Comment construire de toutes pièces un bridge/firewall sous openbsd

Patrick Marie, pmarie@altern.org

Janvier/Février 2001

Ce manuel (document) décrit l'utilisation d'un OpenBSD, de son installa tion, et de configuration d'IpFilter. Il permet de concevoir un pont filtrant.

1. Introduction

Cet how to est à l'origine d'un projet de conception d'un firewall (pare-feu) de cou
2 (couche liaison dans le schéma OSI) sous OpenBSD (au moment où j'écris
cela, c'est l'un des OS qui soient le plus pratique pour ça). Ce document a été écri
pour la compréhension et l'application de toutes personnes voulant installer et
configurer ce firewall, sans n'avoir aucunes notions en linux ou BSD (ou tout autre
système Unix possible).

Si tel est le cas, je vous conseille fortement de lire ce document dans son
intégralité, et d'être extrêmement sur de ce que vous faites en le faisant. C'est po
cela que
nous essayerons d'être le plus clair possible, afin de ne pas vous perdre en route a
cours de votre installation et de votre configuration.

Qu'est ce qu'un firewall ? Un firewall est une machine qui est situé entre deux rése
et qui permet de filtrer (soit bloquer, soit laisser passer) des paquets.

Figure 1: Notre firewall est bien entre les 2 réseaux, près à voir ce qu'il passe en
les deux. Le firewall pourra alors protéger le réseau A du réseau B, et vice versa.
Dans une telle configuration, un firewall aura 2 interfaces réseaux (soit deux carte
réseaux, une carte réseau et un modem configuré ...)

Quel intérêt d'un firewall de couche 2 par rapport à un autre firewall sous Windows
(erk) ou alors des implémentations sous linux (à savoir ipfwadm, ipchains et les
suivant) ?

Si vous connaissez un peu les réseaux TCP/IP, vous devriez savoir que toutes machine
sur un tel réseau à une adresse IP. Hors, toutes machines avec une adresse
IP est apte à recevoir des données pour elle-même, et donc est accessible. Accessibl
quoi ? et bien aux mauvaises intentions d'hypothétiques personnes. Si votre
firewall peut être piraté, alors tout votre réseau peut l'être aussi pour peu que ce
machine soit compromise.

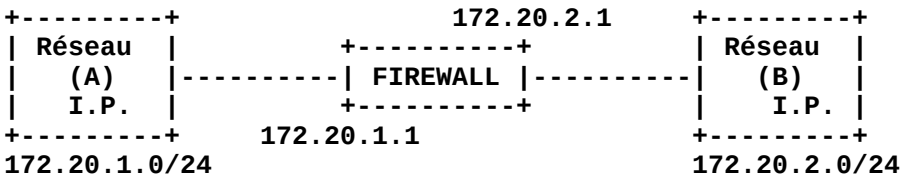


Figure 2: le firewall ici est obligé d'avoir 2 adresses IP, une sur le segment de
réseau 172.20.1.X, et une sur le segment de réseau 172.20.2.X. Notre 'firewall' peut
aussi faire office de routeur, connaissant le 'chemin' des deux réseaux. (il faudra
qu'il soit configuré pour faire du NAT)

La notation 172.20.1.0/24 est la notation CIDR. Elle signifie généralement un réseau
Pour notre exemple ici, elle signifie que les 24 premiers bits de l'adresse sont

fixes. Hors, un octet = 8 bits, donc les 3 (24/8) premiers octets sont fixes. Ce rés
comportera toutes les machines de 172.20.1.1 à 172.20.1.255 (une IP
finissant par 0 ne peut être utilisé que pour désigner un un réseau).

Que se passe-t-il alors pour un firewall sur un réseau TCP/IP qui n'a pas d'adresse
? Et bien celle ci peut voir les paquets qui viennent via le réseau (par exemple:
vous avez 4 machines reliées à un hub (concentrateur) simple, et une machine posséda
une IP envoie à une autre machine possédant aussi une adresse IP un
paquet. Le hub, ne sachant pas sur quel câble est la machine de destination, il va
envoyer ce paquet à toutes les machines du réseau, même notre machine qui n'a pas
d'adresse IP. Au contraire, cette machine ne pourra pas envoyer de paquets, ni en
recevoir pour elle. C'est ici qu'intervient notre firewall de couche 2.

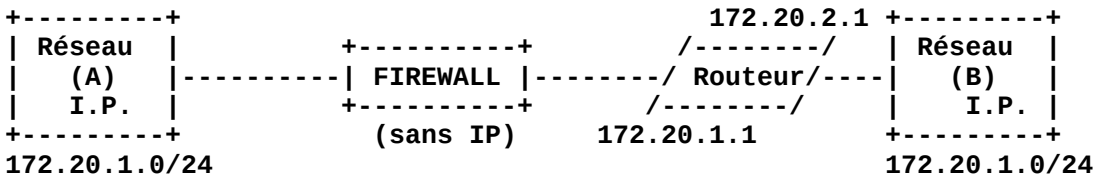


Figure 3: Le problème est que l'on doit rajouter un routeur pour ce schéma, qui rout
(dirigera) les bons paquets aux bons endroits. Notre firewall n'a plus d'IP à
lui, mais il est capable après configuration de filtrer les paquets qui passent d'ur
interface réseau à une autre.

C'est ce style de firewall que nous allons installer et configurer.

2. Pré-requis

2.1 notes des rédacteurs.

Comme annoté dans l'introduction de ce document, il a été rédigé par des étudiants
humains, et par conséquences, est capable d'intégrer des erreurs. Même si cela à
été évité au maximum, bien entendu, nous n'assurons pas que tout ce qui est contenu
dedans est 100 % sans erreurs.

2.2 matériel pour la conception de ce firewall

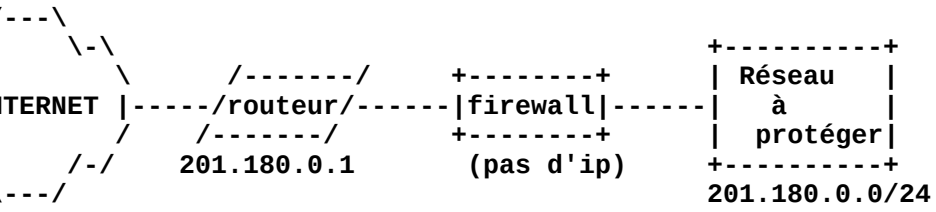
Un ordinateur 'complet' est nécessaire. J'entends par complet une unité centrale, av
un disque dur (au moins 800 mega octets), de la mémoire (64 meg semble un
bon minimum), deux interfaces réseaux (indispensables pour le firewall), une carte
vidéo, un clavier, un écran. Bien entendu, il est préférable que tout cela marche
avant de lancer l'installation d'openbsd. Si vous n'êtes pas sur que la totalité
marche, appelez un technicien qu'il vérifie le tout. Il est préférable d'en ère sur
avant, au
lieu de lancer 4 fois l'installation et de remarquer à la fin qu'une des deux cartes
réseaux ne marchent pas et qu'en fait, qu'il ne s'agisse que d'un conflit d'irq.

Une autre chose requise est un grand calme de votre part. Il se peut qu'il arrive de
problèmes et, pire, qu'ils ne soient pas documentés dans le document suivant. Il
existe tant de périphériques différents, et de situations possible que tout n'est pa
possible à documenter. S'il intervient un quelconque problème, je vous suggère de
vous renseigner sur Internet (www.openbsd.org, www.google.com est aussi une grande m
d'information.). Si vous avez du temps aussi, passez sur #openbsd (ou
#freebsd) sur Undernet(IRC), il y a toujours quelqu'un pour vous aidez (si vous
connaissez l'irc bien entendu.) Sinon il existe toujours usenet (les news) avec des
multitudes de personnes qui peuvent vous aider.

2.3 réseau que l'on va protéger par la suite (dans le cadre d'un exemple *complet*)

Après l'installation du fameux firewall (installation du système, et configuration d
bridge (qui permet de faire circuler les paquets d'une interface à une autre), on va

devoir utiliser et configurer notre firewall pour que celui ci soit utile à quelque chose. Pour qu'il le soit, il faut un réseau, qui va être (simplifié bien sur) celui ci:



On est dans le réseau 201.180.0.0/24 (fictif).

On aura beaucoup de stations de travail, mais on aura 2 serveurs. le premier d'ip 201.180.0.10/32 et le second d'ip 201.180.0.20/32

3. Installation d'openbsd

3.1 avant l'install

Ou avoir Openbsd ? grande question ! une réponse est sur le ftp d'openbsd officiel, est ftp.openbsd.org

Pour pouvoir installer openbsd, vous devez aller dans

ftp://ftp.openbsd.org/pub/OpenBSD/
le répertoire de la distribution en cours (le 2.8 est sortis le 1er décembre 2000), soit:

ftp://ftp.openbsd.org/pub/OpenBSD/2.8/

et récupérer (puis graver, en gardant bien la hiérarchie des sous-répertoires)
ftp://ftp.openbsd.org/pub/OpenBSD/2.8/* (tous les fichiers de ce répertoire, sauf les sous répertoires)
ftp://ftp.openbsd.org/pub/OpenBSD/2.8/tools/ (tout fichiers et sous reps)
ftp://ftp.openbsd.org/pub/OpenBSD/2.8/I386/ (tout fichiers et sous reps)

et c'est tout. Vous pouvez aussi prendre
ftp://ftp.openbsd.org/pub/OpenBSD/2.8/packages/
mais ce répertoire est volumineux, et l'intégralité ne pourra être gravée sur un cd de taille normale.

Une autre solution pour installer OpenBSD, est d'installer via un FTP. Pour cela, vo avez seulement besoin d'une disquette de boot d'install (voir ci dessous)

La disquette de boot.

Pour installer OpenBSD, il y a plusieurs moyens. Et dans les plusieurs moyens, il y encore plusieurs autre moyens. Soit vous avez les CD originaux (vendus), qui sont bootable (mettez le CD dans le lecteur et démarrer votre ordinateur. S'il ne démarre pas sur le lecteur de CD, et vérifier l'ordre de boot (dans l'incertitude, appelez un technicien.) Sinon, il ne vous reste plus qu'à faire une disquette de boo d'install. Pour cela, allez dans le répertoire I386 ou vous trouverez des images disks (sur un autre ordinateur, ou néanmoins un autre système d'exploitation.) L'ima disque pour cette disquette est par défaut floppy28.fs.

Si vous étés sous DOS, mettez une disquette vierge dans le lecteur, et utilisez l'utilitaire rawrite que vous trouverez dans le répertoire tools que vous avez du récupérer. Vous pouvez lancer rawrite à partir du lecteur de CD. Il va vous demande dans un premier temps l'image disk, et ensuite le lecteur de destination (en prin- cipe a:), tapez sur Entrée, et ça sera bon.

Si vous êtes sous linux, la commande à utiliser est 'dd' de la manière comme telle: if=floppy28.fs of=/dev/fd0a bs=1440k count=1 (if = input file, fichier d'entrée, of = output file, fichier de sortie, bs = nombre d'octets lu au moment et count = nombre de sessions (un bloc de 1440k (soit une disquette entière) au même moment.) Sous Openbsd et sans doute plusieurs autre unix (comme freebsd d'ailleurs), commande est sans doute la même.

Si tout c'est bien passé, vous devriez être capable de booter sous cette disquette. Laissez la dans votre lecteur, et redémarrez.

Si cela ne marche pas quand vous bootez, essayez de recommencez l'opération, ou à la rigueur essayez avec le fichier floppyB28.fs.

3.2 L'installation proprement dite.

Si la disquette boot, alors vous devriez être au même point maintenant que l'utilisateur avec son CD, et vous devriez avoir cela:

```
>> OpenBSD/i386 BOOT 1.23
boot>
```

Une chose que vous devez apprendre maintenant, est la touche <CR>. Cette touche, vou vous en servez tout le temps, sans même savoir que son vrai nom est <CR>. C'est la touche qui prend toujours les bonnes décisions, soit formater un disc dur, soit vider tout votre compte en banque, ou même partir avec votre femme à St Tropez. Cette touche (<CR>), veut dire 'Carriage Return', soit la touche Return ou Entrée. Pourquoi je vais utiliser <CR> à la place de Return ou d'Entrée ? car c'est plus court à écrire, tout simplement (je suis un informaticien, donc je suis feignant :))

Ceci dit, appuyez sur <CR> maintenant.

Il devrait apparaître:

```
Booting fd0a:/bsd:[bla bla bla bla
avec bla bla bla bla des nombres.
```

Si ça marche, alors un certain nombre de lignes en bleu devraient apparaître, avec t les périphériques détectés etc. Prenez du temps, car ça peut être long, surtout si vous avez un lecteur de CD lent, ou êtes sur disquette.

Et, miracle, au bout d'un moment apparaîtra:

```
sh: ./etc/rc: No such file or directory
Enter pathname of shell or RETURN for sh:

Appuyez sur <CR>.

(I)nstall, (U)pgrade or (S)hell?
```

Appuyez sur I puis <CR>, c'est l'installation que l'on fait.

```
Specify terminal type [pcvt25]:
```

On vous demande le nom du terminal que vous voulez utiliser. Tapez <CR> Tapez <CR> après une chose sous crochets signifie que vous prenez ce qu'il y a par défaut, soit la chose entre crochets.

Configurer le disque dur

Cette partie vous guide à partitionner puis formater votre disque dur pour OpenBSD. quelque chose passe mal, vous avez toujours la possibilité de rebooter si vous voulez tout recommencer; mais si vous avez des données sur d'autre partitions s

le même disque, et n'êtes pas expérimenté pour la chose, je vous conseille de faire appel à un assistant technique.

L'installeur va vous demander en un premier temps après le disque root (/). Il va vo lister tout les disques de votre système (sans doutes qu'un, si vous n'avez qu'un disque dur). Son nom sera sous la forme "wd0". Ca devrait d'ailleurs être la réponse par défaut sous crochets, et donc pressez <CR> encore.

La question suivante est cruciale: Voulez vous utiliser tout le disque pour OpenBSD On suppose que vous avez un disque dur neuf pour lui, donc 'yes' puis <CR>.

Il devrait vous dire qu'une partition de type 'BIOS "A6" (OpenBSD)' vient juste d'êt créé (avec un paquet d'explications en anglais.)

Et finalement:

```
Treating sectors 63-### as the OpenBSD portion of the disk.
You can use the 'b' command to change this.
Initial label editor (enter '?' for help at any prompt
```

(avec ### le nombre total de secteurs dans votre disque dur.)

Bien. Maintenant vous avez ">". Vous allez pouvoir maintenant modifier la table des partitions. Je vous conseille de lire avec attention. Ne vous inqui- etiez pas, l'affiche ">" ne va pas partir.

La première chose à faire ? Et bien appuyer sur <CR>. Cela va vous permettre d'avoir liste des commandes. La première commande dans la liste est 'p', pour afficher (print). Voyons ce dont le disque a l'air. Appuyez sur "P" et appuyez sur <CR>. Vous obtenez quelques informations sur la géométrie du disque, suivit par la liste des partitions actives. Notez le nombre qui suit: bytes/sector. Vous allez en avoir besoin plus tard.

