD

□ Fonctionnalités nouvelles

- ★ Normalisation des paquets
 - ▷ Protège les piles TCP/IP fragiles contre des paquets anormaux
 - ▷ Vérifie et supprime certaines anomalies dans les en-têtes
 - ▷ Réassemble les fragments
 - ▷ Action scrub
- ★ Renforcement des numéros de séquences initiaux (ISN) TCP
 - ▷ Pour des systèmes dont les ISNs sont prévisibles car trop peu aléatoires
 - ▷ Option `modulate state`
- ★ Ponts filtrants
 - ▷ Ponts Ethernet en standard sous OpenBSD : `bridge(4)`
 - ▷ Filtrage sur l'une des interfaces avec Packet Filter

Packet Filter : règles de filtrage

□ Règles

- ★ Syntaxe très proche de celle d'IP Filter
- ★ Sucre syntaxique pour écrire des règles de façon plus synthétique
 - ▷ Ensembles (*sets*) regroupant des listes d'attributs
 - ▷ Définition et expansion de variables
- ★ Même mode d'évaluation qu'IP Filter

□ Journalisation

- ★ Option `log` au niveau d'une règle
- ★ Paquets journalisés sur l'interface `pfllog0`
- ★ Journalisation au format `tcpdump(8)` avec `pfllogd(8)`

□ Différences avec IP Filter

- ★ Ordre des règles optimisée à la volée
 - ▷ Options `head` et `group` inutiles et non-supportées
- ★ Gestion des fragments
 - ▷ option `keep frags` remplacé par la directive `scrub`

Packet Filter : exemple

```
# Extrait de la section 6 de la FAQ d'OpenBSD
#
# http://www.openbsd.org/faq/faq6.html#PF

# Define useful variables
ExtIF="fxp0"           # External Interface
IntNet="1.1.1.0/24"    # Our internal network
NoRouteIPs="{ 127.0.0.1/8, 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }"
Services="{ www, https }"

# Clean up fragmented and abnormal packets
scrub in all

# don't allow anyone to spoof non-routeable addresses
block in quick on $ExtIF from $NoRouteIPs to any
block out quick on $ExtIF from any to $NoRouteIPs

# only allow our machines to connect via ssh
pass in quick on $ExtIF inet proto tcp from $IntNet to any port = 22

# allow others to use http and https
pass in quick on $ExtIF inet proto tcp from any to any port $Services flags S/SA

# finally lock the rest down with a default deny
block in quick on $ExtIF from any to any

# and let out-going traffic out and maintain state on established connections
pass out on $ExtIF from any to any keep state
```

Références : netfilter

- netfilter/iptables

- ★ <http://netfilter.samba.org>

- HOWTO

- ★ Networking Concepts HOWTO

- ★ Packet Filtering HOWTO

- ★ NAT HOWTO

- ★ <http://netfilter.samba.org/documentation/>

- Les caractéristiques de netfilter sous Linux 2.4

- ★ <http://www.hsc.fr/ressources/presentations/netfilter-2.4/netfilter.html.fr>

Références : IP Filter

- IP Filter
 - ★ <http://www.ipfilter.org>
- Pages de manuel FreeBSD et NetBSD
 - ★ ipf(5), ipf(8)
 - ★ <http://www.freebsd.org/cgi/man.cgi>
- IP Filter HOWTO
 - ★ <http://www.obfuscation.org/ipf/>
 - ★ Traduction française par IDEALX :
<http://openbsd.idealx.org/doc/index.fr.html>
- Fonctionnalités avancées (des règles) d'ipfilter
 - ★ <http://www.hsc.fr/ressources/breves/ipfilter.html>
- Présentation de IP Filter à la BSDCON 2000
 - ★ <http://www.madison-gurkha.com/publications/bsdcon2000/>

Références : Packet Filter

- OpenBSD Packet Filter
 - ★ <http://www.benzedrine.cx/pf.html>
- Pages de manuel OpenBSD
 - ★ pf.conf(5), pfctl(8)
 - ★ <http://www.openbsd.org/cgi-bin/man.cgi>
- Section 6 de la FAQ OpenBSD
 - ★ <http://www.openbsd.org/faq/faq6.html>
- The OpenBSD Packet Filter HOWTO
 - ★ <http://www.inebriated.demon.nl/pf-howto/>
 - ★ Traduction française par IDEALX :
<http://openbsd.idealx.org/doc/index.fr.html>