Sécurisez votre serveur sous *BSD avec PF

Cette page vous indiquera comment mettre en place un firewall de type PF sous OpenBSD mais les différences sont très mince pour faire la même chose sous NetBSD par exemple.

Introduction

Comme tout bon article qui se respecte, il faut une bonne accroche. Plutôt que de vous la faire à la façon d'un film où le suspense est intense, je vais aller droit au but. Vous trouverez dans cette article tout les éléments pour mettre en place de façon très simple PF sur votre OS préféré afin de mettre en route un serveur Apache Mysql php et verrez pourquoi j'aime tant ce firewall à la config très facile. De plus je me permet de signaler que ce logiciel a été la raison de la migration de ma passerelle personnel sous OpenBSD car je ne supportais plus la lourdeur de IPTABLES.
Bref, trêve de plaisanterie et passons à l'action !

Kesako PF ?

Si vous êtes tombé ici par hasard vous devez vous demander ce que c'est ce PF que vous ne trouvez pas dans vos dépot apt ou autre.
PF est l'initiale de Packet Filter, le firewall développé par OpenBSD et utilisé par NetBSD et FreeBSD. Il fut créé afin de remplacer IP Filter dans OpenBSD qui ne convenait plus aux attentes des développeurs. Il gère bien entendu le filtrage des paquets, les NAT (Network Address Translation), la redirection de port, la normalisation des paquet, QoS (Quality of Service) et plein d'autre chose. De plus il possède une syntaxe de configuration simple et efficace.

La syntaxe

Avant tout je vais commencer par la description de la syntaxe du fichier de configuration de PF qui se trouve dans `/etc/pf.conf`

Les macros

Nous allons commencer par le plus simple : les macros.

Les macros permettent d'enregistrer un certain nombre d'information dans une variable.

```
ext_if = "fxp0"
```

On appellera la macro avec $ext_if qui sera traduit automatiquement en fxp0 par PF.

Les listes

Parfois on veut créer une même règle avec seulement un élément qui change. Les listes sont la pour éviter de taper 25 fois la même ligne. Elle prenne la forme de `{ Element1, Element2, Element3, etc }`.
Un exemple concret d'utilisation d'une liste avec des adresse IP :

```
{ 192.168.0.1, 192.168.0.12, 192.168.0.56, 192.168.40.5 }
```

En utilisant cette liste dans une ligne de la configuration, la ligne sera interpréter par PF plusieurs fois avec chaque élément de la liste comme argument.
Cela donnera si on l'utilise avec `block in from [IP]` qui sert a bloquer tout le trafique entrant venant de [IP] :

```
block in from { 192.168.0.1, 192.168.0.12, 192.168.0.56, 192.168.40.5 }
```

Sera interprété comme :

```
block in from 192.168.0.1
block in from 192.168.0.12
block in from 192.168.0.56
block in from 192.168.40.5
```

Les règles de base

Certaines règles vont être énuméré ici afin de mieux comprendre la suite.

block

La règle `block` sert a bloquer les paquets.
Exemple :

```
block in on fxp0 proto tcp from any to any port ssh
```

Cette ligne se traduit par : bloque les paquets entrant sur l'interface `fxp0` utilisant le protocole `tcp` venant de partout (n'importe quel adresse réseau) et allant partout (n'importe quel adresse réseau) sur le port `ssh` (soit le port 22). Maintenant relisez l'exemple et la phrase précédente en même temps pour vous rendre compte de la grande clarté de la syntaxe de PF.

pass

La règle `pass` est l'inverse de `block` : elle sert a laisser passer les paquets.
Exemple :

```
pass out on fxp0 proto tcp from 192.168.0.1 to 192.168.0.0/24 port www
```

Cette ligne veux dire : laisse passer les paquet sortant sur l'interface `fxp0` utilisant le protocole `tcp` venant de l'adresse IP `192.168.0.1` et allant sur la plage réseau `192.168.0.0/24` sur le port `80`.

Passons à la pratique

A ce stade vous en savez déjà assez pour comprendre ce jeu de règle très simple que l'on peut utiliser dans `/etc/pf.conf`.

```
ext_if = "fxp0"
tcp_port = "{ 20, 21, 80, 443 }


# Ne pas filtrer sur l'interface de bouclage
set skip on lo0

# Mise en place d'une politique d'interdiction par défaut.
block in all

# Règle pour laisser passer les connections entrantes vers le serveur web et
ftp.
pass in on $ext_if proto tcp from any to any port $tcp_port
```

Et si on compliquait un peu

Et oui l'exemple précédent laisse passer les paquets sur les ports que l'on désir et bloque le reste mais est ce réellement suffisant ?
Et bien non ! En effet, un firewall qui ne ferait que bloquer le trafic sur les ports qui ne doivent pas être accessible ne sécurise en rien les ports qui eux sont ouvert. C'est pourquoi je vais ajouter ici la notion de filtrage des paquets. Non ne partez pas vous verez ce n'est pas sorcier.