Si votre disque était vide avant cette installation, alors vous devriez avoir une partition notée 'C' et une partition notée 'A'. La partition 'C' ne devrait pas être modifiée, c'est la taille totale du disque dur. La partition devrait être maintenant effacée, pour pouvoir modifier sa taille. Pour faire cela, appuyez sur "d a" et puis <;CR>;. 'd' est la commande pour supprimer une partition. 'd a' dit de supprimer la partition 'A'.

Maintenant, on peut créer nos partitions. On en a besoin au minimum de deux. Une partition de root (/), et une partition de swap, d'environ 200 meg. Vous pouvez bien sur en utiliser moins, surtout si vous possédez qu'un disque dur de taille rédu (= à 2 fois la taille de la ram, en général).

Quand vous ajouter une partition, l'installeur va vous demander après la taille de l partition. Il y a deux moyens pour: 1) la taille en mega octets, ou la taille en secteurs. Par simplicité, on va seulement voir en octets.

Vous devez d'abord créer votre partition de swap. Pour cela, pressez 'a b'. 'a' pour une nouvelle partition, 'b' pour lui dire de créer la partition 'B'. L'installeur va vous demander après un offset. Par défaut, ça doit être 63 (soit le premier utilisable). Pressez <CR> si cela est correct. Entrez 200M (ou la taille que vous voulez,
suivie de M (méga)). Il va vous arrondir au cylindre le plus près. Le type de FS devrait être 'swap'. (c'est par défaut. <CR> alors)

Maintenant il est temps de faire la partition principale. Tapez 'a a', et <CR> C'est pour créer la partition 'a'. La valeur par défaut pour l'offset devrait être un gran nombre. Pressez <CR>. Sa taille devrait être tout le reste du disque. Pour lui dire, (et pour être sur), pressez sur '*' (pour lui dire d'utiliser tout le reste de l'esp pour la partition.) Le type de structure de fichier (FS) pour cette partition doit ê "4.2BSD". Pressez sur <CR>, ou changez la si nécessaire. Le point de montage

(mount point), doit être "/". Pour tout le reste, utilisez les valeurs par défaut.

Maintenant, re-pressez sur "p". Si vous êtes content des valeurs et de vos partitior sauvegardez le tout en pressez "w" puis <CR>, et enfin sur "q" pour quittez. Si vous voulez modifier quelque chose, supprimez le tout ("d a<CR>", "d b<CR">), et recommencez.

Vous avez vos deux partitions, OpenBSD les connaît (swap et root (/)). Il peut vous donner l'opportunité de vous laisser configurer un autre disque, mais comme on a supposé que vous en avez qu'un seul, la valeur par défaut a ce moment doit être [done]. Pressez donc <CR> pour continuer. On va vous demander si vous voulez éditer /etc/fstab avec "ed". Dites "n" ! Finalement, il va être près pour vous prépa votre partition root. Dites lui "y" et souvenez vous que vous n'avez rien à perdre, donc allez-y !

Configuration du réseau

Dans cette partie, nécessaire si vous n'avez pas de CD d'openbsd et que vous voulez continuer l'installation par la suite via un FTP), on va apprendre à configurer pour l'installation le réseau, et plus précisément les interfaces réseaux.

L'installeur va vous rendre la vie simple. Si la carte est supporté avec le noyau pa défaut, alors l'installeur va le détecter. Si vous en possédez plusieurs, alors il v vous laisser le configurer.

Il devrait d'ailleurs vous demander à ce moment si vous voulez configurer le réseau. Taper sur 'y' puis <CR>.

On doit vous demander maintenant le nom de machine court, puis ensuite le nom de domaine. Si votre machine doit s'appeler 'londres.angleterre.org', alors 'londres' sera son nom de machine (hostname),et 'angleterre.org' sera le nom de domaine. Si vo n'avez pas de nom de domaine, alors vous pouvez le laisser blanc.

On va vous demander si vous voulez configurer votre IP via DHCP. Si vous possédez ur serveur DHCP, dites 'y' puis <CR>, et dans le doute, tappez 'n'<CR>

Vous devriez voir maintenant une liste d'interfaces reseaux correspondants aux votre Cela devrait ressembler à:

```
[ ] ni0
[ ] ni1
[ ] ni2
```

Vous aurez peut être pas 'ni' ou 0,1,2 .., mais ce qui correspond à votre configuration, c'est à dire peut être xl, ep, ... La seule chose qui importe, est qu est listé ici
autant de cartes que vous en possédez. Si ce n'est pas le cas, alors vous rencontrer sûrement des problèmes plus tard. Mais dans plus de 99% des cas, ça devrait être bon.

Admettons que vous les avez toutes. Il est alors temps de toutes les configurer. Il assez simple de modifier la configuration plus tard, surtout si vous n'avez pas répondu à toutes les questions.

Choisissons de configurer une interface. Par défaut, l'installeur choisira de configurer la première de la liste. Appuyez sur <CR>, et il va vous demander après s adresse IP. Comme on configure un firewall avec 2 cartes réseaux, vous devez savoir quelle carte correspond à cette interface. Si vous ne le savez pas, essayez de vous referez aux marques des deux cartes (si elles sont différentes, par exemple une 3com et une SMC), sinon, faites comme si la première est l'interface qui est reliée au réseau interne, et l'autre au réseau externe. De toutes façons, la configuration maintenant n'est pas très importante (sauf si vous installez OpenBSD v un

FTP), et elle pourra être très aisément être modifiée plus tard.

On va alors vous demandez l'adresse IP de cette interface. Si vous avez un serveur D
bien configuré dans le réseau, choisissez la configuration en DHCP. Si
vous êtes comme moi, que vous voulez être sur de ce que vous faites, ou alors change
tout le temps les cartes de places, ou alors que vous n'avez pas confiance en
la personne qui configure le DHCP, préféré la configuration statique. Choisissez bie
sur une IP dans votre réseau, non-utilisée (par exemple 10.0.0.1 si vous être
dans le réseau 10.0.0.0, avec un netmask de 255.255.255.0 (voir plus bas)). Une fois
l'IP rentrée, tapez sur <CR>, et l'installeur va demander le nom de la
machine. En principe, vous n'avez pas besoin de le redonner car cela a déjà été fait
peu plus tôt. (Donc <CR>)

Si vous avez choisis 10.0.0.1 comme adresse IP, vous devriez rentrer 255.255.255.0
comme netmask. Sinon, référez vous à la configuration d'une autre machine
sur le réseau, ou adressez vous a votre administrateur pour être sur de ce que vous
faites.

Maintenant vous devriez voir un ensemble d'informations quand au mode d'utilisation
votre carte réseau.(media directives) Le choix que vous devez effectuer ici
est totalement dépendant de la carte que vous avez. Si vous avez une carte 10/100 et
qu'elle est capable de détecter si le réseau est en 10 ou 100 meg/s, alors
choisissez "media auto select". Sinon, entrez ce que vous voulez dans les choix, et
puis tapez sur <CR>. En principe, la valeur par défaut est généralement la
meilleure.

C'est bon ! Vous venez de configurer une interface réseau, il ne reste plus qu'à fai
de même pour chaque carte, si vous voulez. Comme l'on configure un firewall
qui par la suite n'aura pas d'adresse IP, je vous conseille ici de ne pas configurer
les autres interfaces, car la, on a besoin d'une seule IP si l'on veut installer
OpenBSD via un FTP ou autre.

Cela étant dit, à "Configure which interface", écrivez "done" puis <CR>. La
configuration réseau est terminée.

On va vous demandez la route par défaut. Encore une fois, demandez conseil à votre
administrateur, ou appuyez simplement sur <CR> si vous n'en avez pas (C'est
l'adresse de votre routeur si vous en avez un, ou appelé aussi gateway, ou encore
passerelle en français.) Ensuite, les DNS. Toujours pareil, laissez ça vide, ou
rentrez celui de votre fournisseur Internet (ou celui de votre serveur local, si vou
en possédez un.) Si vous en mettez un, dites oui à la question "like to use the name
server now" sauf si la configuration est mauvaise.

Après, vous aurez le choix d'allez dans un shell pour finir la configuration. Si vou
êtes sur de tout ce que vous avez fait, vous n'avez pas besoin d'y aller. Si le rése
ne marche pas, et que vous possédez déjà les connaissances sur comment configurer un
réseau, alors vous pouvez tenter d'aller modifier la configuration. Dans la
plupart des cas, vous devriez ignorer cette possibilité.

De toutes façons, comme j'espère que vous installez OpenBSD grâce à un CD (j'ai dis
je vous le conseillais très fortement ?), vous ne devriez pas avoir un
besoin ultime du réseau maintenant.

L'installeur va monter la partition root dans le système de fichier, et procéder à l
copie des fichiers, puis va vous demander de choisir un mot de passe pour
l'utilisateur root. Si vous ne le savez pas déjà, l'utilisateur root est celui qui a
tout les droits, c'est l'administrateur du système. Pour la sécurité du système, et
cause
de ma paranoïa avancée, j'ai toujours préféré choisir des mots de passes du genre
'rt6ml80sl', c'est à dire assez long, difficiles à retenir à première vue, et compos
de lettres, de chiffres, et parfois de signes de ponctuations. Il faut savoir que
maintenant, les machines sont tellement puissantes en manière de calcul, qu'il est
parfois

facile de cracker des mots de passes, donc il est fortement recommandée (enfin, je l
recommande) d'en choisir un de cette forme. Bien entendu, gardez celui ci
pour vous. Faites tout de même attention à une chose si vous lisez ceci. Vous êtes e
train de taper un mot de passe avec un clavier américain (et oui, tapez sur un
A, et ça fait apparaître un Q, c'est magique.) Faites donc attention aux lettres
A,Z,W,Q maintenant, et au signes de ponctuations. Vous pourrez configurer le clavier
en français et changer de mot de passe après la fin de l'installation de toutes façc

Tapez votre mot de passe, et confirmez le en le retappant.

Installation des fichiers.

On va maintenant vous demander d'à partir de quel média il faut copier les fichiers.
c'est CDROM, alors c'est C puis <CR>. Via FTP, F puis <CR> ... Cette
partie devrait être très simple si vous avez un lecteur de CD reconnu. Si vous avez
choisis C, alors vous devriez avoir la liste des lecteurs près à être utilisé. Si la
liste
est vide, alors il(s) ne le sont pas. L'installeur va vous demander à partir de quel
média installer openbsd, et vous devrez taper son nom (quelque chose comme
'acd0'); Quand il demande le type de structure de fichier (file system), choisissez
cd9660, qui doit être la valeur par défaut. Le répertoire relatif au point de montag
qui contient les fichiers est /2.5/i386 (qui doit être aussi la valeur par défaut.)

Que faire si le lecteur de CD n'est pas reconnu ?

La meilleure façon est l'installation via FTP (on y vient). Si vous avez Internet (s
une ligne importante), alors vous pouvez l'installer à partir d'un serveur officiel
(ftp.openbsd.org, par exemple). Sinon, si vous avez un autre PC relié au réseau, vou
pouvez toujours installer et configurer un serveur FTP gratuit sur cette
machine (war-ftpd sous windows, pro-ftpd pour linux/autres BSD), et faites en sorte
le lecteur de CD de cette machine soit accessible en FTP anonyme. Quand
le tout est en place, faites en sorte que l'openbsd puisse communiquer avec cette
machine (bonne configuration IP entrées pendant l'installation, les deux machines
dans le même sous réseau, câble branchés ..), et quand l'installeur vous demande la
méthode d'installation, choisissez FTP. Spécifiez le serveur grâce à son adresse
IP, et ensuite donnez le bon chemin vers le CD ROM. Dans beaucoup de cas, après
l'installation, OpenBSD devrait détecter votre lecteur de CD. Si ce n'est pas le
cas, vous devriez le renvoyer et en changer pour un autre. N'achetez pas de Mitsumi,
ils ont tendance à créer beaucoup de problèmes.

Sélectionner les packages

Il y a trois sortes de systèmes de paquetages sous OpenBSD: ce qu'on appelle 'disk
set', qui sont tous les programmes importants au système. Même si l'on vous
demande si vous vous voulez les installer, ils sont vitaux. Le second système s'appe
le système 'package', qui sont dans un répertoire séparé, et qui contiennent des
outils pré compilés pour OpenBSD. Enfin, on a les 'ports' qui sont pour les programm
qui sont sous développement. Les sources sont fournies et devront être
compilées si besoin. Mais pour maintenant, on reste avec le premier type: ce sont de
fichiers compressé en .tar.gz (tarballs gzippés) qui vont être décompressé au
répertoire racine (/). L'installeur va alors vous demander ce que vous voulez instal
dans la liste suivante:

```
        [x] base28.tar.gz
        [x] etc28.tar.gz
        [ ] misc28.tar.gz
        [ ] comp28.tar.gz
        [x] man28.tar.gz
        [ ] game28.tar.gz
        [ ] xbase28.tar.gz
        [ ] xshare28.tar.gz
        [ ] xfont28.tar.gz
        [ ] xserv28.tar.gz
```

[x] bsd

Ceux qui sont sélectionnés par défaut (base, etc, man et bsd) sont requis pour que l
système. Vous aurez aussi sans doute besoin de misc et de comp. Vous pouvez
choisir ces paquets par plusieurs moyens. Le plus simple est d'écrire à l'invite le
du fichier l'un après l'autre et d'appuyer sur <CR>. (misc28.tar.gz<CR>, et
ensuite comp28.tar.gz<CR>) Quand vous avez fini, entrez done<CR>, et retapez <CR>
encore quand il vous demande si vous être près. Ca devrait lancer le
processus d'installation. Avec de la chance, il aura fini en quelques minutes, et vo
demandera si vous voulez installer d'autre sets. Le défaut est [n], donc pressez
juste sur <CR>.

Maintenant, vous êtes devant le panneau d'information sur la cryptographie avec
OpenBSD. Faites attention si vous êtes une entité commerciale, et renseignez vous
quand au point de vue légalité d'installer le paquet de cryptage ou pas. Si vous
l'installer, vous devriez probablement l'installer à partir d'un serveur FTP (il est
peut
être pas fournis sur le CD).

La prochaine étape est la configuration du fuseau horaire de votre localité. (presse
sur ? pour connaître votre fuseau. En principe, si vous êtes en France, Europe,
cela devrait être CET, ou CEST). Rentrer en grand ce fuseau, et <CR>

Il va vous demander si vous voulez faire marcher X plus tard sur le système. Dites n
suivis de <CR>. (X sur un firewall, c'est que vous n'avez rien compris)

C'est bon ! Vous avez finis d'installer OpenBSD. Il va vous demander de taper 'halt'
sur la console. Vous pouvez si vous voulez retirer le CD et taper 'reboot' à la
place.

Dernières notes: pensez à lire INSTALL.i386, les news, les faqs en cas de problèmes.

