FIREWALL ET SÉCURITÉ D'UN RÉSEAU PERSONNEL SOUS LINUX

Olivier ALLARD-JACQUIN olivieraj@free.fr

Version 0.5.3 - 23 juillet 2003

Ce document a pour but d'expliquer les rudiments de la sécurité d'une machine Linux placée dans un réseau local (typiquement une maison on un appartement), reliée à Internet. Il est tout particulièrement destiné aux utilisateurs néophytes, ou n'ayant pas ou peu de connaissance sur la sécurité informatique en général, et sous Linux en particulier.

Concernant ce document, il aurait été possible de le faire de manière beaucoup plus synthétique. Mais il n'était pas mon but de faire une documentation trop ardue à comprendre... Donc si vous trouvez que je suis trop long dans mes explications ou que je me répète, je vous autorise à en arrêter la lecture !

Vous trouverez la dernière version de ce document :

- sur son site principal : http://olivieraj.free.fr/fr/linux/information/firewall/.
- sur le miroir de la Guilde.
- en version mono-page (très long à télécharger...), en cliquant sur l'icone situé en haut et à droite du document.
- en version PDF.
- sous forme d'archive complète, au format .tgz (.tar.gz), afin de lire en hors ligne.

Ce document a été écrit en parallèle d'une conférence organisée par la Guilde le 2 juillet 2003. Vous pouvez télécharger les transparents de la présentation au format OpenOffice.org IMPRESS (.sxi), Powerpoint (.ppt), ou les consulter directement en ligne.

INTRODUCTION

Ce document est assez long à lire (il l'a été encore plus à concevoir), aussi ne perdez pas patience au fur et à mesure des paragraphes. Le style utilisé s'est voulu le plus didactique et progressif possible, afin que les utilisateurs ayant le moins de connaissance de Linux puisse comprendre les concepts décrit ici.

Pourquoi parler de la sécurité d'un système informatique ? Cela a t' il une réelle importance ? Où n'est-ce là qu'un sujet purement théorique, issus des cerveaux fébriles de quelques paranoïaques qui voient le mal partout ?

Dans un premier temps, vous trouverez la réponse à cette question en vous posant vous même ces quelques questions. Est-ce que vous laisseriez votre maison, toutes portes et fenêtres ouvertes, à n'importe quelle heure du jour et de la nuit ? Que vous soyez ou pas présent ? Sans doute non, car votre domicile contient des biens qui pourraient tenter un voleur ou un vandale. Par contre, est-ce que vous laisseriez sur une plage votre serviette sans surveillance, pendant que vous iriez piquer une tête dans la mer ? Peut-être bien que oui ?

En informatique, et du moment que la machine que vous utilisez est connectée en réseau, c'est un peu la même chose. En fonction de votre lieu de connexion, de la machine que vous utilisez, et de l'importance des documents que vous pouvez accéder grâce à elle, vous serez plus où moins sensibilisé à ces problèmes de sécurité.

En effet, il existe sur les réseaux, et tout particulièrement sur Internet, des personnes peu scrupuleuses qui voudront atteindre, pour une multitude de raisons différentes, le contenu de votre machine. Cela va du "script kiddy" (un néophyte en terme de piratage informatique) qui a téléchargé de puissants outils de recherche de vulnérabilité, et qui, à défaut de les comprendre, les utilise afin d'épater ses amis ou rencontres sur IRC. Jusqu'à la grosse multinational du logiciel ou de la société de publicité, qui rechercheront diverses informations personnelles vous concernant, histoire de les revendre en vu d'envois massifs de publicités, ou à d'autres fins encore moins avouables...

Par contre, dans un second temps les problèmes de sécurité nous concerne tous, qui que nous soyons. En effet, que diriez vous d'une personne qui irait emprunter votre voiture, faire quelques délits, puis vous la rendrait sans que vous ne vous en aperceviez ? Dans le moindre des cas, personne (vous notamment) ne se sera rendu compte des délits. Dans le pire des cas, on viendra vous demander des comptes sur vos agissements, ou plus exactement de ceux du conducteur de votre voiture... Et vous risqueriez d'avoir du mal à expliquer la situation.

Là encore, l'informatique se trouve face aux mêmes problèmes que la vie de tout les jours. En effet, des personnes peuvent s'amuser à prendre le contrôle de votre machine, et lui faire faire des actions peu recommandables : Saturation d'une machine à distance via une attaque DOS distribuée (comme c'est régulièrement le cas avec des virus-vers qui sévicent sur Internet). Attaque par rebond d'un serveur d'une grosse société commerciale, ou d'une administration. Que sais je encore... Une chose en tout cas est sûre : Quelque soit les actions perpétrées par l'intrus, c'est vous, propriétaire d'une machine connectée sur un réseau, que l'on accusera...

Ce tableau de la situation n'est pas particulièrement réjouissant, n'est-ce pas ? Mais il n'est pas non plus inutilement alarmiste, et il n'a pas pour but de vous faire acheter au plus vite la dernière suite de logiciels de protection (firewall, anti-virus, anti-spyware, ou que sais je encore...). Ce n'est que la réalité, un peu froide et brutale de la situation. Mais maintenant que vous avez conscience de l'importance de la sécurité en informatique, nous allons pouvoir passer à la mise en place de remèdes. Car ils existent !

Afin de parler de la personne externe à votre machine qui désire y accéder sans votre autorisation, j'utiliserai dans la suite de ce document le terme d'intrus (même si celui-ci peut se révéler être du sexe féminin, ce type de comportement n'était pas réservé uniquement à la gent masculine). Je me refuse en effet à utiliser le mot de hacker, dont je préfère garder le sens premier. Quand à utiliser le mot de cracker, celui-ci a tellement de définitions différentes, qu'il n'en n'est pas une d'assez bien établi pour l'utiliser dans ce contexte. Et je préfère ne pas non plus utiliser pour les nommer, la multitude de noms d'oiseaux qui me passe par la tête...

En choisissant d'utiliser Linux comme système d'exploitation (ou OS pour "Operating System" en anglais), vous n'avez pas fait le pire choix en terme de sécurité informatique. Et surtout, contrairement à un autre OS bien connu du grand public, vous n'aurez pas à passer votre temps à vous poser l'angoissante question "Mais qu'est-ce que fait mon ordinateur ? Est-ce qu'il n'est pas entrain de diffuser des informations sans mon autorisation". De plus, sous Linux la plus grande partie des logiciels que vous utilisez sont sous la licence GPL, ce qui vous permet d'accéder à leurs codes sources, afin de savoir de quoi il en ressort exactement. Et si vous ne le pouvez / voulez pas, d'autres personnes peuvent s'en charger à votre place.

Dans les chapitres qui vont suivre, nous commencerons par un rapide rappel des notions de bases du réseau, puis nous verrons comment ne pas provoquer l'intrus, en ne l'invitant pas à passer par des portes grandes ouvertes pour commettre ses méfaits. Dans un troisième temps, nous verrons comment mettre en place le firewall de Linux (l'élément essentiel de la protection de votre machine) sur des kernels 2.4 et supérieurs, puis quelques outils permettant de faciliter son utilisation.

Installez vous confortablement, prenez un verre de votre boisson favorite à cette heure ci, et accrochez votre ceinture. C'est parti pour un plongeon dans les limbes de votre Linux !