Normalisation des paquets

Une attaque courante est d'envoyer des paquets erronés à un serveur pour le faire planter ou pire encore. PF permet de filtrer cela pour que les paquet reçu soit toujours bon même s'ils ne l'étaient pas forcement au départ.
Pour cela rien de plus simple il suffit de mettre :

```
scrub in all
```

Je suis sur que vous aussi vous allez y arriver vous verrez c'est pas si dur que ça.

Usurpation d'adresse

Ce type d'attaque bien connu fut rendu célèbre par Kevin Mitnick le 25 décembre 1994. Elle consiste a faire croire à l'ordinateur cible qu'on est quelqu'un d'autre en utilisant une adresse qui n'est pas la notre.
Pour s'en protéger une simple ligne suffit :

```
antispoof for fxp0 inet
```

Traduction de la ligne : Contre l'usurpation d'adresse (`antispoof`) pour l'interface `fxp0` en IPv4 (`inet` mais on peut aussi utiliser `inet6` pour l'IPv6).

Priorité sur une règle

PF lis sa configuration du haut vers le bas. Toute règle se trouvant après une autre en cas de conflit à donc la priorité. Par exemple établissons ce jeu de règle simple :

```
block in all
pass in on fpx0 proto tcp from any to any port 25
```

La première règle dit de tout bloquer mais la seconde dit de laisser passer le port 25 en tcp sur l'interface `fpx0`. Dans ce cas la règle d'ouverture de port s'appliquera si un paquet arrive sur le port 25 car elle a la priorité.
Mais si je veux que ma première règle s'applique quoi qu'il arrive ?
Et bien c'est la qu'intervient l'option quick.

```
pass in quick on fpx0 proto tcp from any to any port 25
block in on fpx0 proto tcp from any to any port 25
```

Dans ce cas le port 25 sera ouvert même s'il est indiqué ensuite comme fermé.

Les logs

On va mettre un système de log pour avoir un suivi des connections qui ont été accepté pour le cas où. Pour cela c'est toujours aussi compliquer : il suffit d'ajouter l'option log à la règle qui doit être surveillé.
Exemple :

```
pass in log on fpx0 proto tcp from any to any port 80
```

Configuration finale

Et voilà un exemple de règle sécurisé pour votre serveur web :

```
ext_if = "fxp0"
tcp_port = "{ 20, 21, 80, 443 }

# Ne pas filtrer sur l'interface de bouclage
set skip on lo0

# Normalisation de tous les paquets entrants.
scrub in all

# Mise en place d'une politique d'interdiction par défaut.
block in all

# Activation de la protection contre l'usurpation sur toutes les interfaces.
antispoof log quick for ext_if inet

# Règle pour laisser passer les connections entrantes vers le serveur web et
ftp.
pass in log on $ext_if proto tcp from any to any port $tcp_port
```

Et maintenant je le fait marcher comment ?

C'est bien beau d'avoir un fichier de configuration mais maintenant il faut le faire marcher !
Cette section est la pour ça. Nous allons voir ensemble comment dire à PF de se mettre en route ou comment faire des modification dans la configuration et les faires prendre par PF.

Vérifier la validité de pf.conf

Pour cela rien de plus simple, taper dans votre terminal :

```
pfctl -nf /etc/pf.conf
```

Si il ne dit rien c'est que ça marche.

Démarrer PF

Allez dans le fichier `/etc/rc.conf` et chercher la ligne où se trouve écrit :

```
pf=""
```

Il faudra le modifier en :

```
pf=YES
```

Redémarrer l'ordinateur et vous voilà avec pf fonctionnel !

Recharger la configuration

Vous avez modifier votre `pf.conf` et vous voulez que ce soit pris en compte. Pas de panique il faut lancer la commande suivante et ça sera fait :

```
pfctl -f /etc/pf.conf
```

Conclusion

Voila une bonne base pour mettre en route un firewall Paquet Filter sous OpenBSD ou pourquoi pas NetBSD ou autre.
Pour ceux qui connaissent déjà le fonctionnement d'un firewall s'étonneront sans doute de voir que je ne parle pas ici de gestion d'état. En effet depuis OpenBSD 4.0, PF le fait tout seul. Et oui vous voyez que c'est simple à mettre en route.
Cette article n'est pas complet et n'est pas une référence pour PF mais juste une introduction pour la mise en place d'un serveur web. Si vous voulez approfondir le sujet où que vous vous posiez des questions supplémentaire je vous invite a lire la documentation de PF disponible sur le site de OpenBSD qui est très bien et très complète.

PF.html v1.0 04/08/07 par Philippe BEAUMONT