4. Configuration d'openbsd (post-installation)

On va voir ici certains point de configuration facultatifs ou importants, afin que v
puissiez réagir en cas de problème.

Avant de commencer, pour mettre le clavier en français, tapez:

```
$ kcon -m f8<CR>
(pour le m, essayez la touche ',')
```

Après avoir fini de rebooter, vous devriez avoir une invite (login:) mettez root,
validez par <CR>, donner le mot de passe, revalidez, et vous devriez avoir la consol
(taper sur <CR> si il vous demande quel type de terminal vous désirez).

Avant de faire quoi que ce soit, vous devriez prendre votre temps pour lire la page
manuel afterboot: (ne tapez pas le $, il représente le shell, ou vous pouvez
taper les commandes)

```
$ man afterboot<CR>
```

Même si vous ne comprenez pas tout, vous devriez être aider pour ce qui suit:

4.1 Réseau

Le premier point de configuration que nous allons effectuer est la configuration
réseau, et cela grâce à "ifconfig" principalement.

ifconfig est l'outil qui permet de gérer en temps réel la configuration des interfac
Il est important de savoir que les changements effectués avec ifconfig sont
appliqués, mais pas sauvés. Vous pouvez, accessoirement, lire l'aide:

$ man ifconfig<CR>

Pour voir votre configuration IP actuelle, tappez:

$ ifconfig -a<CR>

Ca fait apparaitre toute votre configuration, avec le détail de chaque interfaces présentes sur le système. Nous, on s'occupe seulement des cartes réseaux. Vérifiez déjà que toutes vos cartes sont dans la liste. Si ce n'est pas le cas, vous pouvez, grâce à la commande 'dmesg', qui permet de consulter les messages du noyau au boot, de voir si toutes vos cartes sont détectées ou pas. Si il en manque une, vérif bien qu'elle soit compatible avec OpenBSD, ou encore qu'elle soit bien enclenché dans le port PCI.

Une autre option d'ifconfig est d'utiliser -am (ou -Am), qui permet de voir les opti du media (pour, entre autre, savoir si la carte est en 100baseTX, 10baseT, ou encore en autoselect.

Bien sur, le principal intérêt d'ifconfig n'est pas de voir ce que l'on a, mais de l modifier. Pour cela, on doit connaître le format de la commande:

ifconfig <interface> [<address family>] [<address>] [<parameters>]

ou <interface> est le nom de l'interface physique que l'on veut configurer.

Par exemple, si votre carte est apparue comme xl0 quand vous avez tappé 'ifconfig -a alors c'est ca le nom de l'interface. Le reste des paramètres peuvent être optionnels, et dépendent de ce que vous essayez de faire. (note: les arguments entre [crochets] signifient qu'ils sont optionnels). Par exemple, si vous voulez désactiver la carte, vous pouvez le faire simplement:

$ ifconfig xl0 down<CR>

L'option <address family> décrit le type de réseau que vous possédez. Pour nous, on utiliser 'inet', car on va se connecter sur l'Internet. On pourrait aussi avoir 'ipx', 'atalk' (appletalk), et aussi d'autre types. Enfin, <address> est l'adresse I que vous voulez assigner à cette interface.

Si vous donnez juste l'adresse à ifconfig, il va simplement fixer cette adresse. Si vous vous trompez, vous n'avez juste qu'à réutiliser ifconfig pour choisir la bonne adresse, et l'ancienne va être effacée.

Les paramètres (<parameters>) sont nombreux et complets, voici la liste des plus importantes:

down : éteindre l'interface (l'interface n'agit plus du tout) up : la rallume netmas suivit du masque de réseau (ex: 255.255.255.0) alias : permet d'attribuer une ip en alias à une interface (une seconde IP). ex: $ ifconfig xl0 192.168.0.10 alias delete retire une adresse de l'interface (pour, par exemple retirer un alias). media : pour sélectionner le mode de transmission de la carte (si par exemple elle fait du 10/100 Pour vérifier ce qu'est capable votre carte, essayez 'ifconfig -am' mediaopts : vous permet de configurer les options du media (ex: "full-duplex", "half-duplex")

Les fichiers de configurations

Comme il a été dis, ifconfig ne sauve pas vos changements. Cela veut dire que tout v changements seront remis à 0 au prochain reboot. Si vous voulez que ces changements soient permanents, vous devez les sauvez dans des fichiers qui sont dans répertoire /etc, et sont nommés 'hostname.xxx', ou xxx est le nom de chaque interfaces. Le format de ce fichier est celui la:

<address family> <ipaddress> <netmask> <broadcast address> [<parameters>] [dest

<destination address>]

Si cette interface est configurée en DHCP, alors le fichier devrait contenir que le 'dhcp'. Autrement, ce fichier contrôle tout les attributs de l'interface que vous pouvez choisir grace à ifconfig. Ce fichier est lu au boot, grâce au script /etc/netstart, et le contenu est passé comme tels en arguments à ifconfig.

Comment éditer ces fichiers ?

Si vous n'avez jamais édité de fichier sous un environnement Unix, alors il est probable que vous soyez choqués. Il n'y a pas d'équivalent de 'edit' ou de notepad e ligne de commande. A la place, on a 'vi', qui est un éditeur très puissant, et qui e sans doute l'outil le moins intuitif jamais créé par l'homme. Si vous me croyez pas, tapez 'vi' puis <CR>, et regardez. Essayez de taper quelque chose. Ca ne marche pas comme vous le voulez. Pour quitter, tapez trois fois sur ESC (echap, en haut à gauche), puis ':q!'

Dans l'éditeur, il y a trois modes de fonctionnement: l'édition, le mode navigation, monde commandes. Pour l'édition, tout ce que vous tapez s'écrit à l'écran. En tapant sur echap, vous cesserez d'éditer, et vous pourrez naviguer dans le texte. Ur fois en navigation, vous pouvez taper sur ':' pour écrire des commandes, comme w pour écrire, q pour quitter, x pour écrire+quitter .. En navigation, si vous voulez écrire, essayez 'i' (comme insert). Si vous voulez passer du mode commande en mode navigation, effacez toute la commande.

Une fois votre fichier écrit, vous pouvez forcer à relire hostname.xxx en utilisant commande:

```
$ sh /etc/netstart<CR>
```

Mais le problème, c'est que la configuration ne s'arrete pas à la configuration des mais on a aussi, par exemple, les routes. (qui permettent de choisir la bonne interface pour envoyer le paquet.)

Dans notre cas, un firewall passerelle qui n'agit pas comme routeur, on ne devrait p avoir à configurer cela. Pour ce qui est de la configuration des cartes réseaux, tout ce que nous avons besoin de mettre dans les fichiers /etc/hostname.xxx, c'est ' dans chaque fichiers, et rien que cela.

D'ailleurs, tout ce qui suit n'est utile que si votre firewall n'agit pas comme un bridge.(contrairement à notre objectif final)

Dans le cas des routes, les commandes sont exécutées automatiquement par les scripts '/etc/netstart', et 'newifaliases'. Vous pouvez un jour regarder la page du manuel de route, mais pour le moment, on en a pas l'ultime besoin.

Il y a un paramètre qui ne marchera pas avec hostname.xxx, c'est 'alias'. Si vous voulez vraiment utiliser les aliases, vous aurez besoin d'éditer '/etc/ifaliases', e d'y
placer les commandes ifconfig (sans mettre ifconfig), comme pour les fichiers /etc/hostname.xxx Le format devrait alors être: <interface> <address> <netmask>

Si vous avez plusieurs alias pour une interface, utilisez plusieurs lignes. Pour activer ces changements, vous aurez besoin de rebooter, et de lancer la commande ifconfig avec le paramètre 'alias':

```
$ ifconfig alias<CR>
```

Un autre fichier important est '/etc/hosts'. Il contient les différentes machines du réseau, avec les IP, leurs noms avec et sans le domaine. Ca devrait être une bonne idée de mettre la configuration du firewall ici:

127.0.0.1 localhost 192.168.0.1 firewall-interne firewall-interne.mondomaine.com

201.180.0.1 firewall-externe firewall-externe.mondomaine.com

Si vous voulez faire un firewall-bridge, alors ne mettez que la 1er ligne, car notre machine n'a pas d'IP en principe.

Maintenant, si vous voulez configurer votre firewall en pont (bridge). On doit utili l'outil brconfig, pour configurer un pont entre nos 2 interfaces réseaux.

Mais avant, nous devons être sur que dans les fichiers /etc/hostname.xxx des cartes réseaux comportent bien toutes que 'up'.

Ceci fait, vous devez faire des petites modifications dans le fichier '/etc/sysctl.conf', et dé-commenter la ligne:

```
#net.inet.ip.forwarding=1 --> net.inet.ip.forwarding=1
```

(et si vous voulez, pour l'ipv6:)

```
#net.inet6.ip6.forwarding=1 --> net.inet6.ip6.forwarding=1
```

Cela a effet d'autoriser le forwarding entre les interfaces réseaux.

Dans /etc/rc.conf, vous avez besoin de modifier 2 lignes

```
ipfilter=NO  -->  ipfilter=YES
ipnat=NO     -->  ipnat=YES
```

Ca enclenchera l'exécution d'ipfilter et d'ipnat au boot.

Rebootez.

Une fois rebooté, vous devez configurer le bridge:

```
$ brconfig bridge0 add xl0 add xl1 up<CR>
```

Cela va creer un pont entre les 2 interfaces xl0 et xl1, et de le mettre en route (u

Si vous voulez sauver cela pour le prochain redémarrage, créer un fichier hostname.bridge0, et mettez 'add xl0 add xl1 up' dedans.

C'est bon, vous avez un bridge, et le réseau est configuré.

4.2 Lecteur de CD

Monter le lecteur de CD

Il est toujours utile d'avoir accès au CD (pour par exemple, installer le bash) Pour cela, il va falloir utiliser la commande 'mount'. Monter signifie raccorder un périphérique au système de fichier, ce qui signifie aussi qu'il va falloir le placer dans un endroit. Un répertoire vide conçu pour à été créé à l'installation, /mnt, da lequel on va créer des points de montages, pour monter ces périphériques.

Créons en un pour notre lecteur de CD:

```
$ mkdir -p /mnt/cdrom
```

(et pressez <CR>)

Maintenant, on va devoir monter le lecteur. Mais pour cela, on va avoir besoin de connaître son nom, et cela par la commande 'dmesg'

```
$ dmesg | more
```

Cela vous permettra de voir ou il est. Chercher des mots comme "ATAPI", ou "volume levels". Si vous trouvez, alors vous devriez voir à coté quelque chose du genre "acd0". Rajouter un "a" à la fin (c'est pour spécifier la lettre de la partiti et vous aurez votre lecteur de CD: "/dev/acd0a". Mettez un CD dans le lecteur, et tapez:

```
$ mount_cd9660 /dev/acd0a /mnt/cdrom
$ cd /mnt/cdrom
$ ls
```

Et voila, vous venez de lister le contenu du lecteur de CD. Si vous voulez changer d disque, il va falloir démonter puis remonter le lecteur:

```
$ cd /
$ umount /mnt/cdrom
$ mount_cd9660 /dev/acd0a /mnt/cdrom
```

Il est possible des fois que umount vous dise que le périphérique est en cours d'utilisation ('device is busy'). Dans ce cas, vous pouvez essayer "umount -f", qui permet de forcer le démontage.

Si vous voulez éjecter le CD, c'est grâce à eject:

```
$ eject /dev/acd0a
```

Si vous voulez rendre le processus un peu plus automatique, éditez /etc/fstab et rajoutez: (grâce à vi, tapez sur a) /dev/acd0a /mnt/cdrom cd9660 ro 0 0 (sauvegardez et quittez grâce à :wq)

si c'est bon, alors vous devriez pouvoir taper

```
$ mount /mnt/cdrom
```

pour monter le lecteur de CD, et en plus, il essaiera de le monter à chaque démarrag

4.3 (re)Compiler le noyau

Une grande étape facultative est le patchage et changement de noyau. En effet, il es toujours possible qu'au moment de la sortie de la distribution, le noyau possède un ou plusieurs bugs, plus ou moins mineures, et abaisse les capacités du système. P rectifier cela, il est possible de patcher et recompiler les sources. C'est tout à fait facultatif (la première fois ou j'ai installé ce système pour en faire un firewall, j'ai eu aucun problème sur ce point), mais c'est toujours utile de savoir faire (ou
d'avoir une doc dessus :)

Déjà, la première chose à faire, c'est de récupérer les sources du noyau, sûrement l fichier srcsys.tar.gz qui se trouve dans le répertoire de la version d'OpenBSD (sans doute 2.8, soit ftp.openbsd.org/pub/OpenBSD/2.8/), ou alors, si vous possedez CD officiels d'openbsd, elles se trouvent dans le répertoire sys/

Si vous n'avez PAS les cd officiels, et que vous avez le fichier srcsys.tar.gz (10-1 meg), alors copier le dans /usr/src, et de-tarball-gzippé le:

```
cp srcsys.tar.gz /usr/src
tar zxf srcsys.tar.gz
```

Ca va avoir pour effet de le décompresser, et donc de créer un répertoire /usr/src/s et de mettre tous les fichiers nécessaires dedans.

Si vous avez le CD officiel, alors vous aurez besoin de copier le répertoire sys/ du

1er CD. (après l'avoir monté)

```
cp -R sys/ /usr/src
```

Ca risque de durer assez longtemps.

Il faut maintenant éditer le fichier de configuration du noyau. Si vous avez bien le sources, alors vous pouvez lire le fichier: /usr/src/sys/arch/i386/conf/GENERIC Il correspond à la configuration du noyau par défaut.

On va alors le copier, et l'éditer:

```
cd /usr/src/sys/arch/i386/conf/
cp GENERIC MONNOYAU
vi MONNOYAU
```

On va, pour commencer, rajouter quelques lignes à la fin du fichier, pour améliorer firewall. Allez à la fin du fichier (plusieurs fois Ctrl+d), et appuyez sur 'o', et entrez les lignes:

```
option NMBCLUSTERS=8192
option NKMEMCLUSTERS=8192
option MAX_KMAP=120
option MAX_KMAPENT=6000
```

Sauvez (ESC, ':', 'w' puis <CR>), et quittez (ESC, ':', 'q' puis <CR>)

Je vous conseille de regarder un peu comment est structuré ce fichier, et ce qu'il comporte. A la rigueur, vous pouvez désactiver en rajoutant un '#' en début de ligne les périphériques dont vous n'avez pas besoin. Par exemple, si vous n'avez pas de SC vous pouvez désactiver tout les contrôleurs SCSI.

Attention: si vous ne savez pas ce que vous faites, ou n'êtes pas sur, alors n'y touchez pas !

Pour activer le filtrage dans le noyau, il faut regarder dans le fichier /usr/src/sys/conf/GENERIC, et etre sur que les deux lignes suivantes sont décommenté

```
option IPFILTER
option IPFILTER_LOG
```

Si il existe des # devant, retirez ce caractère. Il faut que les deux lignes soient actives sinon vous ne pourrez pas utiliser le filtreur. (Par défaut, elles sont actives, n'y
touchez pas !)