I RAPPELS SUR LES RÉSEAUX IP

PLAN :

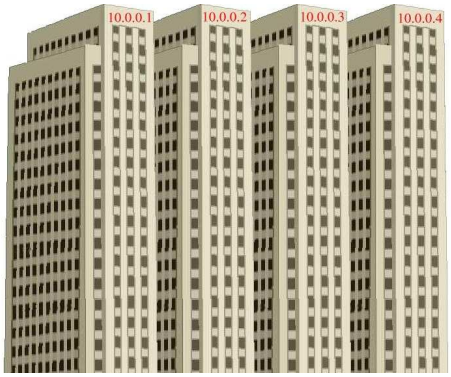
Comme l'indique le titre de cette page, nous allons voir ici quelques notions de base sur les réseaux IP. Ici, je ne vous parlerai pas de modèle OSI, d'encapsulation de couches, de sockets, d'ARP, et de tant d'autres choses. Je me bornerai à vous expliquer les quelques rudiments de base nécessaires à la compréhension des chapitres suivants. Cet exposé se limitera au strict cadre des réseaux IP, même si il existe de nombreux autres types de réseaux (NETBeui, IPX/SPX, Transpact, etc...)

Si vous maîtrisez déjà les concepts d'adresses IP, de ports, de masque de sous-réseaux, de réseau IP, de passerelle, de type de trames, de la "handshake", et des interfaces réseaux, je vous invite à passer directement au chapitre suivant. Sinon nous allons commencer à assimiler ces notions.

Si par la suite vous désirez en apprendre un peu plus sur les réseaux IP, je vous conseille la lecture de quelques sites intéressants qui parleront de ce sujet bien plus en détail que moi. Je vous conseille tout particulièrement ceux de Christian Caleca et de CommentCaMarche.net.

I-1 Adresses IP

Imaginez que les machines d'un réseau sont des immeubles complètement uniformes, et qu'ils forment une petite ville. Pour aller d'un immeuble à un autre, le seul moyen à votre disposition est alors d'en connaître l'adresse exacte. En France par exemple, nous utilisons un système de rues associées à un numéro, qui permettent bon an mal an, de trouver plus ou moins par hasard son lieu de destination. Aux Etats-Unis par contre, l'organisation des grandes villes se fait généralement en damier, ce qui permet d'identifier les immeubles avec une combinaison de numéros d'avenues / rues, ce qui est un poil plus pratique.



Dans notre "ville réseau", chaque machine possède une adresse IP, qui l'identifie de manière unique. Cette adresse est composée de 4 chiffres numérotés de 0 à 255 séparés par des points ("."). Par exemple 62.158.78.243.

L'élément le plus important d' IP, c'est que dans un même réseau, deux machines ne doivent pas avoir la même adresse IP, sinon, il serait impossible de communiquer correctement avec celles-ci.

Par contre, l'inverse n'est pas vrai : Une machine peut avoir plusieurs adresses IP.

Pour contacter une machine d'un réseau, il faut impérativement connaître son adresse IP. Mais comme il n'est pas très pratique de se souvenir de ces adresses, un système appelé DNS ("Domain Name Server") permet d'associer un nom de machine un peu plus facilement mémorisable (par exemple " olivieraj.free.fr") à une adresse IP (exemple : 62.158.78.243). Mais je ne rentrerai pas plus dans ces détails, car ce n'est pas nécessaire à la compréhension de la suite du document.

Parmi toutes les adresses IP disponibles (environ 256*256*256*256 = 4 294 967 296), il y a 3 catégories d'adresses IP très particulières. Ce sont les classes IP privées . Dans un réseau privé, comme celui d'une entreprise ou chez un particulier, on peut utiliser ces classes d'adresses IP en toute liberté, sans avoir de compte à rendre à personne. La seul condition est que les machines ayant une adresse IP d'une classe privée ne peuvent pas se connecter directement à Internet.

Les classes IP privées sont :

Nom de classe privée	Information	Nombre maximum de machines connectables
A	10.x.y.z, où : 0 <= x <= 255 0 <= y <= 255 0 <= z <= 255	(256*256*256) - 2 = 16 777 214
B	172.x.y.z, où : 16 <= x <= 31 0 <= y <= 255 0 <= z <= 255	(15*256*256) - 2 = 1 048 574
C	192.168.x.y, où : 0 <= x <= 255 0 <= y <= 255	(256*256) - 2 = 65 534

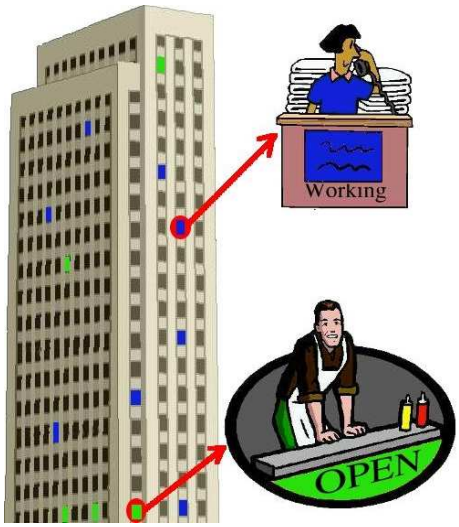
Enfin toute machine utilisant le protocole IP possède par défaut une série d'adresse IP privées, utilisable uniquement par la machine elle-même. Ainsi, toute machine peut s'adresser à elle-même en utilisant les 16 777 214 adresses IP comprises entre 127.0.0.1 et 127.255.255.255. C'est ce que l'on appelle les adresses de loopback.

A noter que l'on ne parle ici que des adresses IP du protocole IPv4. A l'heure actuelle (juin 2003), une nouvelle version de ce protocole (IPv6) est entrain de se déployer, qui à terme devrait remplacer IPv4. Le principe de fonctionnement de cette nouvelle version est grosso modo le même, mais elle apporte de nombreuses fonctionnalités qui la rendent plus intéressante. Mais, les détails d'IPv6 débordent largement du cadre de ce document...

I-2 Ports

Toujours en utilisant notre analogie ordinateur <-> immeuble, voyons maintenant la notion de ports.

En bas d'un immeuble, on peut trouver un certain nombre de commerces, ouverts ou fermés, ou vides. Et dans les étages, on trouvera un certain nombre de fenêtres : certaines ouvertes ou fermées, d'autre avec des habitants entrain de parler avec leurs voisins de l'immeuble d'en face (encore faut il qu'ils aient suffisamment de voix...)



Ports TCP ouverts, fermés, en cours d'utilisation

Sur un ordinateur, c'est un peu la même chose. Une machine fonctionnant sur un réseau IP possède 65535 ports de connexion (numérotés de 1 à 65535), équivalents aux magasins et fenêtres de notre immeuble. Tous ces ports sont interchangeables, c'est à dire que chacun peut servir à :

- Émettre des données.
- Recevoir des données.
- Émettre et recevoir des données à tour de rôle.

Lorsqu'un logiciel d'une machine cliente va vouloir accéder aux informations d'une machine serveur, ce logiciel va :

- ouvrir un port (comme un habitant ouvrirait une fenêtre de son immeuble).
- envoyer une requête à la machine serveur, sur le port qui héberge l'information qu'il désire.
- recevoir cette information via le port qu'il avait précédemment ouvert (*).
- refermer son port.

(*) Remarque : Avec certains protocoles particuliers, comme le FTP, l'information demandé arrivera par un 2nd port ouvert sur la machine cliente.

Par contre, en aucun cas un port d'une même machine ne peut être utilisé par deux logiciels différents. Par analogie, on ne peut pas avoir deux habitants utilisant en même temps la même fenêtre pour parler avec ses voisins d'en face.