Compiler le noyau.

Les commandes à effectuer pour compiler le noyau sont :

```
cd /usr/src/sys/arch/i386/conf
config s /usr/src/sys b . MYKERNEL
make clean
make
```

Les 2 premières lignes permettent de préparer la compilation à partir du fichier de configuration. Les 2 autres compilent le noyau. Le make final devrait durer entre 2 et 5 minutes, sauf si vous possédez une machine lente, ou alors faites autre chose e même temps sur la station.

Si ca c'est bien passé, vous devriez à la fin voir quelque chose du genre:

```
        rm f bsd
        ld z Ttext F0100000 e start -x o bsd $(SYSTEM_OBJ) vers.o
        text data bss dec hex
        2023424 143360 1063348 3230132 3149b4
```

(ca peut changer pour votre ordinateur.)

Si vous avez des messages d'erreurs, il vous faudra vous référer aux docs. Si vous a
le message "cc not found" ou "gcc not found.", alors vous n'avez surement
pas installé le paquet 'comp', qui possède le compilateur. Si c'est le cas, il faudr
l'installer:

```
             $ cd /
             $ umount /mnt/cdrom
             $ mount_cd9660 /dev/cdrom /mnt/cdrom
             $ tar xzf /mnt/cdrom/2.8/i386/comp28.tar.gz
     (ou : $ tar xzf /mnt/cdrom/i386/comp28.tar.gz)
```

Ca aura pour effet de décompresser le paquet contenant les compilateurs. Tout devrai
bien se passer à partir de maintenant.

Installer le noyau

Il va falloir maintenant, si tout c'est bien passé, installer le noyau. Pour cela:

```
     Sauvegardez l'ancien:
             $ mv /bsd /bsd.old
     Copiez le nouveau à la racine:
             $ cp ./bsd /
```

Vous pouvez maintenant rebooter. Si ça va mal, vous avez le choix de rebooter sur
l'ancien noyau, grâce à l'OpenBSD Loader:

```
     >> OpenBSD/i386 BOOT 1.23
     boot> boot hda1:/bsd.old
```

(ou hda1 = votre partition)

L'ancien noyau rebootera, et vous pourrez à la rigueur refaire un autre noyau, ou
remettre l'ancien par défaut:

```
             $ mv /bsd /bsd.marchepas
             $ mv /bsd.old /bsd
```

4.4 Appliquer un patch au noyau

Le problème avec chaque systèmes d'exploitation, c'est qu'ils sont tellement utilisé
et si complet que certains bugs sont trouvés des fois. Il est parfois utile alors de
patcher le noyau, c'est à dire changer un peu (beaucoup) le code source du noyau, pc
le rendre plus sécurisé.

Pour cela, il faut dans un premier temps récupérer le dit patch, de patcher, et de
recompiler le noyau:

allons cherchez ce patch, et téléchargeons le:

```
             $ cd /tmp<CR>
             $ ftp ftp.openbsd.org<CR>
             (bla bla bla)
             login: ftp<CR>
             password: ftp@mail.com<CR>
```

(bla bla bla)

> cd /pub/OpenBSD/patches/<CR> > bin<CR> > get 2.8.tar.gz<CR>(télécharge le patch)
> bye

C'est bon, vous devriez avoir le patch.

une fois de retour au shell, décompressez les patchs:

$ tar zxf 2.8.tar.gz<CR>
$ cd 2.8<CR>

une fois ici, vous devriez avoir 2 répertoires (peut être plus), dont un appelé 'common'. Il contient les patchs communs pour toutes les plate formes. Un autre répertoire peut être i386, contenant les patchs spécifiques à Intel.

Si vous voulez installer un patch, il suffira d'effectuer la commande head sur le patch, pour voir le mode d'installation de ce patch.

Une installation devrait avoir la forme:

$ cd /usr/src/sys<CR>
$ patch p0 < bmap.patch<CR>

si ca ne marche pas (ca ne devrait pas marcher en fait :-), essayez:

$ patch p0 < /tmp/2.8/common/bmap.patch<CR>

Si le patch est bien appliqué, vous n'avez plus qu'à recompiler le noyau. (Faites to les patchs avant de recompiler :)

Machine sécurisée

Que vous installez un firewall bridge ou pas, vous aurez sûrement besoin d'utiliser moyen fiable pour communiquer avec votre machine à distance. La solution la plus fiable et sécurisée, est d'utiliser SSH. C'est un protocole similaire au telnet elle permet de se logger grace à un login/mot de passe à distance sur la machine, mais de facon sécurisée (les communications sont cryptées.)

Si vous avez accès au net et que vous voulez installer openssh:

$ pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/2.8/packages/i386/ssh-1.2.26-intl.tar.gz<CR>

(certainement plus à jour.)

le serveur et le client ssh seront installés, et ils seront toujours opérationnels s vous rebooter le système.

5. Configuration du firewall: Introduction à ipfilter

5.1 Introduction (si vous connaissez ipchains sous linux)

Voila la partie que je préfère :) IpFilter... Comme tout bon étudiant en informatiqu j'ai déjà été mis nez à nez face à Linux, et tout spécialement à la configuration d'ipchains, et à l'écriture de règles avec. Il est inutile de dire qu'ipchains est u bon firewall, néanmoins, en passant sous BSD, et particulièrement en utilisant IPfilter,
j'ai commencé à me demander si c'était moi qui commençais à délirer devant la puissa d'IPfilter comparé avec ipchains. IPfilter est de loin plus performant, car

est capable de reconnaître les différentes connections, de savoir dans quel sens vor
les connections ...

Prenons par exemple un schéma tout bête. J'ai deux machines, une avec un serveur SSH
(port tcp 22), et l'autre un client SSH. Entre les 2, un firewall. (on se fout
de la configuration des 2 machines, on dit qu'elles ne filtrent rien du tout, le
firewall est la pour.). Néanmoins, la machine cliente n'a pas du tout confiance en l
serveur, et veut simplement avoir accès au SSH quand elle veut, sans avoir a
reconfigurer le firewall.

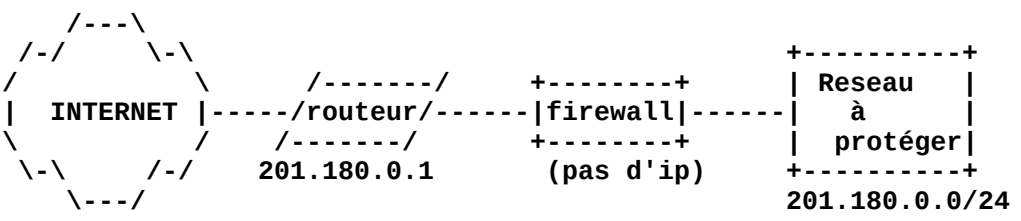
Si le firewall est sous Linux, avec un noyau 2.2.x (utilisant ipchains), alors on pe
effectivement tout bloquer, mais on doit laisser ce qu'il vient du port 22 du serveu
Cela reste correct, mais le problème reste que le serveur peut envoyer des données
'illicites' en disant qu'elles viennent du port 22 de cette machine (ou même une
tierce personne le peu), et ces paquets peuvent être acceptés sans problèmes par
Ipchains. Quant à notre IPfilter sous BSD, une fois bien configuré, ces paquets
auraient été tout simplement été ignorés. Il est possible de gérer les connections
(début, milieu, fin) avec ipfilter, et ce avec n'importe quel protocole IP (même icm
et udp qui en principe n'ont pas de début ni de fin de connection), de gérer les
différents aspect du TCP (comme les flags, la fragmentation des paquets ...), et mêm
de mentir l'état de certaines choses: par exemple, sous Ipchains, quand quelque chos
est bloquée, cela est très vite vu car il y a aucun retour. Avec Ipfilter, cela n'en
est rien, car il est possible d'envoyer une réponse, par exemple un icmp avec port
unreachable, pour dire "c'est fermé" alors que c'est ouvert.

En gros, n'allez pas penser que je n'aime pas Ipchains, je vais dire tout simplement
que je le trouve pas si complet qu'ipfilter (attention, je suis un être humain, qui
surcroit ne connaît pas tout, et donc peut en rater, mais en gros, je crois que j'ai
bien résumé :).

5.2 Rappel

Dans la configuration qui va suivre, toutes les IPs, machines, personnes sont FICTIV
Toutes ressemblances avec la réalité ne serait que pure coïncidence. (si si !)

Voila le réseau que l'on protège: On admet que la classe C appartienne à un ISP,
fournisseur de services Internet (web hosting, shells ...)



On est dans le réseau 201.180.0.0/24, on a des machines dedans, dont un serveur
multifonctions d'ip 201.180.0.127. Pour illustrer les exemples, on aura recourt à
3 personnages: Alice, Bob et Oscar. Alice est l'administratrice du firewall et du
réseau qu'elle protège. Bob est un ami d'Alice, aussi client des services fournis su
le
web par l'entreprise d'alice. Oscar est un 'pirate' sur l'internet, malicieux, et
essaye de faire tout pour gagner des accès illégaux sur le réseau d'Alice.

Il existe des autres machines bien sur le réseau, mais cette configuration suffit po
les exemples qu'on a besoin d'illustrer pour avoir un aperçu complet de toutes les
performances d'IPfilter.

Vous, vous êtes administrateur du firewall. Vous DEVEZ avoir des notions de sécurité
et des réseaux. Même si ce ne sont que les bases, c'est très utile. Je vais faire
mon mieux pour être le plus clair possible, sans toutefois réécrire 2 bouquins sur l
TCP/IP, ou bien la sécurité par firewalling. (J'en ai ni les connaissances, ni le
temps, ni le besoin).

5.3 Commandes de bases utiles pour IPF

Deja, ipf est configurable via un fichier de configuration. Différemment de certains autre firewall, ce n'est pas un script qui execute 45 fois la commande en ajoutant ou retirant quelque chose. Pour le fichier de configuration, le standard est '/etc/ipf.rules'. Néanmoins, si vous ne voulez pas le mettre la, libre à vous. Vous allez
devoir modifier ce fichier. Utilisez un éditeur de texte simple, comme 'VI', et si v faites partis des 99% des personnes qui n'aiment PAS 'VI' (honte à vous), vous avez la possibilité d'utiliser 'emacs', 'joe,'ed' (si vous savez vous en servir), .. Une fois votre fichier de configuration édité et sauvé, vous aurez la possibilité de lancer
la commande suivante:

```
$ ipf -Fa -vf/etc/ipf.rules<CR>
```

Commande qui vide toutes les anciennes règles (-Fa = flush all), et qui applique les nouvelles via -vf/etc/ipf.rules (-v = view, voir les règles (et si le cas se présent les erreurs), -f = prendre les règles du fichier /etc/ipf.rules)

Le fichier de configuration se présente comme tel: Dans une ligne, on a une règle. C peut mettre des commentaires en mettant un # devant. Une ligne peut
comporter une règle et un commentaire, et on peut avoir des lignes vides.

5.4 Règles de bases.

Pass,block, in,out, all

Chaque règles commencent déjà par l'un des deux mots suivants: soit "pass", soit "block", qui correspondent à laisser passer, ou bloquer.

Sur une règle, on doit aussi avoir le sens, soit "in", soit "out", qui signifient: i qui rentre, et out = qui sort.

Enfin, la configuration minimale nous oblige à appliquer cette règle à 2 ensemble de machines (un paquet vient d'une entité à une autre). Pour simplifier cela dans le début, on a "all", qui permet de dire de n'importe quelle machine vers n'importe que machine. (D'ailleurs, 'all' est un alias de 'from any to any', voir plus bas.)

On peut alors créer nos premières règles:

```
# Début du fichier /etc/ipf.rules
        pass in all
# Fin du fichier
```

L'instruction 'pass in all', correspond tout simplement à: je laisse passer ('pass') tout ('all') ce qui rentre ('in').

Pas encore perdu ? non ? parfait.

On peut avoir maintenant:

```
# Début du fichier /etc/ipf.rules
        block in all
        pass in all
# Fin du fichier
```

Alors la il y a problème: on bloque tout ce qui rentre, et après on laisse tout pass Il y a alors un nouveau point à éclaircir dès maintenant, c'est de la facon par laquelle ipf fait pour pratiquer ses règles. En fait, pour chaque paquets, il va lir les lignes une par une, et la règle qu'il appliquera sera la dernière correspondante paquet qui sera appliquée. Cela doit vous paraître ambigu, je le sais, ça l'a été pc

moi aussi :-)

Par exemple, on reçoit un paquet d'une machine X d'internet. Le firewall reçoit le paquet, et la commence la lecture du fichier:

```
block in all
pass in all
```

La première ligne lu est 'block in all'. IPF lit la règle et se dit: "Tient le paque correspond (il rentre 'in'), donc je dois le bloquer". IPF lit la seconde règle et s dit:
"Tient le paquet correspond (il rentre 'in'), donc je dois le laisser passer". La dernière règle qui correspond au paquet reçu est celle appliquée.

Le paquet passera donc.

A l'inverse, si notre fichier était:

```
pass in all
block in all
```

Le paquet ne serait pas passé (dernière règle qui correspond est 'block in all')

Si vous avez:

```
pass in all
pass in all
pass in all
pass in all
block in all
```

Le paquet ne serait pas passé. La dernière règle correspondante est celle appliquée. n'y a pas effet de cumulation ou autre.

Contrôler ce qu'il se passe: le mot clé "quick"

Admettez que vous ayez 500 règles (évidement pas composé uniquement de ce qu'il y a dessus, ce serait ridicule), et que vous voulez qu'il applique la première qui est un 'pass in all', sans voir tout le fichier de règle, par soucis d'économie temps. Il existe une solution à cela, qui est le mot clé "quick". Celui ci permet de dire: Tient, si la règle correspond, tu l'appliques immédiatement, sans lire le rest Plaçons notre quick:

```
pass  in quick all
block in       all
```

IPF lit la première règle: 'pass in quick all'. Le paquet correspond (il entre), dor grâce à quick la recherche est terminée (grâce à quick). La seconde règle n'est pas lue, elle ne le sera jamais d'ailleurs. Le paquet passe, sans préavis.

Filtrage par adresse IPs; 'from' et 'to'

C'est bien les 'pass in all', mais pas très utile, car soit on laisse tout passer (p d'intérêt à avoir un firewall), soit on bloque tout, et la il suffirai de couper tou simplement le câble. On va donc filtrer ce qu'il vient, et ce qu'il va vers des machines précises.

Prenons un exemple simple: On veut bloquer tout ce qu'il vient d'une IP précise comm par exemple celle d'Oscar, car il ne fait que d'envoyer des paquets étrange à l'ensemble des machines du réseau. Si son IP est 212.157.35.65, on peut alors écrire

```
block in quick from 212.157.35.65 to any
pass  in       all
```

La première ligne dit: 'je bloque des maintenant tout ce qu'il vient de cette ip et va n'importe où.' La seconde dit: 'je laisse tout passer' (mais ici a le contexte: j laisse tout passer ce qui n'a pas été bloqué par la première.) Tout passera, sauf ce qu'il vient d'Oscar.

Mais maintenant, le problème, c'est qu'Oscar a remarqué qu'on filtrait tout provenar de son IP, donc il va utiliser d'autres machines pour taquiner Alice. Cette dernière, ne voulant pas couper tout le traffic entrant pour ne pas défavoriser Bob utilise les services, elle a prit comme décision de couper tout ce qui rentre en destination d'une machine qu'Oscar tente de conrompre, d'ip 201.180.0.40. On a donc:

```
block in quick from any to 201.180.0.40
pass  in       all
```

Le problème est que maintenant, cette machine ne peut plus du tout rien recevoir de l'internet. Imaginons que l'on veut bloquer tout ce qui va vers cette machine, en provenance de l'une des machines d'Oscar (on a récupéré les ips grâce à un sniffeur, programme qui permet de visionner le trafic TCP/IP qui passe sur le réseau).
On bloque alors:

```
block in quick from 121.57.56.52 to 201.180.0.40
block in quick from 212.157.35.65 to 201.180.0.40
block in quick from 212.157.35.66 to 201.180.0.40
block in quick from 212.157.35.67 to 201.180.0.40
pass  in       all
```

Voila. Oscar aura du mal à embêter cette machine. Mais si l'on continue sur toutes l IP d'Oscar (il a une classe C rien qu'à lui, soit 254 adresses commencent ici par 212.157.35.), on a pas fini. On peut regrouper ces 254 règles, bloquant chacune chaque ip de ce réseau, en bloquant par une seule et unique règle qui va bloquer directement tout le réseau. Pour réaliser cela, on va utiliser la notation CIDR:

```
block in quick from 212.157.35.0/24 to 201.180.0.40
```

Cela signifie: je bloque directement tout ce qui rentre des IP commencant par 212.157.35.0, en ne prenant en compte que les 24 premiers bits (soit 3 octets (1 octect = 8 bits)) vers 201.180.0.40.

De même, Alice peut bloquer tout le réseau du traffic d'Oscar par:

```
block in quick from 121.57.56.52 to 201.180.0.0/24
block in quick from 212.157.35.0/24 to 201.180.0.0/24
pass  in       all
```

D'un point de vu plus général, il est possible que l'on reçoive quelquefois des paqu 'spoofés', c'est à dire dont les IPs ont été changées pour paraître anonyme et pour provoquer des bugs. Il est alors conseiller, si comme nous, on utilise des IPs le net, et non pas des IPs privées, de bloquer ces dernières:

```
block in quick from 10.0.0.0/8 to any
block in quick from 172.16.0.0/12 to any
block in quick from 192.168.0.0/16 to any
```

Ces IPs n'étant pas routables sur l'Internet, il est impossible d'en recevoir via l'internet. On peut aussi bloquer ce qu'il va a ses IPs:

```
block in quick from any to 10.0.0.0/8
block in quick from any to 172.16.0.0/12
block in quick from any to 192.168.0.0/16
```

Contrôler via les interfaces réseaux. mot clé 'on'

On peut choisir de filtrer les paquets provenant sur une interface physique spéciale
Par exemple, admettons que notre firewall a 2 cartes 3com, xl0 et xl1, l'xl0 étant
celle raccordée au routeur, donc à l'Internet, et xl1 l'interface réseau interne.

On veut recevoir des données sur xl0, mais pas venant d'ip étranges comme
192.168.0.0/16. Pour cela:

```
block in quick on xl0 from 192.168.0.0/16 to any
pass  in              all
```

Voila. Si l'on a des IP privées 192.168.0.0/16 sur notre réseau, elles ne seront pas
bloquées par le firewall, tandis que si l'on reçoit des paquets provenant d'ip
privées, ils seront bloqués.

De même, on peut maintenant bloquer les paquets provenant de 127.0.0.0/8 sur les 2
interfaces, car cette classe est uniquement utilisée pour l'interface locale, (lo),
et donc ne doit (peut) pas circuler via le réseau en principe. Soit:

```
block in quick on xl0 from 127.0.0.0/8 to any
block in quick on xl1 from 127.0.0.0/8 to any
pass  in              all
```

On peut aussi bloquer ce qu'il rentre par l'interne quelconque paquet ayant comme
source une adresse IP de notre réseau, car on est sur qu'il est spoofé. De même,
on peut bloquer ce qui veut sortir vers l'extérieur avec une IP de notre réseau inte
(faire attention au routeur, qui a une IP de notre réseau):

```
pass  in quick on xl0 from 201.180.0.1 to 201.180.0.0/24
block in quick on xl0 from 201.180.0.0/24 to any
pass  in quick on xl1 from 201.180.0.0/24 to 201.180.0.1
block in quick on xl1 from any to 201.180.0.0/24
```

Comment loguer ? le mot clé 'log'

Bon, si l'on prend toutes nos règles crées jusqu'à maintenant, et les avoir un peu
triée, on doit avoir quelque chose du genre:

```
#Début du fichier
# Ip privées
        block in quick on xl0 from 127.0.0.0/8 to any
        block in quick on xl1 from 127.0.0.0/8 to any
        block in quick on xl0 from 192.168.0.0/16 to any
        block in quick        from 10.0.0.0/8 to any
        block in quick        from 172.16.0.0/12 to any
# Méchant Oscar :)
        block in quick        from 121.57.56.52 to 201.180.0.0/24
        block in quick        from 212.157.35.0/24 to 201.180.0.0/24
# Ip publiques, mais ne devant pas arriver
        pass  in quick on xl0 from 201.180.0.1 to 201.180.0.0/24
        block in quick on xl0 from 201.180.0.0/24 to any
        pass  in quick on xl1 from 201.180.0.0/24 to 201.180.0.1
        block in quick on xl1 from any to 201.180.0.0/24
        pass  in              all
#Fin du fichier
```

Ca devient intéressant. Mais admettons qu'il arrive quelque chose, et que vous voule
voir ce qui est bloqué ou ce qui passe. Il existe un mot clé spécial, 'log', qui
permet de loger ce qui se passe sur le réseau.

On va loger par exemple les pseudos attaques d'Oscar, et aussi le spoof sur nos IPs:

```
     (...)
             block in log quick        from 121.57.56.52 to 201.180.0.0/24
             block in log quick        from 212.157.35.0/24 to 201.180.0.0/24
             pass  in     quick on xl0 from 201.180.0.1 to 201.180.0.0/24
             block in log quick on xl0 from 201.180.0.0/24 to any
             pass  in     quick on xl1 from 201.180.0.0/24 to 201.180.0.1
             block in log quick on xl1 from any to 201.180.0.0/24
             pass  in              all
```

Une fois cela fait, on pourra utiliser 'ipmon' pour voir ce qui est bloqué ici.

Je bloquai avec la configuration ci dessus, et j'ai reçu ça en lisant la sortie d'ip
:

```
     virginie:/usr/home/patrick/cours/firewall-l2$ ipmon
     24/12/2000 16:50:26.623685 xl0 @0:1 b 201.180.0.127 -> 201.180.0.10 PR icmp len
21504 icmp 8/0 IN
     ^C
     virginie:/usr/home/patrick/cours/firewall-l2$
```

Par exemple ici, j'ai reçu une demande de PING, sur l'interface xl0 (sur l'Internet)
d'adresse source 201.180.0.127 (donc ne devant pas arriver) pour l'adresse
201.180.0.10

Les protocoles; 'proto'

On peut filtrer les protocoles de la couche IP et de la couche TCP en précisant tout
simplement le mot clé 'proto'

Par exemple, si l'on veut filtrer toutes les requêtes icmp venant d'Oscar, il nous
suffit de faire:

```
             block in quick proto icmp from 121.57.56.52 to 201.180.0.0/24
             block in quick proto icmp from 212.157.35.0/24 to 201.180.0.0/24
```

On peut pareillement filtrer le tcp, l'udp, l'igmp ... tout les protocoles répertori
dans /etc/protocols

Les types d'icmp

Le problème avec l'icmp, c'est que c'est utile des fois, dans certains cas. On a ci
dessous bloqué tout les icmps venant des machines d'Oscar. Le problème est que
l'on veut par moment savoir si ses machines sont online ou pas, donc on veut quand m
pouvoir lui envoyer un ping et en recevoir la réponse, tout en lui bloquant
la possibilité de nous en envoyer. Si l'on connaît un peu les différents types d'icm
on sait alors que pour faire une requête, c'est le type 8 (echo request), et qu'une
réponse positive est le type 0 (echo reply).

```
           pass in quick proto icmp from 201.180.0.0/24 to 121.57.56.52 icmp-type 8
           pass in quick proto icmp from 121.57.56.52 to 201.180.0.0/24 icmp-type 0
           block in quick proto icmp from 121.57.56.52 to 201.180.0.0/24
```

Autre exemple, on veut aussi pouvoir faire des traceroutes un peu partout sur le net
Pour cela, on a besoin de laisser rentrer les icmp de type 11 rentrer:

```
pass in quick proto icmp from any to 201.180.0.0/24 icmp-type 11
```

Voila, pour information, les différents types d'icmp existant:

echo reply (0),
destination unreachable (3),
source quench (4),
redirect (5),
echo request (8),
router advertisement (9),
router solicitation (10),
time-to-live exceeded (11),
IP header bad (12),
timestamp request (13),
timestamp reply (14),
information request (15),
information reply (16),
address mask request (17),
address mask reply (18).

Les ports TCP et UDP, la commande 'port'

Comme l'icmp avec ses types, on peut bloquer les paquets udp et tcp visant certains ports. Pour cela, on utilise le mot 'port'.

Par exemple, on a un serveur SSH sur 201.180.0.10, et on veut qu'il n'y ait que le trafic SSH qui circule entre l'extérieur et l'intérieur:

```
pass  in quick proto tcp from any to 201.180.0.10 port = 22
pass  in quick proto tcp from 201.180.0.10 port = 22 to any
block in       all
```

Si on veut avoir des clients SSH qui vont se connecter vers l'extérieur, alors:

```
pass  in quick proto tcp from 201.180.0.0/24 to any port = 22
pass  in quick proto tcp from any port = 22 to 201.180.0.0/24
block in       all
```

De même, on peut bloquer tout ce qui vient vers nos ports telnet

```
block in quick proto tcp from any to 201.180.0.0/24 port = 23
```

Il ne faut pas oublier de spécifier le type de protocole à laquelle la règle agit. S l'on veut bloquer les deux protocoles pour un port, il faut le préciser :

```
block in quick proto tcp/udp from any to any port = 123
```

6. Construction de notre firewall: Stratégies, et fonctions avancées

A partir de maintenant, vous devez (je le re conseille), avoir votre machine en marc configurée, connaître grosso modo comment marche le tcp/ip, connaître le réseau que vous administrer, protéger, connaître les besoins et les risques venant d l'intérieur, et de l'extérieur.

6.1 Paranoïa

Sortons un peu du concept un admin, un gentil et un méchant, mais réfléchissons plus largement. Vous voulez protéger à tout prix votre réseau par ce firewall, et vous êtes aussi conscient que tout et n'importe quoi qui passe peut etre malsin. Si, comme moi, vous souffrez de paranoïa, faites comme moi, bloquez tout:

```
block in all
```

Bon, c'est vrai, des fois je préfère débrancher le câble réseau pour être sur. Mais, fait, ça n'a pas vraiment d'intérêt, car je veux de l'extérieur pouvoir voir les sit web hébergés sur mon serveur. Pour ça, je laisse passer ce qui rentre via xl0 pour m serveur web:

```
block in       all
pass  in quick on xl0 proto tcp from any to 201.180.0.127 port = 80
```

Jusque là, on sait faire. Il est vrai que c'est simple, mais c'est aussi efficace. N vous cassez pas la tête, il suffit simplement de laisser ce qui doit être laissé, et c'est
tout. Parfois, le plus simple est le meilleur :)

6.2 Les flags

Nous avons vu qu'il existait des moyens de bloquer les types de paquets ICMP. Tout comme l'icmp, l'udp et le tcp ont leurs spécificités. Par exemple, le TCP admet l'existence de 6 flags, permettant, entre autre, de dire ou en est le paquet vis à v de la connection. IP Filter admet un mot clé, 'flags' qui permet de bloquer ou laisser passer certains paquets composé de certains flags. Pour savoir exactement à quoi sert ces flags, je vous conseille de trouver une bonne doc sur le TCP. Les 6 flags existant sont URG (le flag d'urgence est valide), ACK (le numéro d'acquittemer est valide), PSH (le récepteur devrait passer cette donnée le plus tôt possible), RST (réinitialise la connection), SYN (synchronise les numéros de séquence pour initialiser les connections), FIN (L'émetteur à fini d'envoyer des données).

Par exemple, une connection TCP sur telnet s'effectue ainsi:

Le client envoie un paquet SYN au serveur, qui lui renvoie un paquet SYN/ACK (avec l 2 flags S et A), et finalement le client finit l'initialisation de la conenction par un flag Ack. Ensuite, le client (ou le serveur) envoie des paquets, avec le flag et finit chaque envoie par un paquet A & F. Le serveur peut utiliser aussi des flags P (push). (cela n'est qu'un exemple, ça peut différer.)

```
/* (extrait de TCPdump, sur un réseau réel:)
09:37:52.245812 tp4-pc11.iut.fr.1037 > 172.20.0.22.telnet: S
09:37:52.246240 172.20.0.22.telnet > tp4-pc11.iut.fr.1037: S ack
09:37:52.246584 tp4-pc11.iut.fr.1037 > 172.20.0.22.telnet: . ack
09:37:52.247120 tp4-pc11.iut.fr.1037 > 172.20.0.22.telnet: P ack
09:37:52.247203 tp4-pc11.iut.fr.1037 > 172.20.0.22.telnet: F ack
09:37:52.247612 172.20.0.22.telnet > tp4-pc11.iut.fr.1037: . ack
09:37:52.247741 172.20.0.22.telnet > tp4-pc11.iut.fr.1037: . ack
09:37:52.261715 172.20.0.22.telnet > tp4-pc11.iut.fr.1037: P ack
09:37:52.262048 tp4-pc11.iut.fr.1037 > 172.20.0.22.telnet: R
*/
```

Il est peu évidement, mais possible de régler une telle communication, grace a des règles avec des flags.

Si l'on veut modéliser cela, par exemple en admettant que quelqu'un sur le routeur (201.180.0.1) veuille se connecter en telnet sur le serveur (201.180.0.127), alors il faut retracer la communication point par point

Avant de faire cela, il est peut être subtil de spécifier maintenant qu'il est possi de dire que, si l'on dit: 'flags S', cela correspond a un 'flags S/AUPRFS', c'est a dire: S doit y être sur tout les flags AUPRFS. Seulement S devra être la pour que la règle soit applicable.