D'une manière générale :

- Les ports dont le numéro est inférieur à 1024 serviront à recevoir des informations. En fait, derrière ces ports se trouverons des logiciels de type serveur (et que l'on appellera par la suite tout simplement "serveur") qui attendrons des requêtes d'autres ordinateurs, afin de fournir une certaine information. Ces ports là sont comme les portes de boutiques dans lesquelles attendent des commerçants. Voici quelques exemples de ports et leur utilisation type :

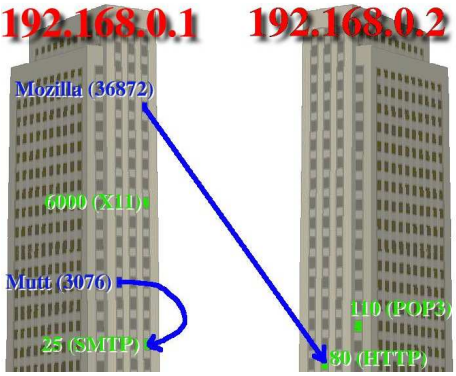
Numéro du port	Type de service
21	FTP : File Transfert Protocol (Échange de fichiers)
23	Telnet : Terminal en mode texte
25	SMTP : Simple Mail Transfert Protocol (envoi de mails)
80	HTTP : HyperText Transfert Protocol (le web, que vous utilisez lorsque vous tapez "http://www...." dans votre navigateur)
110	POP3 : Réception de mails

Cependant, et comme écrit plus haut, n'importe quel port peut abriter un serveur, y compris ceux supérieur à 1024. Ainsi sur une machine Linux/Unix, les ports 6000 et au-delà servent au serveur graphique X11.

De même, ce n'est pas parce qu'un port est actif sur une machine servant de serveur que forcément celui-ci sert à un usage précis. On peut en effet très bien déclarer sur une machine un serveur logiciel HTTP sur le port 21, au lieu du 80. Simplement, il faudra que le logiciel qui se connecte sur cette machine sache que, si il veut avoir du HTTP, il doit contacter le port 21, et non le port 80.

Pour résumer : Il y a des règles dans la déclaration des ports utilisés par les serveurs logiciels, mais celles-ci peuvent être ignorées.

- Les ports compris entre 32768 et 61000 sont généralement utilisés par les logiciels clients pour dialoguer avec les ports serveurs d'une autre machine.
- Les autres ports peuvent servir à n'importe quel usage (serveur ou client)...



Une erreur qui est communément faite à propos des ports, est de penser qu'un logiciel client doit utiliser un port symétrique par rapport au serveur qu'il contacte. Par exemple on peut penser qu'un navigateur web (HTTP donc) doit ouvrir son propre port 80 pour accéder au serveur web d'une machine distante (toujours sur le port 80). Comme vu précédemment, ceci est complètement faux ! En effet, pour se connecter au port 80 du serveur, le client peut ouvrir un autre port, par exemple le 32768. Et il recevra la page HTML qu'il a demandé sur ce même port.

En fait, c'est très logique car :

- Le logiciel client pourrait fort bien vouloir accéder à un logiciel serveur qui se trouve sur la même machine. Et dans ce cas là, le port 80 ne peut pas servir à la fois pour le logiciel serveur et le logiciel client.
- Sur une même machine, plusieurs logiciels peuvent accéder à plusieurs serveurs en même temps, en utilisant des services identiques. Par exemple, vous pouvez très bien faire une recherche sur www.google.fr tout en lisant des informations sur linuxfr.org, et en téléchargeant la dernière version de la distribution Debian sur www.debian.org.

Enfin, vous trouverez une liste (non exhaustive) des principaux ports IP en consultant le fichier "/etc/services" sur votre Linux. A toutes fins utiles, en voici une à jour au 1er janvier 2003.
Pour ce qui est de la définition des termes TCP et UDP de ce fichier, vous trouverez l'explication un peu plus loin dans ce chapitre.

I-3 Masque de sous-réseaux

C'est une partie un peu plus compliquée que précédemment, alors accrochez vous...

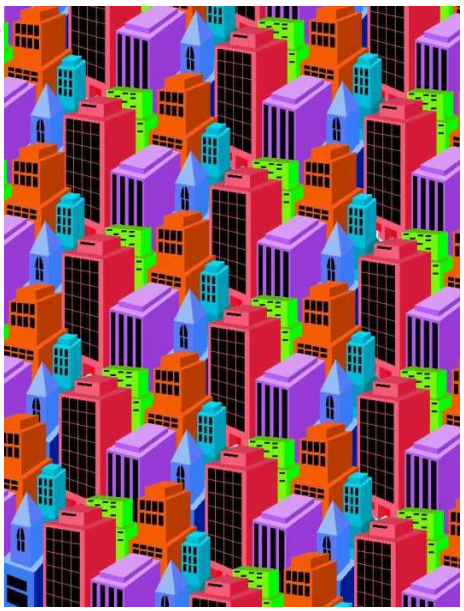
Imaginez que tous les immeubles de la terre soient regroupés en une seul et même immense ville. Même avec un quadrillage précis de l'implantation des bâtiments, l'échange d'informations entre 2 immeubles éloignés seraient excessivement long, car il faudrait parcourir une grand partie de la ville pour ne serait-ce que trouver son correspondant. De plus, toutes ces connexions dans tout les sens formeraient un brouhaha insupportable, qui altérerait la qualité des communications.

A la place de cela, il vaut mieux des villes plus petites (villes, villages, quartiers, lotissements), séparées par des longues routes, où les gens se regroupent par affinité ou par besoin. J'arrêterai là l'analogie, car après, le modèle de réseau IP s'éloigne un peu trop de la réalité des villes...

Donc pour faire simple, les machines d'un réseau IP sont regroupées entre elles dans des mini-réseaux. Mais comment fait on pour déterminer la taille de ce réseau, et si oui ou non une machine fait parti du même réseau que nous ? La réponse est simple : Le masque de sous-réseau , appelé aussi masque IP, ou tout simplement masque.

Tout comme l'adresse IP, le masque de sous-réseau est composé de 4 chiffres séparés par des points ("."). Mais contrairement aux chiffres des adresses IP, les chiffres du masque IP ne peuvent avoir que certaines valeurs : 0, 128, 192, 224, 240, 248, 252, 254, 255. Mais dans la plupart des cas, on ne trouve que les valeurs "0" et "255". Par exemple : 255.0.0.0 ou 255.255.255.0.

Comme on l'a vu précédemment, chaque machine connectée à un réseau doit posséder une adresse IP unique dans le réseau. En plus de cela, toutes les machines d'un sous-réseau possèdent un masque de sous-réseau identique . C'est le couple adresse IP / masque de sous-réseau qui permet à chaque machine de s'identifier, et d'identifier les machines de son propre sous-réseau. En fait, c'est le masque IP qui va même déterminer le nombre maximum de machines pouvant se trouver dans un sous-réseau.


Réseau sans masque : Ingérable !!!