Si l'on a par contre 'flags S/SA', cela voudra dire: on doit avoir S et seulement S l'ensemble SA, mais on pourra aussi avoir un autre flag comme P ou R.

```
     # Début de la communication
     pass in quick on xl0 proto tcp from 201.180.0.1 to 201.180.0.127 port 23 flags
     # 2nde partie du handshake
     pass in quick on xl1 proto tcp from 201.180.0.127 port 23 to 201.180.0.1 flags
     # Fin du TCP handshake
     pass in quick on xl0 proto tcp from 201.180.0.1 to 201.180.0.127 port 23 flags
     # Envoie de données du serveur au client
     pass in quick on xl1 proto tcp from 201.180.0.127 port 23 to 201.180.0.1 flags
A/SAU
     # (ce qui signifie: j'ai le droit d'avoir le A mais pas avec S et U en même tem
tandis que les autres flags sont autorisés (P,F,R)
     # Envoie de donnée client au serveur:
     pass in quick on xl0 proto tcp from 201.180.0.1 to 201.180.0.127 port 23 flags
A/SAU
```

6.3 Gestion des connections, le mot clé 'keep state'

On vient de voir avec les flags qu'il était dur de gérer une connection TCP. Surtout que le modèle que je viens de donner est surement incomplet (j'ai pris qu'un exemple court et rapide), et donc on a besoin d'un outil un peu plus puissant et sim à manipuler. Celui ci s'appelle le 'keep state'. Il est possible avec IP-Filter de gérer toute une connection à partir d'un état initial.

Si l'on reprend par exemple la connection ci dessus, on aura tout simplement:

```
     pass in quick on xl0 proto tcp from 201.180.0.1 to 201.180.0.127 port 23 keep
state
```

Et voila. Quand le premier paquet qui sera applicable à cette règle passera, IPF placera la connection dans la table des connections 'keep state', et tout les autres paquets issus de cette connection passeront.

Il n'y a pas besoin de spécifier un second sens (du serveur vers le client), sauf si serveur doit a un moment donné se connecter au client (dans quel cas le serveur n'est plus le serveur et le client n'est plus le client, mais bon). Cela a aussi un autre grand avantage. Pensez au particulier qui veut se faire firewall qui lui perme de
tout faire, mais de ne laisser personne venir. Il lui suffit de mettre: (on dit qu'i possède une connection via modem (tun0), et un réseau local sur 192.168.0.0/24).

```
     pass out quick on tun0 from 192.168.0.0/24 to any keep state
     block in quick all
```

De plus, cela marche aussi pour l'udp et l'icmp, (même si il n'y a pas vraiment pas connections proprement parlé.)

Par exemple, on peut laisser le trafic spécifique a ICQ (vers le réseau icq.mirabilis.com, 205.188.153.0.24, port 4000)

```
     pass out quick on xl1 proto udp from 201.180.0.0/24 to 205.188.153.0/24 port =
4000
```

Cela nous permettra d'envoyer des messages et d'en recevoir du serveur.

Néanmoins, il y a un problème à cela. Les entrées dans la table 'keep state' ne rest pas indéfiniment. En fait, une bout d'une minute d'inactivité, celle ci est retirée, et il est alors probable que les données ré envoyés derrière soient tout de suite droppées par le filtreur si elle ne sont pas applicable à l'une des règles 'pass'.

Il est aussi possible de cumuler 'keep state' et 'flags', pour par exemple dire de prendre la communication qu'a son début, et d'ignorer le reste si le début n'a pas é une fois présenté, et cela pour éviter certains types de scan, comme le Xmas scan, o le Fin scan (envoie que des paquets F pour voir ce que l'on répond.), et l'on a

donc, si l'on veut avoir que le début de la connection TCP:

```
     pass in quick on xl0 proto tcp from any to 201.180.0.127 port 23 keep state fla
S
```

On devra avoir l'initialisation de la connection, et dès le début, pour que tout pas

Finalement, on peut aussi par exemple envoyer des pings a partir de notre réseau ver
l'extérieur, et ce par:

```
     pass in quick on xl1 proto icmp from 201.180.0.0/24 to any icmp-type 8 keep sta
```

6.4 Conserver piste des fragments, 'Keep frags'

Dans le même esprit, il existe le mot clé 'keep frags'. Cela permet de considérer co
faisant partit d'un paquet tout les fragments de ce paquet. En gardant 'keep
state' et 'keep frags', on gardera alors la piste de chaque paquets fragmentés, et l
traiter comme ci ils ne l'avaient pas été.

```
pass in quick on xl0 proto tcp from any to 201.180.0.127 port 23 keep state flags S
keep frags
```

Renvoyer un message a un paquet bloqué

Le problème si l'on bloque des paquets, c'est qu'il n'y a pas de réponses. On peut
alors aisément savoir qu'il y a à cet endroit même un firewall. Il existe des moyens
pour palier cela, et pour faire croire que la machine n'est pas accessible, que
certains ports sont fermés ...

On a trois façons différentes de renvoyer des paquets. On a 'return-rst', qui permet
TCP de renvoyer que la communication à été réinitialisé. (message qui est
envoyé, par exemple quand un port est fermé.) On a aussi 'return-icmp', qui permet d
renvoyer un ICMP, ou encore 'return-icmp-as-dest', qui permet de faire
croire que c'est le destinataire qui a envoyé le paquet (dans le 1er cas, on peut di
que la machine n'est pas accessible, la 2 ème pour par exemple dire qu'un port
udp est fermé.)

Comment les utiliser ?

Pour le TCP:

avant, on pouvait avoir:

```
        block in log on xl0 proto tcp from any to 201.180.0.127 port = 23
        pass  in     all
```

après modification:

```
        block return-rst in log on xl0 proto tcp from any to 201.180.0.127 port = 2
        pass  in all
```

Il faut bien faire attention que return-rst est uniquement utilisé pour le TCP. C'es
pour cela que l'on utilise return-icmp et return-icmp-as-dest pour les paquets udp e
les messages icmp.

De plus, il est possible avec return-icmp et return-icmp-as-dest de spécifier le
message qui est envoyé. Selon le TCP/IP illustré de Stevens, le message par défaut
est destination non accessible, (destination unreachable, 3), et ceci pour les paque
udp visant des destinations entraînant une erreur (ex: port fermé.)

On passe donc d'une règle (trafic udp vers un dns)

```
        block in log on xl0 proto udp from any to 201.180.0.127 port = 53
```

à

block return-icmp(port-unr) in log on xl0 proto udp from any to 201.180.0.1
port = 53

Le petit problème à cela, c'est que c'est le firewall qui va renvoyer ce paquet, ave
sa propre ip (s'il en a une.) On peut palier ce problème en utilisant
return-icmp-as-dest, qui va permettre au firewall de renvoyer le paquet en signant p
l'ip de la machine à qui le paquet était destitué.

On aura donc, simplement

block return-icmp-as-dest(port-unr) in log on xl0 proto udp from any to
201.180.0.127 port = 53

Enfin, on ne peut utiliser les 'return*' qu'avec les block.

Time to live, Type of Service

Il est possible aussi de tester les ttl et les tos des paquets qui arrivent. Par
exemple, on peut bloquer certains traceroutes

block in quick ttl 0 all

Il est possible de spécifier une type de services (champs TOS dans l'entête IP),
exemple:

block in quick tos 0x00 ttl 0 all

L'utilisation des ttl et des tos dans un fichier ipf reste pour les experts, à mon
avis.

Il peut être judicieux de préciser qu'il existe un moyen de cacher l'existence de la
machine firewall à cet endroit, en falsifiant les ttl. Il existe 'fastroute', qui
permet au
firewall de ne pas modifier le ttl quand le paquet passe par la machine.

block in quick on xl0 fastroute proto udp from any to any port 33434 >< 33465

7. Autres fonctions d'IPF

Il existe beaucoup d'autre fonctions pour ipfilter. Je vais les décrire une par une
rapidement.

7.1 Format d'une ligne IPF

On a vu jusqu'à maintenant beaucoup d'exemples, et nos règles commencent à être
longues. Le problème, est que vous rencontrerez peut être des erreurs de
formatages du fichier, au lancement d'ipf. Je vous présente ici, comment formater
correctement une règle pour ipfilter:

(Comment l'utiliser:

```
objet = [ "truc" ] machin .
machin = "bla1" | "bla2" | "bla3" .
```

Les expressions avec les "" veulent dire que ça doit être écrit comme ça l'est ici.
expressions entre crochets sont optionnelles. Dans les expressions si dessus, on
peut avoir:

"truc bla1" ou "bla2" ou "truc bla2" ...)

```
filter-rule = [ insert ] action in-out [ options ] [ tos ] [ ttl ]
          [ proto ] [ ip ] [ group ].
insert    = "@" decnumber .
action    = block | "pass" | log | "count" | skip | auth | call .
in-out    = "in" | "out" .
options   = [ log ] [ "quick" ] [ "on" interface-name [ dup ] [ froute ] ].
tos  = "tos" decnumber | "tos" hexnumber .
ttl  = "ttl" decnumber .
proto     = "proto" protocol .
ip   = srcdst [ flags ] [ with withopt ] [ icmp ] [ keep ] .
group     = [ "head" decnumber ] [ "group" decnumber ] .
block     = "block" [ icmp[return-code] | "return-rst" ] .
auth    = "auth" | "preauth" .
log  = "log" [ "body" ] [ "first" ] [ "or-block" ] [ "level" loglevel ] .
call = "call" [ "now" ] function-name .
skip = "skip" decnumber .
dup  = "dup-to" interface-name[":"ipaddr] .
froute    = "fastroute" | "to" interface-name .
protocol = "tcp/udp" | "udp" | "tcp" | "icmp" | decnumber .
srcdst    = "all" | from to .
fromto    = "from" [ "!" ] object "to" [ "!" ] object .
icmp = "return-icmp" | "return-icmp-as-dest" .
object    = addr [ port-comp | port-range ] .
addr = "any" | nummask | host-name [ "mask" ipaddr | "mask" hexnumber ] .
port-comp = "port" compare port-num .
port-range = "port" port-num range port-num .
flags     = "flags" flag { flag } [ "/" flag { flag } ] .
with = "with" | "and" .
icmp = "icmp-type" icmp-type [ "code" decnumber ] .
return-code = "("icmp-code")" .
keep = "keep" "state" | "keep" "frags" .
loglevel = facility"."priority | priority .
nummask   = host-name [ "/" decnumber ] .
host-name = ipaddr | hostname | "any" .
ipaddr    = host-num "." host-num "." host-num "." host-num .
host-num = digit [ digit [ digit ] ] .
port-num = service-name | decnumber .
withopt = [ "not" | "no" ] opttype [ withopt ] .
opttype = "ipopts" | "short" | "frag" | "opt" ipopts  .
optname   = ipopts [ "," optname ] .
ipopts  = optlist | "sec-class" [ secname ] .
secname   = seclvl [ "," secname ] .
seclvl  = "unclass" | "confid" | "reserv-1" | "reserv-2" | "reserv-3" |
          "reserv-4" | "secret" | "topsecret" .
icmp-type = "unreach" | "echo" | "echorep" | "squench" | "redir" |
          "timex" | "paramprob" | "timest" | "timestrep" | "inforeq" |
          "inforep" | "maskreq" | "maskrep"  | decnumber .
icmp-code = decumber | "net-unr" | "host-unr" | "proto-unr" | "port-unr" |
          "needfrag" | "srcfail" | "net-unk" | "host-unk" | "isolate" |
          "net-prohib" | "host-prohib" | "net-tos" | "host-tos" |
          "filter-prohib" | "host-preced" | "cutoff-preced" .
optlist   = "nop" | "rr" | "zsu" | "mtup" | "mtur" | "encode" | "ts" |
          "tr" | "sec" | "lsrr" | "e-sec" | "cipso" | "satid" | "ssrr" |
          "addext" | "visa" | "imitd" | "eip" | "finn" .
facility = "kern" | "user" | "mail" | "daemon" | "auth" | "syslog" |
         "lpr" | "news" | "uucp" | "cron" | "ftp" | "authpriv" |
         "audit" | "logalert" | "local0" | "local1" | "local2" |
         "local3" | "local4" | "local5" | "local6" | "local7" .
priority = "emerg" | "alert" | "crit" | "err" | "warn" | "notice" |
         "info" | "debug" .
hexnumber = "0" "x" hexstring .
hexstring = hexdigit [ hexstring ] .
decnumber = digit [ decnumber ] .
```

```
compare = "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" |
        "gt" | "le" | "ge" .
range     = "<>" | "><" .
hexdigit = digit | "a" | "b" | "c" | "d" | "e" | "f" .
digit     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
flag = "F" | "S" | "R" | "P" | "A" | "U" .
```

Exemple: On veut faire une règle qui va bloquer tout les paquets UDP vers le port 53 d'un pc ou l'ip est 201.180.0.127, venant d'ips de la classe A de 11.0.0.0, en logant, et en renvoyant un message du destinataire.

On prend la 1er ligne:

filter-rule = [insert] action in-out [options] [tos] [ttl] [proto] [ip] group].

On doit mettre une 'action', et un 'in-out' pour commencer

L'action: action = block | "pass" | log | "count" | skip | auth | call . Nous on veu bloquer, on prend 'block'. Il correspond à un sous ensemble, on regarde:

block = "block" [icmp[return-code] | "return-rst"] .

Déja, on doit écrire 'block'. De plus, nous on veut retourner un message. C'est de l'udp, pas le droit d'utiliser return-rst. On va se referez à 'icmp':

icmp = "return-icmp" | "return-icmp-as-dest" .

On utilise return-icmp-as-dest, c'est ce que l'on veut.

Jusqu'à maintenant, on a: 'block return-icmp-as-dest'

On remonte un peu plus haut, a 'in-out':

in-out = "in" | "out" .

On prend 'in', c'est ce qu'on cherche à bloquer On a 'block return-icmp-as-dest in'

Maintenant, on va parcourir les sous sections 'options', 'proto', 'ip' pour arriver notre règle complète.

Vous voyez, c'est très simple :) (enfin, j'espère ...)

7.2 Logger

Tout administrateur se doit d'être parano. Si ce n'est pas votre cas, alors changer métier, c'est pas celui pour lequel vous êtes fait :) Moi, je le suis (enfin, c'est qu'on dit de moi), et j'ai tendance à vouloir savoir ce qui se passe, et surtout ce passe sur mon réseau. IPFilter permet plusieurs techniques pour loger, en plus du mot 'log'.