On a vu précédemment les 3 classes de réseaux privés, voyons maintenant quels sont les masques IP que l'on y trouve habituellement :

Classe réseau	Adresse IP	Masque de	Nombre de	Exemple d'adresses IP	Nombre de machines

privé		sous-réseaux	sous-réseaux		par réseau
A	10.x.y.z	255.0.0.0	1	Toutes les adresses IP comprises entre : 10.0.0.1 et 10.255.255.254	16 777 214
B	172.x.y.z Rappel : 16 < x < 31	255.255.0.0	16	16 réseaux différents, avec des adresses IP comprises entre : 172.16.0.1 et 172.16.255.254 172.17.0.1 et 172.17.255.254 ... 172.31.0.1 et 172.31.255.254	16 x 65 024
C	192.168.x.y	255.255.255.0	256	256 réseaux différents, avec des adresses IP comprises entre : 192.168.0.1 et 192.168.0.254 192.168.1.1 et 192.168.1.254 ... 192.168.255.1 et 192.168.255.254	256 x 254

L'explication du pourquoi et du comment marchent les masques IP est un peu trop complexe, et sort du cadre de ce chapitre. Pour l'aborder, il faudrait utiliser des notions de logique booléenne (XOR, bit, etc...), ce qui sera trop long à traiter. Si vous désirez plus d'informations à ce sujet, je vous invite à trouver plus de documentations sur Internet, et notamment ici.

I-4 Réseaux IP

Nous avons vu jusqu'à présent que chaque machine possédait une adresse IP et un masque de sous-réseau, permettant de définir les limites du réseau. Cela nous amène tout naturellement à parler de la notation de réseau IP.

Le réseau IP définit un ensemble d'adresses IP mitoyennes, exactement comme le ferait la bordure d'une ville telle que décrit plus haut. Le réseau IP est composé de 2 séries de 4 chiffres, séparés par un " /".

- La première série est la plus petite adresse IP du réseau, moins une unité. Par exemple, pour un réseau dont les machines ont une adresse IP variant de 192.168.0.1 à 192.168.0.254, ce chiffre sera 192.168.0.0 .
- La seconde série est tout simplement le masque de sous-réseau : Dans notre exemple, il s'agira de 255.255.255.0 .

Ainsi donc pour notre exemple précédent, les 254 machines dont les adresses IP varient de 192.168.0.1 à 192.168.0.254, et dont le masque de sous-réseau est 255.255.255.0, forment le réseau IP : 192.168.0.0 / 255.255.255.0 . C'est cette notation que l'on utilisera lorsque l'on voudra parler de ce réseau.

Machines groupées en réseaux IP

Enfin, il existe une seconde notation pour les réseaux IP, tout aussi utilisée que la première, mais plus pratique car plus rapide à écrire. Il s'agit de remplacer la notation du masque de sous-réseau (dans l'exemple; 255.255.255.0) par le nombre de bits à "1" de chaque nombres qui le compose. Il faut donc utiliser la notation binaire pour trouver ce chiffre. Pour ceux qui ont la flemme de calculer, ou qui n'ont pas de calculette sous la main, voici le nombre de bits à "0" et à "1" pour les différentes valeurs des nombres composant les masques de sous-réseau :

Nombre	Nombre de bit(s) "0"	Nombre de bit(s) "1"
0	8	0
128	7	1
192	6	2
224	5	3
240	4	4
248	3	5
252	2	6
254	1	7
255	0	8

Ce qui nous fait pour des masques de sous-réseaux que l'on trouve habituellement (les "..." indiquent qu'il y a évidement des valeurs intermédiaires) :

Masque de sous-réseaux	Nombre de bit(s) "1"
...	...
255.255.255.0	24
...	...
255.255.0.0	16
...	...
255.0.0.0	8
...	...

Ainsi, pour les différentes classes IP privées que nous avons à notre disposition, nous pouvons écrire leur notation en réseau IP :

Classe réseau privé	Réseau IP (1ère écriture)	Réseau IP (2nd écriture)	Nombre de sous-réseaux	Exemple d'adresses IP	Nombre de machines par réseau
A	10.0.0.0 / 255.0.0.0	10.0.0.0 / 8	1	De 10.0.0.1 à 10.255.255.254	16 777 214
B	172.16.0.0 / 255.255.0.0	172.16.0.0 / 16	1 / 16	De 172.16.0.1 à 172.16.255.254	65 024
	172.17.0.0 / 255.255.0.0	172.17.0.0 / 16	2 / 16	De 172.17.0.1 à 172.17.255.254	65 024
	...	...	...	...	65 024
	172.31.0.0 / 255.255.0.0	172.31.0.0 / 16	16 / 16	De 172.31.0.1 à 172.31.255.254	65 024
C	192.168.0.0 / 255.255.255.0	192.168.0.0 / 24	1 / 256	De 192.168.0.1 à 192.168.0.254	254
	192.168.1.0 / 255.255.255.0	192.168.1.0 / 24	2 / 256	De 192.168.1.1 à 192.168.1.254	254

...	...	...	...	254
192.168.255.0 / 255.255.255.0	192.168.255.0 / 24	256 / 256	De 192.168.255.1 à 192.168.255.254	254

I-5 Passerelle

Maintenant que nous avons abordé les notions de masque IP et de réseau IP, celle de passerelle n'est qu'une formalité. En reprenant notre analogie d'une petite ville entourée d'une barrière, on va considérer maintenant que cette barrière est infranchissable, et qu'elle ne possède qu'un seul et unique point de sortie, fermée par un garde barrière.

Lorsqu'un habitant d'un immeuble veut échanger des informations avec celui d'un autre immeuble, alors :

- soit l'immeuble en question se trouve dans la même zone, et dans ce cas l'habitant fera directement l'échange d'informations.
- soit l'habitant va donner l'information au garde barrière qui devra se débrouiller pour trouver l'immeuble de destination. Les détails de ce qui se passe après ne sont pas trop important. Ce qui compte c'est la réponse du garde barrière à l'habitant. Soit celui-ci a réussi à transmettre l'information, et donc il rapportera à l'habitant la réponse. Soit il n'a pas réussi à transmettre cette information dans un délai acceptable, auquel cas il le dira à l'utilisateur, qui prendra les décisions qui s'imposent.

La passerelle permet aux réseaux de communiquer entre eux

Et voila, nous venons d'expliquer le fonctionnement de la passerelle dans un réseau IP. Une passerelle, appelée gateway en anglais, n'est autre qu'une machine ayant une certaine adresse IP. Habituellement, il s'agit de la dernière adresse disponible dans un réseau, mais ce n'est pas une obligation. Par exemple pour un réseau 192.168.0.0 / 255.255.255.0, habituellement l'adresse IP de la passerelle est 192.168.0.254 .

Comme vu ci-dessus, chaque machine du réseau doit connaître l'adresse IP de la passerelle. Si celle-ci n'est pas renseignée, la machine considérera qu'il n'y a pas de passerelle sur le réseau. Et donc, si elle doit accéder à une machine dont l'adresse IP n'est pas dans le réseau local, elle renverra immédiatement un message d'erreur disant que la machine n'est pas accessible. C'est le fameux message "no host reachable" en anglais.

Bilan : Au point où nous en somme, nous avons appris qu'une machine dans un réseau IP,

- Possède une adresse IP unique dans ce réseau. Exemple : 192.168.0.1.
- Est dotée d'un masque de sous réseau, qui est commun avec les autres machines du réseau. Exemple : 255.255.255.0 .
- Peut ou non connaître l'adresse IP de la passerelle (gateway), seul point de sortie avec l'extérieur du réseau. Exemple : 192.168.0.254 .
- Le réseau qui englobe ces machines possède une dénomination explicite. Exemple : 192.168.0.0 / 255.255.255.0 ou 192.168.0.0 / 24 .

I-6 Type de trames

Afin de s'échanger des informations, les habitants de nos immeubles ne se parlent pas directement (on dira pour simplifier que leur voix ne porte pas assez loin...), mais ils échangent des lettres. Principalement trois différents types de lettres pour être exact.

Dans notre réseau IP, c'est la même chose : Les logiciels clients et serveurs vont utiliser en général 3 différents types de trames pour communiquer, TCP, UDP, et ICMP. La trame est donc un "emballage" qui va transporter les informations d'une adresse IP à une autre. Il existe d'autres type de trames, mais nous ne nous y intéresserons pas.

I-6-1 TCP

Il s'agit du type de trame le plus utilisé. C'est pourquoi par abus de langage, on parle très souvent de réseau TCP/IP plutôt que de réseau IP . En fait ce type de support ressemble un peu à un envoi de lettre par recommandé, où avant de commencer un échange d'information avec son interlocuteur, on enverra une lettre dont le destinataire devra lui-même envoyer une réponse disant qu'il a bien reçu le recommandé. C'est le mécanisme de la poignée de mains , qui sera expliqué un peu plus loin .

La seule chose à retenir, c'est que du fait de sa robustesse, ce type de trame est utilisé aussi bien sur les réseaux éloignés que proches. Par contre, son inconvénient est qu'il rajoute une certaine "lourdeur" à la transmission d'informations, ce qui ralenti un peu les réseaux.

I-6-2 UDP

Ce type de trame tranche franchement avec le précédent, car cette fois ci, l'expéditeur va supposer que le destinataire de l'information n'a rien d'autre à faire que de l'écouter, et donc il peut envoyer sans préavis sa requête d'information. C'est exactement le comportement de ces personnes qui, pendant que vous vous concentrez sur un sujet particulier, vous posent une question sans même commencer leur phrase par un "Je peux te poser une question ?" ou "Dis moi, est-ce que...".

TODO

Ce type de communication est particulièrement adapté dans des réseaux locaux, afin de transmettre rapidement de grosses quantités d'informations. Du fait de son aspect "sans fioriture", les paquets d'informations sont plus compacts, et engorgent moins les réseaux. Cependant, le transfert par UDP est moins fiable que celui par TCP, et impose aux logiciels clients et serveurs de pouvoir se satisfaire d'une perte d'information, ou tout du moins, d'être capable de redemander une information qui aurait pu être perdue en cours de route.

I-6-3 ICMP

Ce dernier type de trame est assez complexe, et en général uniquement utilisé par des logiciels destinés à analyser la qualité du réseau. Par exemple les commandes ping, traceroute (Unix/Linux), tracert (DOS®/Windows®), etc...

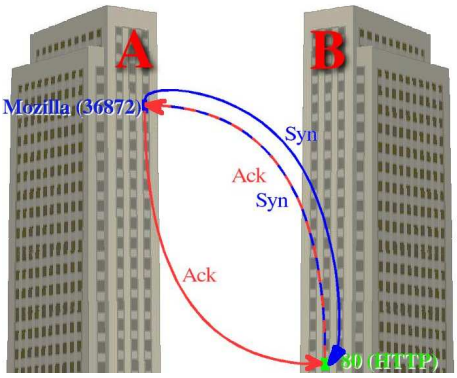
Je sais que je vais faire hurler les puristes d'IP en mélangeant dans un même paragraphe un élément de la couche 2(*) (ICMP) avec d'autres de la couche 3(*) (TCP et UDP), mais en fait, cette approximation n'est pas si primordiale que cela dans le cadre de cette documentation... 😀
(*): Dans le modèle TCP/IP, et non le modèle OSI. Voir à ce sujet ces pages ici et là.

Il n'est pas spécialement utile de rentrer dans les détails de ce type de trame, donc passons à la suite.

I-7 La poignée de mains (handshake)

Lorsque deux personnes se rencontrent dans la rue, la première chose qu'elles font est de se serrer la main, de se faire une bise, de se dire bonjour ou de se présenter. C'est un moyen de commencer une discussion (ie : initialiser une communication) qui permet de mettre en confiance les deux partenaires. Pour le type de trames TCP, c'est la même chose... Et oui, les ordinateurs sont polis (même si certains plantent tout le temps !)...

Supposons que la machine A veuille demander une information à la machine B :



- A va tout d'abord envoyer un paquet de données à B lui demandant l'autorisation de lui parler. Ce paquet s'appelle un SYN (pour "SYNchronisation").
- B, en machine polie, va renvoyer un paquet à A pour lui dire qu'elle est d'accord pour entamer le dialogue. Ce paquet envoyé en retour devrait s'appeler un ACK (ACKnowledge, pour "reconnaissance"). Mais comme c'est la première fois que B envoie un paquet à A, elle doit lui aussi synchroniser cette connexion. Le paquet est donc un SYN / ACK.
- A est donc contente, elle sait que B est bien présente sur le réseau, et en plus disposée à lui parler. Elle peut donc lui envoyer sa requête. Mais comme le paquet que lui a envoyée B contient un "SYN", c'est que cette dernière attend elle aussi une confirmation que A a bien reçue sa réponse. Donc A va envoyer un second paquet à B, pour l'informer. C'est donc un ACK.
- Et maintenant, B va envoyer à A...
  Naaannnnn, c'est une plaisanterie, c'est juste histoire de vérifier que vous suiviez toujours 😀. En fait, la poignée de mains est maintenant terminée . A et B sont prêtes à s'échanger des informations. Donc A va envoyer sa requête à B, qui décidera quoi répondre à A.

Tout ce mécanisme de poignée de mains peut paraître un peu lourd, mais en fait cela a le mérite de synchroniser les deux machines afin qu'elles puissent entamer leur discussion en parfaite harmonie. On verra un peu plus loin en quoi ce mécanisme de "handshake" est important, et comment il permet de sécuriser un peu plus son firewall grâce à la technique du suivi de connexion.

I-8 Interfaces réseaux

Il n'est peut-être pas très logique, pour une explication sur les réseaux, de terminer par les interfaces. Mais du fait de l'utilisation de mes analogies, c'est ce qui m'a semblé le plus efficace à faire... Les experts réseaux (y en a t' il encore à ce niveau du chapitre ?) voudront donc bien m'excuser d'avoir ainsi un peu bousculé les habitudes... 😀

Pour cette partie, nous ne parlerons donc pas d'immeubles et de personnes voulant s'échanger des informations par courrier. Un ordinateur, pour communiquer physiquement avec d'autres ordinateurs, a besoin d'au moins 2 choses : un adapteur réseau et un câble.

Le câble en lui même peut être une fibre optique, une paire de fils de cuivre, un câble RJ-45, ou un câble coaxial. Bon, c'est juste un morceau de connecteur (les vendeurs de câbles, ne pas taper SVP...), donc on ne va pas en parler plus que cela...

L'adapteur réseau lui est bien plus intéressant. C'est en fait un élément électronique qui peut être soit :

De multiples interfaces réseaux

- Une carte réseau, enfichée dans le boîtier de l'ordinateur.
- Un modem interne ou externe, de type RTC (Réseau Téléphonique Commuté) ou Numéris, dialoguant avec la machine par une interface série (COM 1, 2, 3, 4 par exemple).
- Un modem ADSL interne ou externe, dialoguant réciproquement avec la machine via le bus PCI ou une interface USB.
- Une interface virtuelle, donc pas du tout électronique (voir le paragraphe suivant)
- Ou autre chose...