Déjà, il est possible de loger le trafic, sans utiliser de mot "block" ou "pass", ma tout simplement "log"

log in from any to 201.180.0.127

Le soucis, c'est qu'on va loger plein de choses. Moi ce que je veux, c'est pouvoir é informé tout de suite en cas de problème important. Vous pouvez appliquer à vos règles une nouvelle façon de loger, pour loger via le daemon syslog. En le configurant correctement, vous pourrez voir, par exemple, sur votre console ce que vous voulez. Il nous suffit, par ipf, de rediriger les logs vers une section de sysl et de son niveau d'importance. On a, comme choix:

```
            facility = "kern" | "user" | "mail" | "daemon" | "auth" | "syslog" |
                   "lpr" | "news" | "uucp" | "cron" | "ftp" | "authpriv" |
                   "audit" | "logalert" | "local0" | "local1" | "local2" |
                   "local3" | "local4" | "local5" | "local6" | "local7" .
            priority = "emerg" | "alert" | "crit" | "err" | "warn" | "notice" |
                   "info" | "debug"
```

Pour loger avec syslog, il faut utiliser le mot clé 'level' après 'log', ex:

```
pass in log level logaltert.emerg on xl0 from 127.0.0.0 to any
```

Si vous voulez seulement voir déjà ce qui passe avec le mot log, il faut savoir que
redirige tout les flux 'log' vers /dev/ipl. Vous pourrez le lire en direct grâce a l
commande 'ipmon'.

On peut aussi parler de l'action 'count' qui ne permet simplement que de compter le
paquet. Il ne sera ni logué, on n'agira pas dessus, simplement compté:

```
count in all
```

7.3 Les autres actions

On trouve call, skip <n>, auth, preauth.

La première, call, permet d'appeler une fonction du noyau, au passage d'un paquet.
Cette action est réservée aux codeurs, et à le privilège de ne pas être
documentée.

skip <n>, permet de sauter n règle si cette règle est rencontrée. auth et preauth so
des fonctions d'authentifications dans le noyau pour la distribution vers les
programmes, et vice-versa.

7.4 Les groupes

Au bout d'un certain moment, il est possible que vous vouliez regrouper vos règles p
vous simplifier la vie, par exemple, vous avez 2 machines différentes (2 ips),
et vous voulez avoir des règles différentes pour ce qui va sur l'une ou sur l'autre.

Pour cela, ipf facilite les choses, et permet le regroupement.

On peut 'créer' un groupe en rajoutant a la fin de la règle 'head X', ou X est un
entier, et dire qu'une règle fait partis de ce groupe, grâce à 'group X'.

exemple:

```
         block in quick on xl0 all head 1
           block in     quick on xl0 from 192.168.0.0/16 to any  group 1
           block in     quick on xl0 from 172.16.0.0/12 to any   group 1
           block in     quick on xl0 from 10.0.0.0/8 to any      group 1
           block in     quick on xl0 from 127.0.0.0/8 to any     group 1

         block out quick on xl1 all head 10
           pass out quick on xl1 proto tcp from any to 201.180.0.127/32 port = 80 fl
S keep state group 10
           pass out quick on xl1 proto tcp from any to 201.180.0.127/32 port = 21 fl
S keep state group 10
           pass out quick on xl1 proto tcp from any to 201.180.0.127/32 port = 20 fl
S keep state group 10
           pass out quick on xl1 proto tcp from any to 201.180.0.127/32 port = 53 fl
S keep state group 10
```

8. Cas pratiques/Misc.

8.1 Utilitaires autour d'IPF

Manipuler les règles du filtreur: ipf

Les fichiers de règles doivent etre chargés avec 'ipf'. Les règles peuvent etre stockées dans n'importe quel fichier du système, mais on préferera /etc/ipf.rules, /usr/local/etc/ipf.rules, ou bien encore /etc/opt/ipf/ipf.rules

Il faut savoir qu'ipf gère en meme temps 2 ensembles de règles: les actives, et les inactives. Par défaut, toutes les actions sont faites sur l'ensemble actif. On peut jouer sur l'ensemble inactif en rajoutant l'extension -I à la ligne de commande avec ipf. On peut interchanger les 2 ensembles (actifs et inactifs) grace à -s. Cela permet de tester un nouvel ensemble de règles, sans effacer l'ancien.

On peut rajouter des règles ayant pour sources un fichier, mais on peut aussi les retirer si besoin; on utilisera pour cela l'extension -r, mais on trouve plus simple d'utiliser -F, qui permet de tout vider.

Finalement, il est pratique de charger une nouvelle configuration grace à la commanc

```
$ ipf -Fa -f /etc/ipf.rules
```

(ou ipf -Fa -vf /etc/ipf.rules pour voir les erreurs.) Pour plus d'informations, on vous conseille de lire la page du manuel: man 8 ipf

L'utilitaire ipfstat

ipfstat est un utilitaire de monitoring. Il permet d'avoir des statistiques sur ce c passe ou qui est bloqué via ipf.

```
virginie:/usr/home/patrick$ ipfstat
 input packets:         blocked 0 passed 28104 nomatch 27022 counted 0 short 0
output packets:         blocked 0 passed 28062 nomatch 26943 counted 0 short 0
 input packets logged:  blocked 0 passed 0
output packets logged:  blocked 0 passed 0
 packets logged:        input 0 output 0
 log failures:          input 0 output 0
fragment state(in):     kept 0  lost 0
fragment state(out):    kept 0  lost 0
packet state(in):       kept 0  lost 0
packet state(out):      kept 0  lost 0
ICMP replies:   0       TCP RSTs sent:  0
Invalid source(in):     0
Result cache hits(in):  1082    (out):  1119
IN Pullups succeeded:   0       failed: 0
OUT Pullups succeeded:  0       failed: 0
Fastroute successes:    0       failures:       0
TCP cksum fails(in):    0       (out):  0
Packet log flags set: (0)
        none
virginie:/usr/home/patrick$
```

Ipfstat est capable de vous montrer les statistiques de chaque règles actives grace -i ou -o, qui permettent de voir les règles en in ou en out, et combiné avec -h, qui permettent d'avoir le nombre de hits pour chacunes d'elle.

On peut voir le numéro de chaque règles grace à -n;

On peut avoir des stats sur la table d'états (keep state), grace à -s.

N'oubliez pas de lire 'man ipfstat'.

L'utilitaire ipmon

On l'a déjà décrit, on peut se servir d'ipmon pour lire dans le périphérique /dev/ip
qui recense tout les logs des règles avec le mot log.

On peut aussi voir les connections dans la table d'états, grace à -o S:

```
$ ipmon -o S
```

De meme, on peut voir les logs des traffics par rapport au NAT, grace à -o N

```
$ ipmon -o N
```

Sinon, 'man ipmon'.

8.2 Manipuler les règles du NAT: ipnat

Bien que ne rentrant pas directement dans la configuration d'un firewall/bridge, il
utile de préciser l'existance ipnat. Il permet de faire de l'ip masquerading, appelé
aussi NAT, pour Network Adress Translation. Et cela sert généralement à partager une
connection à l'internet à l'interieur d'un réseau local (l'ip sur l'internet est
distribuée au sein du Lan.)

Comme ipf, ipnat est configurable via un fichier de règles, /etc/ipnat.rules. On pou
préciser un fichier de configuration grace à -f, et flusher toutes les règles grace
-CF (-C pour effacer toutes les règles, et -F pour effacer toutes les entrées dans l
table de translation NAT.)

La commande sera alors de la forme:

```
$ ipnat -CF -f /etc/ipnat.rules
```

Exemples de règles ipnat

L'utilisation basique d'ipnat se résume à ce que fait l'ip masquerading de linux:
(admettons que 201.180.0.37 soit une passerelle entre un sous réseau 192.168.0.0/24
qui veulent avoir accès à internet. Cette classe n'est pas routable sur le net, donc
a besoin de faire du NAT: - la carte réseau xl0 est du coté du réseau
201.180.0.0/24)

```
map xl0 192.168.0.0/24 -> 201.180.0.37/32
```

Voila. Les machines du réseau local sont maintenant capable d'aller circuler sur
l'internet (pour peut que la table de routage de ces machines sont bien configurées.
et ayant comme ip sur le net 201.180.0.37.

Mais le problème, c'est que chez vous, vous avez un accès ppp (dial up), adsl .. ou
alors que vous ne connaissez pas votre ip tout le temps, et vous voulez que ca
soit automatique. Si votre interface réseau est tun0 (ppp sous les *bsd), alors
utilisez:

```
map tun0 192.168.0.0/24 -> 0/32
```

ipnat translatera tout seul, et tout marchera bien.

Maintenant, vous voulez un peu voir ce qui se passe, et comment les connections (et
lequelles) sont translatées. Il est possible pour cela de configurer ipnat, afin qu'
utilise tels et tels ports tcp/udp lors des translations:

```
map tun0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:30000
```

Grace au mot clé portmap, les connections translatées utiliseront les ports 20000 à
30000.

Maintenant, vous voulez pouvoir mapper des machines sur le réseau, et pas d'autre. C
utilisera pour cela les actions 'map' et 'map-block'

```
map tun0 192.168.0.0/16 -> 0/32
map-block tun0 192.168.1.0/24 -> 0/24
```

L'exemple ci dessus permet le mappage de toutes les machines de la classe
192.168.0.0/16, mais interdis les machines du sous réseau 192.168.1.0/24

On est riche, on a plusieurs IPs sur le net. Il est possible de ne plus mapper sur u
seule adresse sur le net, mais sur tout un tas grace à:

```
map tun0 192.168.0.0/16 -> 201.180.0.0/24
```

Comme cela, tout ce qui sortira prendra l'une des ips dans 201.180.0.0/24 que la
passerelle proposera selon sa configuration.

On peut aussi bi-mapper 2 adresses. Si l'on a 2 réseaux interconnectés, et que l'on
une machine sur l'un des 2 réseaux qui doit avoir une ip sur l'une (par ex:
192.168.0.17 et un autre sur l'autre: 201.180.0.17), on peut utiliser bimap:

```
bimap xl0 192.168.0.17/32 -> 201.180.0.17/32
```

Admettons maintenant que vous avez une IP fixe sur le net, et disposer de plusieurs
machines. Vous voulez qu'une machine fasse serveur web, l'autre serveur nntp,
par exemple. Vous voulez aussi faire de la sécurité, le serveur web sur le 1er serve
ne sera pas sur le port 80, mais 8080, tout en gardant la transparence
d'utilisation pour l'utilisateur:

```
rdr xl0 201.180.0.37/32 port 80 -> 192.168.0.12/32 port 8080 tcp
rdr xl0 201.180.0.37/32 port 113 -> 192.168.0.14/32 port 1113 tcp
```

Quand quelqu'un voudra aller sur le serveur web du 201.180.0.37, il se retrouvera de
facon transparente sur le serveur 192.168.0.12, et sur le port 8080. De meme
pour le serveur de news. On a précisé 'tcp' à la fin, pour le traffic en tcp, mais c
n'est aucunement obligé de préciser le protocole.

Cependant, on ne peut pas utiliser rdr pour l'exemple suivant:

```
rdr tun0 201.180.0.5/32 port 80 -> 201.180.0.6 port 80 tcp
```

Il n'y a plus de translation, ce qui fait que ici ipnat ne sera d'aucune aide.

On joue maintenant le role d'un gros provider, qui possède un site web pour ses
abonnés. Le problème est que le nombre d'abonné qui se connecte en meme temps
sur le site est trop important, on veut donc dire: cette partie d'abonné ira sur cet
machine, l'autre partie sur celle la. On le fait grace a 'from mask':

```
rdr xl0 from 201.181.0.0/16 201.180.0.17/32 port 80 -> 192.168.0.5 port 8080
rdr xl0 from 201.182.0.0/16 201.180.0.17/32 port 80 -> 192.168.0.6 port 8080
```

Autre exemple: si notre serveur est un pot de miel (honey pot: serveur mal configuré
pour voir comment un intrus réagit fasse à ce genre de machine), et que l'on
protège un réseau composé de machines sous windows, et l'on étudie le fonctionnement
Back Orifice (un de ces chevaux de troie sous Windows). On ne veut
pas que quelqu'un touche à l'un des pc du réseau, alors on redirige toutes les requè
vers son port (31337) vers le port 31337 de l'interface loopback:

```
rdr tun0 201.180.0.0/24 port 31337 -> 127.0.0.1 port 31337 udp
```

8.3 Note pour IPF: le Ftp

Le ftp n'est pas si simple à gérer lors d'une configuration ipf; étant donné qu'il

utilise 2 modes différents de fonctionnement, on doit maintenant expliquer comment avoir une bonne configuration et avoir le ftp qui marche.

Dans le cas d'un serveur FTP

Le FTP possède 2 modes de fonctionnement: le mode normal (port mode) et le mode pass
Le mode normal marche comme cela:

le client se connecte sur le serveur FTP, sur le port 21. Lors de la transmission de
donnée, c'est le port 20 du serveur qui envoit les données (celle des fichiers
transférés, des listages de répertoires ...) Testons sur un réseau:

```
virgo:/usr/home/patrick$ ftp 192.168.0.1
Connected to 192.168.0.1.
220 ProFTPD 1.2.0 Server (ProFTPD Default Installation) [virginie.toone.org]
Name (192.168.0.1:patrick):
331 Password required for patrick.
Password:
230 User patrick logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
drwxr-xr-x   4 patrick  Users         512 Feb  4 09:41 dev
drwxr-xr-x  14 patrick  Users         512 Feb  3 11:18 doc
drwxr-xr-x   4 patrick  Users         512 Feb  3 11:18 fun
-rw-------   1 patrick  Users       75753 Jan 29 17:44 mbox
drwxr-xr-x   9 patrick  Users        1024 Jan 13 10:37 package
drwxr-xr-x   6 patrick  Users         512 Feb  3 19:56 prog
drwxr-xr-x   3 patrick  Users         512 Jan 28 12:35 public_html
drwxr-xr-x   3 patrick  Users         512 Feb  3 11:17 sav
-rw-r--r--   1 patrick  Users        1996 Feb  4 09:38 tODO
226 Transfer complete.
ftp> quit
221 Goodbye.
virgo:/usr/home/patrick$
```