Sous Linux, car c'est ce qui nous intéresse ici, on peut dialoguer avec ces adapteurs réseaux via :

- Pour une carte réseau, l'interface ethX, ou X est un nombre. Par exemple, eth0, eth1, eth2,... En général, une machine possède une seule carte réseau, donc on ne trouvera que eth0.
- Pour un modem, quel que soit le type, l'interface pppX, ou X est un nombre. Par exemple, ppp0, ppp1, ppp2,... Là encore, en général une machine possède qu'un seul modem, donc on ne trouve que ppp0.
- Enfin, même si une machine utilise le protocole IP mais qu'elle n'a pas de carte réseau, elle possède une interface réseau particulière appelée loopback (littéralement "boucle de retour"). On en a déjà parlé précédemment , à propos du réseau 127.0.0.0/8. Et oui, pour qu'un tel réseau existe, il faut qu'il s'appuie sur une interface réseau, et c'est le rôle de l'interface loopback, appelée tout simplement lo. Mis à part une utilité qui peut paraître parfaitement cosmétique, cette interface réseau est primordiale, et vous l'utilisez sans doute parfois sans vous en rendre compte. Si par exemple vous avez une machine dont l'adresse IP est 192.168.0.1 sur l'interface réseau eth0, et que depuis cette machine là vous faite un "ping 192.168.0.1", ce n'est pas votre interface réseau physique "eth0" que vous allez utiliser, mais bien l'interface "lo". En effet, le système utilisera "lo", mais vous laissera à penser que vous avez effectivement utilisé "eth0". Nous verrons plus tard l'importance de cette "auto connexion".

La commande /sbin/ifconfig, lancée éventuellement en temps que root, permet d'afficher la liste des interfaces réseaux.

Le plus important dans tout ceci, c'est que Linux voit de la même manière vos adapteurs réseaux, que ce soient des "eth?" ou des "ppp?" (le "?" signifie qu'il faut remplacer cette lettre par un nombre).

Ainsi, dans la suite du document je vais utiliser une machines ayant 2 cartes réseaux eth0 et eth1, où eth1 simulera une connexion vers Internet. Mais chez vous, vous pourrez reprendre les exemples de configuration en remplaçant eth1 par ppp0. De ce fait vous pourrez utiliser (presque) directement les exemples de configuration pour votre propre connexion Internet via modem. A moins bien sûr que vous n'utilisiez un routeur ADSL...

Ce qu'il y a d'intéressant avec Linux, c'est que même si on ne possède qu'une seule et unique carte réseau, le système peut simuler la présence d'autres cartes réseaux, mais en utilisant au final toujours la même carte physique pour dialoguer. C'est très pratique si on veut par exemple avoir 2 adresses IP sur une machine, afin de dialoguer avec 2 réseaux IP différents, mais connectés physiquement entre eux (*). Ainsi on peut avoir quelque chose comme :

Adapteur réseau physique	Interface Linux	Type d'interface	Adresse IP	Masque IP	Réseau IP
(non physique)	lo	Virtuelle	127.0.0.1	255.0.0.0	127.0.0.0 / 8
eth0	eth0	Réelle	192.168.0.1	255.255.255.0	192.168.0.0 / 24
eth1	eth1 (*)	Réelle	10.0.0.1	255.255.0.0	10.0.0.1 / 16
	eth1:0 (*)	Virtuelle	10.0.1.1	255.255.0.0	10.0.1.1 / 16
	eth1:1 (*)	Virtuelle	10.1.1.1	255.255.0.0	10.1.1.1 / 16
	eth1:2 (*)	Virtuelle	10.2.1.1	255.255.0.0	10.2.1.1 / 16
eth2	eth2 (*)	Réelle	172.16.0.1	255.255.255.0	172.16.0.1 / 24
	eth2:0 (*)	Virtuelle	172.16.1.1	255.255.255.0	172.16.1.1 / 24
	eth2:1 (*)	Virtuelle	172.16.2.1	255.255.255.0	172.16.2.1 / 24

(*) Avis aux amateurs : Cette solution est particulièrement mauvaise en terme de sécurité réseau, car elle donne une vrai - fausse impression de sécurité réseau.

Enfin, on peut rencontrer parfois, plusieurs interfaces "eth?" ou "ppp?" qui partagent la même adresse IP. Hein ? Qu'est-ce qu'il dit ? Mais je vous rassure c'est assez rare, surtout dans un réseau local. Car le but d'une telle configuration est d'augmenter la quantité d'informations passant entre 2 machines (on parle alors de multiplier la bande passante du réseau).

I-9 Résumé

Pour résumer notre analogie entre les réseaux IP et une sorte de vie réelle, voici un tableau comparatif. Si vous ne devez vous souvenir que d'une seul chose dans tout ce chapitre, c'est bien ceci...

Concept réseau	Analogie	Exemples
Interface réseau	Aucune analogie	eth0, eth1, ppp0,...
Adresse IP	Immeuble	192.168.0.1, 56.123.8.47,...
Port	Fenêtre	80(HTTP), 21(FTP), 25(SMTP), 110(POP3), ...
Masque de sous-réseau	Quartier	255.255.255.0, 255.0.0.0,...
Réseau IP	Barrière délimitant un quartier	192.168.0.1/255.255.255.0, 56.0.0.0/255.0.0.0,...
Passerelle	Garde barrière	192.168.0.255, 10.255.255.255,...
Type de trame	Type de courrier	TCP, UDP, ICMP,...
Poignée de mains	Établissement d'un dialogue	(SYN), (ACK, SYNC), (ACK)

Bien, maintenant que nous avons vu ces quelques rappels réseaux, rejoignons les experts d'IP, et rentrons dans le vif du sujet !

II SÉCURITÉ DE BASE

PLAN :

Lorsque l'on met une machine en réseau, c'est vraisemblablement pour 2 choses : accéder à de l'information, ou partager de l'information. Sinon, on utilise des supports amovibles pour transférer de l'information du ou vers le monde extérieur : disquette, CDROM, bande, clef USB,...

Lorsque cette mise en réseau se fait dans un réseau local, dont on a toute confiance envers les autres machines ou utilisateurs de ce réseau, et que cette impression est réciproque, les risques encourus sont très faibles. Par contre, du moment que l'on ne contrôle pas tout ce qu'il peut se passer sur le réseau, on peut commencer à se poser des questions sur la sécurité des données contenues dans les machines. Le fin du fin est évidement lorsque l'on connecte la machine à un réseau mondiale comme Internet, où tout utilisateur a la capacité potentielle de venir farfouiller dans vos données.

Cette introduction, bien qu'un peu sombre, n'a pas pour but de pousser le lecteur vers des solutions comme TCPA, Palladium, et le "trusted computing", car de toute façon, ce sont des projets qui ne sont pas conçu pour servir les intérêts de l'utilisateur. Et bien entendu, j'y suis complètement opposé .

Nous verrons dans ce chapitre quels sont les points sur lesquels vous, en temps qu'utilisateur, devez faire attention. Et nous verrons que même pour un système d'exploitation fraîchement installé, en générale beaucoup de choses sont à contrôler.

II-1 Présentation du réseau

Dans toute la suite du document, nous allons nous intéresser à un réseau composé de 3 machines, qui a pour but de simuler le réseau que vous avez chez vous.

- En premier lieu, on trouve la machine Phoenix qui représente soit votre propre machine, soit la machine qui servira de passerelle / firewall pour votre réseau interne. Phoenix est constituée de 2 cartes réseaux, eth0 et eth1, chacune possédant une paire d'adresse IP / nom différente :

Interface réseau	Adresse IP	Masque de sous-réseau	Réseau	Nom
eth0	192.168.0.1	255.255.255.0	192.168.0.0/24	phoenix0.sky.net ou phoenix.sky.net(*)
eth1	10.0.0.1	255.0.0.0	10.0.0.0/8	phoenix1.internet.net

(*) : phoenix.sky.net est un alias sur phoenix0.sky.net, ce qui veut dire que ces 2 noms pointent sur la même adresse IP. C'est un peu bizarre, mais c'est à usage uniquement cosmétique. Ne vous prenez pas trop la tête avec cela...

Le domaine "sky.net" est bien entendu complètement fictif, ce qui n'a pas d'importance car c'est un réseau privé de classe C. Le réseau "internet.net" est lui aussi fictif, mais il a pour but de simuler la connexion de Phoenix au réseau Internet. J'ai utilisé un réseau privé de classe A pour simuler ce pseudo Internet, mais bien évidement dans le cadre de votre connexion, il s'agirait du réseau public Internet.

Dans le cadre de votre connexion personnelle à Internet, vraisemblablement via un modem ADSL, Numéris, ou RTC, vous pouvez utiliser le tableau d'équivalence suivant :

Interface réseau	Adresse IP	Masque de sous-réseau	Réseau	Nom
eth1	10.0.0.1	255.0.0.0	10.0.0.0/8	phoenix1.internet.net
ppp0	81.53.30.153(*)	255.255.255.255	Point à point	grenoble-153.30.53.81.abo.free.fr(*)

(*) : Ces valeurs sont fictives, et représentent une connexion point à point à Internet via Free. Comment, je n'ai pas encore parlé de connexion "point à point" ? Aucune importance, cela se passe au-delà de l'interface modem "ppp0", donc cela n'a pas d'importance pour ce document .

Donc dans le reste de ce document, considérez que l'interface eth1 de Phoenix est votre propre interface réseau ppp0. Ouf, voila une machine qui ne possède qu'une seule carte réseau, cela ressemble sans doute un peu plus à votre propre configuration !

Phoenix va avoir 3 tâches :

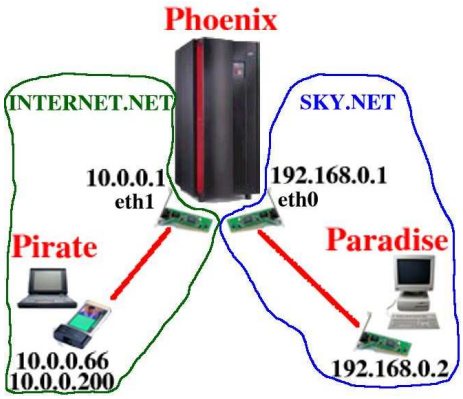
- Fournir des services aux machines du réseau local, comme du partage de fichiers, un service DHCP, de la résolution de noms, une base de données, etc... Il n'est pas très habituel de trouver autant de fonctionnalités de serveur sur une machine personnelle, mais comme sous Linux il est très facile d'en installer (sans pour autant les configurer correctement), j'ai donc grassement chargé cette machine de fonctionnalités... Bien entendu, ces type des services ne sont pas destinés à être exploités par les machines se trouvant sur Internet (donc sur le réseau internet.net), et tout spécialement par pirate.internet.net.
- Servir de passerelle entre le réseau local (sky.net) et Internet (internet.net), afin que les machines du réseau local puissent surfer sur le net. Par exemple, pour que paradise.sky.net aille sur web.internet.net.
- Faire du suivi de port ("port fowarding" en anglais), pour que certains serveurs fonctionnant sur paradise.sky.net puissent être visibles sur internet.net.
- Assurer la sécurité de toutes les machines du réseau sky.net, Phoenix compris.

- Sur le réseau internet.net se trouve la machine Pirate, qui va jouer le rôle d'un vilain fouineur cherchant un moyen d'accéder à votre machine. Pour les raisons de la conférence qui est associée à cette documentation, cette machine jouera en plus de son rôle d'intrus, celui d'un site web tout ce qu'il y a de plus honnête. Cette machine a donc 2 adresses IP, mais si cela vous pose problème, considérez simplement que ce sont 2 machines indépendantes :

Adresse IP	Masque de sous-réseau	Réseau	Nom
10.0.0.66	255.0.0.0	10.0.0.0/8	pirate.internet.net
10.0.0.200	255.0.0.0	10.0.0.0/8	web.internet.net

- Enfin, la dernière machine est Paradise, qui se trouve sur le réseau sky.net. Cette machine aura 2 besoins :
  - Accéder à divers informations et services fournis par Phoenix : partage de fichiers, mails, etc...
  - Se connecter à Internet, et notamment à web.internet.net, pour surfer ou pour faire tout ce que l'on peut faire sur Internet.

Adresse IP	Masque de sous-réseau	Réseau	Nom
192.168.0.5	255.255.255.0	192.168.0.0/24	paradise.sky.net



Configuration du réseau

Pour la suite du document, on notera que les machines fonctionnent sur différentes versions de Linux. Et de ce fait, l'emplacement des fichiers de configuration, des programmes, des fichiers de log, etc... peuvent être différents de votre propre machine. Pas de troll s'il vous plaît sur le choix des distributions Linux !!

Machine	Distribution Linux	Version
Phoenix	Mandrake	9.1
Pirate	Mandrake	9.1
Paradise	Knoppix (Debian)	3.2

Le décors étant planté, passons aux choses sérieuses : Les outils de l'intrus et du défenseur.

II-2 Outils d'analyse et de détection

En fait, quel que soit le rôle que l'on joue, attaquant ou défenseur, on utilise des outils similaires. Car toute personne qui voudra protéger son réseau aura à coeur de le tester lui-même !! Et tout intrus voudra d'abord se protéger lui-même, soit des gens comme lui, soit du défenseur...

II-2-1 Nmap

C'est l'outil par excellence de l'intrus. Il rentre dans la catégorie des "scanner de ports" ("port scanner" en anglais), c'est à dire qu'il va tester un ensemble de ports sur la machine cible, et déterminer lesquels sont ouverts ou fermés. Son exécution est très rapide, et de plus il peut tester automatiquement tout un intervalle d'adresse IP. Bref, si vous avez un logiciel serveur tournant sur votre machine, cet outil le trouvera.

Nmap a une seconde fonctionnalité intéressante, c'est l'identification par empreinte. C'est à dire qu'avec les quelques paquets d'informations que votre machine pourra envoyer à l'intrus, comme par exemple pour dire "ce port est fermé", nmap pourra déterminer quel est le système d'exploitation qui tourne sur votre machine. Quelle importance cela peut il avoir me direz vous ? Très simple : chaque système d'exploitation, voire chaque version de

systèmes d'exploitation, possède des vulnérabilités connues ou inconnues. Ainsi, en sachant quel système tourne sur votre machine, l'intrus pourra lancer des attaques spécifiques, afin de prendre rapidement la main dessus. Donc, moins on donne d'information à l'intrus, mieux on se porte.

Nmap peut être lancé ou non en temps qu'utilisateur classique. Par défaut, nmap ne teste qu'un certain nombre de ports (ceux sur lesquels il pense qu'il va trouver des choses intéressantes), et uniquement en TCP.