En meme temps:

```
virginie:/usr/home/patrick$ tcpdump
tcpdump: listening on xl0
# Initialisation de la connection TCP de virgo.toone.org vers le port 21 du
serveur FTP:
11:17:34.498462 virgo.toone.org.1027 > virginie.toone.org.ftp: S 108462:108462(
win 8192 <mss 1460> (DF)
11:17:34.498592 virginie.toone.org.ftp > virgo.toone.org.1027: S
512014860:512014860(0) ack 108463 win 17520 <mss 1460> (DF)
11:17:34.498808 virgo.toone.org.1027 > virginie.toone.org.ftp: . ack 1 win 8760
(DF)

# Requete sur le serveur Ident du client (ici terminée par un Reset, le serveur
marchait pas)
11:17:34.510180 virginie.toone.org.4432 > virgo.toone.org.auth: S
512049343:512049343(0) win 16384 <mss 1460> (DF)
11:17:34.510387 virgo.toone.org.auth > virginie.toone.org.4432: R 0:0(0) ack
512049344 win 0

# Le serveur demande le login:
11:17:34.510804 virginie.toone.org.ftp > virgo.toone.org.1027: P 1:79(78) ack 1
win 17520 (DF)
11:17:34.612422 virgo.toone.org.1027 > virginie.toone.org.ftp: . ack 79 win 868
(DF)
```

```
     11:17:35.838781 virgo.toone.org.1027 > virginie.toone.org.ftp: P 1:15(14) ack 7
win 8682 (DF)

     # Le serveur demande le mot de passe:
     11:17:35.840202 virginie.toone.org.ftp > virgo.toone.org.1027: P 79:115(36) ack
win 17520 (DF)
     11:17:36.024281 virgo.toone.org.1027 > virginie.toone.org.ftp: . ack 115 win 86
(DF)
     11:17:37.721296 virgo.toone.org.1027 > virginie.toone.org.ftp: P 15:30(15) ack
win 8646 (DF)

     # affichage 230 User patrick logged in. / Remote system type is UNIX. / Using
binary mode to transfer files.
     11:17:37.735759 virginie.toone.org.ftp > virgo.toone.org.1027: P 115:144(29) ac
30 win 17520 (DF)

     # le client envoit la commande ls (pour lister le répertoire):
     11:17:37.924124 virgo.toone.org.1027 > virginie.toone.org.ftp: . ack 144 win 86
(DF)
     11:17:38.373389 virgo.toone.org.1027 > virginie.toone.org.ftp: P 30:52(22) ack
win 8617 (DF)

     # le serveur renvoit la commande port pour prévenir le client de l'envoie de
donnée sur un autre port.
     11:17:38.373894 virginie.toone.org.ftp > virgo.toone.org.1027: P 144:174(30) ac
52 win 17520 (DF)
     11:17:38.390148 virgo.toone.org.1027 > virginie.toone.org.ftp: P 52:58(6) ack 1
win 8587 (DF)

     # Comme le client a demandé le transfert de donnée (commande ls), le serveur do
initialiser avec le client
     # une connection entre le serveur port 20 (ftp-data), et un port libre du clien
(ici 1028):
     # TCP handshake:
     11:17:38.391045 virginie.toone.org.ftp-data > virgo.toone.org.1028: S
513175670:513175670(0) win 16384 <mss 1460> (DF)
     11:17:38.391318 virgo.toone.org.1028 > virginie.toone.org.ftp-data: S
112355:112355(0) ack 513175671 win 8760 <mss 1460> (DF>
     11:17:38.391369 virginie.toone.org.ftp-data > virgo.toone.org.1028: . ack 1 win
17520 (DF)

     # Suit l'envoit des données, et la coupure de connection...

On a donc 2 flux principaux à faire passer, et dans le bon sens: (le serveur ftp est
sur 201.180.0.127)

     # pour l'initialisation (et keep state) client -> serveur:21
     pass in quick on xl0 proto tcp from any to 201.180.0.127 port = 21 flags S keep
state
     # pour l'initialisation (et keep state)
     pass in quick on xl1 proto tcp from 201.180.0.127 port = 20 to any flags S keep
state

Dans le cas d'un serveur FTP en passif

Le FTP en mode passif ne marche pas de la meme facon que le mode port. Dans le cas d
passif, le serveur dit au client de se connecter sur un autre port (> 1023)
afin de pouvoir proceder a l'envoie de données.

On n'a donc qu'à faire:

     # pour l'initialisation (et keep state) client -> serveur:21
     pass in quick on xl0 proto tcp from any to 201.180.0.127 port = 21 flags S keep
```

state
     # pour l'initialisation (et keep state)
     pass in quick on xl1 proto tcp from any to 201.180.0.127 port > 1023 flags S ke
state

Mais, il est probable que laisser tout les ports > à 1023 ouverts de l'extérieur ver
l'intérieur n'est pas la meilleure solution. (nfsd et X utilisent aussi des ports >
1023). Si votre serveur FTP le permet, vous pouvez restreindre les ports qui peuvent
etre utilisés. (ex: wu-ftpd le permet, grace à l'option 'passive ports' dans
ftpaccess). Si l'on veut par exemple restreindre ces ports à 15001-19999 inclus:

     pass in quick on xl0 proto tcp from any to 201.180.0.127 port = 21 flags S keep
state
     pass in quick on xl1 proto tcp from any to 201.180.0.127 port 15000 >< 20000 to
any flags S keep state

Dans le cas du client FTP

A cause des modes passif et normal, on a aussi pour le client ftp à faire face à ces
problèmes. Si l'on avait que le mode passif, cela serait simple, on n'aurait qu'à fa
un pass out sur tout et c'est tout: (201.180.0.10 = machine cliente, le serveur est
l'internet, on ne sait pas ou.)

     pass in quick on xl1 proto tcp from 201.180.0.10 flags S keep state

Malheureusement, l'existance du mode port nous oblige à accepter les connections sur
port 20 de l'extérieur.

     pass in quick on xl0 proto tcp from any port = 20 to 201.180.0.10 flags S keep
state

Finalement, il faut préciser que le mode port ne marchera pas si vous etes derrière
passerelle en NAT. Si vous voulez faire du FTP en mode port, alors il faudra
une machine avec une IP à vous sur le net.

8.4 Se protéger contre un DoS courant: le 'smurf'

Qu'est ce que le smurf ?

Récemment utilisé contre de grands fournisseurs de services comme Yahoo, Amazone
(99/2000) ... et contre un grand réseau IRC (fin 2000) (ce qui conduit
d'ailleurs à la fermeture de serveurs), le smurf est une arme redoutable quand elle
bien mise en oeuvre. Cette arme permet soit de faire planter des machines
(dans de rares cas), mais les effets les plus courrants est une surcharge du CPU des
machines touchées, ou la saturation de la connection. (il est facile de faire sature
jusqu'à déconnection une liaison modem, cable, T1 ...). Pour cela, il suffit d'envoy
un large nombre de requete ICMP, et à partir d'un grand nombre de machines
différentes.

Comment se protéger ? / Comment ne pas utiliser mes machines pour smurfer ?

Malheureusement, il est quasiment impossible de prévoir ce genre d'attaques. On peut
néanmois, afin que les machines critiques ne soient pas touchées, filtrer tout les
icmps rentrant (requètes/echo)

     block in quick on xl0 proto icmp from any to 201.180.0.0/24

Il faut savoir que si notre routeur est mal configuré, il peut servir de relais à ce
pratiques, et cela nécessite donc une reconfiguration si besoin.

9. Exemple 1: fichier de configuration pendant l'étude d'ipf

Voila le 'firewall' que nous avions à la fin de l'étude d'ipf à l'iut.

```
     #
     # Règles IPF pour firewall BSD

     #
     # Conf réseau:
     #

     #
     #
     #  /-------/      /-------/      /---/ /---/
     #  |Station|------|Avignon|------|   NET   |
     #  /-------/      /-------/      /---/ /---/
     #
     #


     #
     # Config Avignon: (station)--(ne0/ep0)--(autres)
     # ne0 a comme ip 172.20.4.15
     #

     # Station a une config réseau:
     #  eth0:0 inet:172.20.4.11
     #  eth0:1 inet:172.20.4.111
     #
     # Table de routage:
     # 172.20.0.0/16 eth0:0
     # 172.20.4.0/24 eth0:1
     # default/0 eth0:0 (gw 172.20.0.1)
     #


     # On drop tout ce qui doit pas arriver

     block in log quick on ep0 from 172.20.4.11/32 to any
     block in log quick on ep0 from 172.20.4.111/32 to any
     block in log quick from 192.168.0.0/16 to any
     block in log quick from 10.0.0.0/8 to any
     block in log quick on ep0 from 127.0.0.0/8 to any
     block in log quick on ne0 from 127.0.0.0/8 to any

     # machines n'existant pas dans le réseau

     block in log quick from 172.20.1.0/24 to any
     block in log quick from 172.20.6.0/24 to any

     # eux, on les aime pas, on les droppe proprement

     block return-rst in log quick proto tcp from 172.20.9.0/24 to any
     block return-rst in log quick proto tcp from 172.20.100.0/24 to any
     block return-icmp(port-unr) in log quick proto udp from 172.20.9.0/24 to any
     block return-icmp(port-unr) in log quick proto udp from 172.20.100.0/24 to any
     block return-icmp-as-dest(port-unr) in log quick proto icmp from 172.20.9.0/24
any
     block return-icmp-as-dest(port-unr) in log quick proto icmp from 172.20.100.0/2
to any

     # On a fait le tri. Maintenant, on devrait avoir quelque chose d'existant.

     # regle par défaut (pour être sur, je rajoute à la fin 'block in quick all')

     block in all
```

```
     # icmp (on s'alloue les requetes partant de notre station, et uniquement ca

     pass in     quick on ne0 proto icmp from 172.20.4.11/32 to 172.20.0.0/18 icmp-t
8 keep state
     pass in     quick on ne0 proto icmp from 172.20.4.111/32 to 172.20.4.0/24
icmp-type 8 keep state

     # on rejete ce qui reste, (vérifier dans le tcp/illustré si le code de
     # retour est bon)
     block return-icmp-as-dest(port-unr) in log quick on ep0 proto icmp from any to


     # Traffic DNS
     # serveurs DNS utilisés: etud (172.20.0.22)
     #                        admin (172.20.0.21)

     pass in     quick on ne0 proto udp from 172.20.4.11/32 to 172.20.0.21/32 port =
keep state
     pass in     quick on ne0 proto udp from 172.20.4.11/32 to 172.20.0.22/32 port =
keep state

     # // Différence avec la version non-paranoique. On ne s'alloue plus tout les
     # // droits, mais on laisse passer vers l'extérieur que ce que l'on désire.

     # Client FTP (vers etud, vers admin, et vers le net,
     # On drop tout 172.20.0.0/16, sauf serveur et 172.20.4.0/24
     # Pas de passif.

     pass in     quick on ne0 from 172.20.4.11/32 to 172.20.0.21/32 port = 21 flags
keep state keep frags
     pass in     quick on ne0from 172.20.4.11/32 to 172.20.0.21/32 port = 20 flags S
keep state keep frags

     pass in     quick on ne0 from 172.20.4.11/32 to 172.20.0.22/32 port = 21 flags
keep state keep frags
     pass in     quick on ne0 from 172.20.4.11/32 to 172.20.0.22/32 port = 20 flags
keep state keep frags

     pass in     quick on ne0 from 172.20.4.111/32 to 172.20.4.0/24 port = 21 flags
keep state keep frags
     pass in     quick on ne0 from 172.20.4.111/32 to 172.20.4.0/24 port = 20 flags
keep state keep frags

     block return-rst in    quick from 172.20.4.11/32 to 172.20.0.0/16 port = 21
     block return-rst in    quick from 172.20.4.11/32 to 172.20.0.0/16 port = 20

     pass in     quick on ne0 from 172.20.4.11/32 to any port = 21 flags S keep stat
keep frags
     pass in     quick on ne0 from 172.20.4.11/32 to any port = 20 flags S keep stat
keep frags

     # Client SSH
     # nous -> etud, nous -> 172.20.4.0/24 nous -> planet-work

     pass in     quick on ne0 from 172.20.4.11/32 to 172.20.0.22/32 port = 22 flags
keep state keep frags
     pass in     quick on ne0 from 172.20.4.111/32 to 172.20.4.0/24 port = 22 flags
keep state keep frags
     pass in     quick on ne0 from 172.20.4.11/32 to box1.planet-work.com/32 port =
flags S keep state keep frags
     block in log quick from 172.20.4.11/32 to 172.20.0.21 port = 22
     # derniere ligne, pour logguer les tentatives de connection ou l'on a pas le
```

```
     # droit.

     # Client Web
     pass in    quick on ne0 from 172.20.4.111/32 to 172.20.4.0/24 port = 80 flags S
keep state keep frags
     pass in    quick on ne0 from 172.20.4.11/32 to 172.20.0.21 port = 80 flags S ke
state keep frags
     pass in    quick on ne0 from 172.20.4.11/32 to 172.20.0.22 port = 80 flags S ke
state keep frags
     block return-rst in log quick on ne0 from 172.20.4.11/32 to 172.20.0.0/16 port
80
     pass in    quick on ne0 from 172.20.4.11/32 to any port = 80 flags S keep state
keep frags

     # Client Web/443
     block return-rst in log quick from 172.20.4.11/32 to 172.20.0.0/16 port = 443
     pass in    quick on ne0 from 172.20.4.11/32 to any port = 443 flags S keep stat
keep frags

     # Client Mail (relay vers 172.20.0.21)
     pass in     quick on ne0 from 172.20.4.11/32 to 172.20.0.21/32 port = 25 flags
keep state keep frags

     # Traffic Windows (à étudier, quand on sera sous NT) (Netbios power)

     pass in     quick from 172.20.0.0/16 to 172.20.0.0/16 port = 137
     pass in     quick from 172.20.0.0/16 to 172.20.0.0/16 port = 138
     pass in     quick from 172.20.0.0/16 to 172.20.0.0/16 port = 139


     # Notre serveur ssh (etud <--> .4.11)
     pass in     quick on ep0 from 172.20.0.22 to 172.20.4.11 port = 22 keep state
flags S keep frags

     # Notre serveur FTP (etud <--> .4.11)
     pass in     quick on ep0 from 172.20.0.22 to 172.20.4.11 port = 21 keep state
flags S keep frags
     pass in     quick on ep0 from 172.20.0.22 to 172.20.4.11 port = 20 keep state
flags S keep frags
     # le passif suce.

     # Notre serveur Web (.0.0/16 <--> .4.11)
     pass in     quick on ep0 from 172.20.0.0/16 to 172.20.4.11 port = 80 keep state
flags S keep frags

     # Mails
     pass in     quick on ep0 from 172.20.0.21/32 to 172.20.4.11 port = 25 keep stat
flags S keep frags

     # Paranoia propre
     block return-rst in quick proto tcp all
     block return-icmp-as-dest(port-unr) in quick proto udp all
     block return-icmp-as-dest(port-unr) in quick proto icmp all

     # Fin
     block in quick all

     10. Conclusion

10.1 Mot de la fin

Nous esperons que cette doc est le plus clair possible. Dans un soucis de perfectior
nous vous encourageons à nous signaler tout défauts, et toutes les remarques
```

constructives seront prises en comptes. (Nous sommes très ouverts :-)

10.2 Remerciements

Remerciements à nos professeurs de l'IUT d'Amiens (Somme/France), spécialement notre
chef de projet, David tilloy (d.tilloy@nnx.com), pour les connaissances
que nous avons acquises, et aussi pour l'ambiance d'enseignement.

Remerciements aux personnes qui nous ont permis de traiter à son bout ce sujet (et à
celles qui nous ont fournies les moyens techniques nécessaires.)

10.3 Bibliographie

Le site OpenSecurely (www.openlysecure.com)

IP Filter Based Firewalls HOWTO, de Brendan Conoboy <synk@swcp.com> et Erik Fichtner
<emf@obfuscation.org> (trouvalbe à:
http://www.obfuscation.org/ipf/)

TCP/IP illustré, volume 1, les protocoles de W. Richard Stevens

Le créateur d'IP filter :)

Building Internet Firewalls (Chapman & Zwicky, O'Reilly and Associates)

10.4 Copyrights