Voyons par exemple ce que donne la commande nmap lancée depuis pirate.internet.net sur phoenix1.internet.net :

```
[intrus@pirate /]$ nmap phoenix1.internet.net

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on phoenix1.internet.net (10.0.0.1):
(The 1590 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
110/tcp    open        pop-3
111/tcp    open        sunrpc
139/tcp    open        netbios-ssn
443/tcp    open        https
3306/tcp   open        mysql
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Hummm ça, c'est que du bonheur pour notre pirate ! Autant de ports ouverts sur une machine, il va y avoir des choses intéressantes à faire... On notera au passage que nmap est très sympathique car il nous indique quel est probablement le type de serveur (exemple : http) qui se trouve derrière chaque port (exemple : 80/tcp).

Note importante à propos d'UDP : Par défaut, "nmap" ne teste que les ports TCP. Vous pouvez lui faire tester des ports UDP, mais il faut que vous lanciez "nmap" en temps que root, et en plus avec l'option "-sU". Enfin, si la machine cible est un Linux, les temps de réponses seront très long, car la cible attendra un temps de plus en plus long pour toute connexions UDP faites sur des ports fermés. Ce n'est pas un bug, c'est une mesure de protection de Linux contre le port scanner... Donc à moins que nous n'ayez le temps, ne faite pas un nmap sur tous les ports UDP d'une machine cible, mais seulement sur des ports susceptibles d'être intéressants. Par exemple : "nmap phoenix1.internet.net -sU -p 137" pour tester le port UDP de Samba / NetBIOS.

II-2-2 Tcpdump

Tcpdump est un "sniffeur de trames réseau", c'est à dire qu'il est capable d'afficher à l'utilisateur toutes les trames réseaux qui "passent devant" une carte réseau. Et entre autre, les trames réseaux qui ne sont pas destinées à cette machine. Ce n'est pas très sympathique ce genre de choses, car cela permet par exemple de récupérer un mot de passe réseau, ou une page HTML contenant des informations sensibles (numéro de carte de crédit par exemple).

Ceci est une capture par tcpdump lors de l'envoi d'une requête ping de pirate.internet.net sur phoenix1.internet.net ("ping -c 1 phoenix1.internet.net") :

```
[root@phoenix /]# tcpdump -i eth1
tcpdump: listening on eth1
19:31:25.170017 arp who-has phoenix1.internet.net tell pirate.0.0.10.in-addr.arpa
19:31:25.170079 arp reply phoenix1.internet.net is-at 0:4:75:df:d8:bd
19:31:25.170232 pirate.0.0.10.in-addr.arpa > phoenix1.internet.net: icmp: echo request (DF)
19:31:25.170355 phoenix1.internet.net > pirate.0.0.10.in-addr.arpa: icmp: echo reply
```

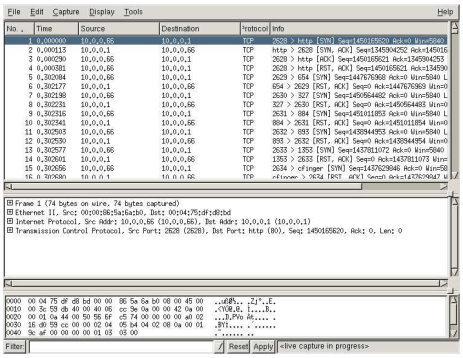
Comme ce type de commande est excessivement sensible pour la sécurité d'un réseau, en général elle ne peut se lancer qu'en temps que root.

Enfin, il est à noter que le lancement de tcpdump demande au système de passer la carte réseau en mode "promiscuous", ce qui a pour effet de laisser une trace dans le log de Linux (le fichier "/var/log/messages"). Cela indique donc à l'administrateur de la machine que des paquets de données ont pu être récupérés illégalement...

```
[root@phoenix /]# grep promiscuous /var/log/messages
Jun 28 19:31:21 phoenix kernel: device eth1 entered promiscuous mode
Jun 28 19:31:28 phoenix kernel: device eth1 left promiscuous mode
```

II-2-3 Ethereal

Contrairement à "tcpdump", "ethereal" est un outil en mode graphique. Mais il fait exactement le même travail que tcpdump. Dans l'exemple ci-contre, on voit la capture des trames lors d'un "nmap phoenix1.internet.net" lancé sur pirate.internet.net.



Ethereal : Capture d'un nmap

Au passage cette capture nous montre des choses très intéressantes :

- On voit bien les demandes de connexions ("SYN") de pirate.internet.net (10.0.0.66), suivi des "ACK/SYN" de phoenix1.internet.net (10.0.0.1), puis le "ACK" en réponse de pirate.internet.net. Bref, c'est toute la poignée de main ("handshake" en anglais) qui se déroule ici.
- On remarque que l'ordre des ports qui sont scannés est croissant, ce qui n'est pas très discret. Mais en fait on peut demander à nmap de faire ses requêtes dans un ordre aléatoire, afin d'être moins visible.
- De même, on voit que les ports sont testés très rapidement (à peine 1/10000ème de seconde pour chaque port), certes sur un réseau rapide (100 Mbit/s). La machine cible est littéralement assaillie. Mais là encore, nmap peut se révéler être plus discret, en espaçant la durée entre chaque connexion.

II-2-4 Netstat

"Netstat" est un outil que l'on va utiliser du coté du défenseur. Il indique l'état des connexions réseaux en cours. Expliquer toutes les options de "netstat" serait trop long, aussi ne va t'on s'intéresser qu'à quelques unes :

- -t : Indique les connexions TCP.
- -u : Indique les connexions UDP.

- -a : Affiche toutes ("all") les connexions, y comprit celle des serveurs en attente de connexion.
- -e : Affiche le nom de l'utilisateur qui a lancé le programme qui utilise ce port.
- -p : Seul le root peut utiliser cette option. Elle indique quel est le PID (Process Identifiant) et le nom du programme utilisant le port.

Ainsi, la commande "netstate -taupe" (combinaison mnémotechnique des options "-t", "-u", "-a", "-e", et "-p") permet d'un seul coup d'oeil de visualiser les programmes en mémoire qui utilisent le réseau IP :

```
[root@phoenix /]# netstat -taupe | sort
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante Etat   Utilisatr  Inode      PID/Program name
tcp        0      0 *:ftp                   *:*              LISTEN root       830048     18450/xinetd
tcp        0      0 *:http                  *:*              LISTEN root       829583     17979/httpd2
tcp        0      0 *:https                 *:*              LISTEN root       829580     17979/httpd2
tcp        0      0 localhost.sky.ne:domain *:*              LISTEN named      835771     18847/
tcp        0      0 localhost.sky.net:rndc  *:*              LISTEN named      835779     18847/
tcp        0      0 localhost.sky.:webcache *:*              LISTEN privoxy    587136     6407/
tcp        0      0 *:mysql                 *:*              LISTEN root       839103     19199/
tcp        0      0 phoenix1.in:netbios-ssn *:*              LISTEN root       839012     19128/smbd
tcp        0      0 phoenix1.interne:domain *:*              LISTEN named      835775     18847/
tcp        0      0 phoenix.sky:netbios-ssn *:*              LISTEN root       839013     19128/smbd
tcp        0      0 phoenix.sky.net:domain  *:*              LISTEN named      835773     18847/
tcp        0      0 *:pop3                  *:*              LISTEN root       830047     18450/xinetd
tcp        0      0 *:smtp                  *:*              LISTEN root       827316     17081/
tcp        0      0 *:sunrpc                *:*              LISTEN root       839095     19208/
tcp        0      0 *:telnet                *:*              LISTEN root       830049     18450/xinetd
tcp        0      0 *:x11                   *:*              LISTEN root       3742       1593/X
udp        0      0 *:bootps                *:*                     root       839078     19191/dhcpd
udp        0      0 localhost.sky.ne:domain *:*                     named      835770     18847/
udp        0      0 *:netbios-dgm           *:*                     root       839024     19138/nmbd
udp        0      0 *:netbios-ns            *:*                     root       839023     19138/nmbd
udp        0      0 phoenix1.in:netbios-dgm *:*                     root       839030     19138/nmbd
udp        0      0 phoenix1.interne:domain *:*                     named      835774     18847/
udp        0      0 phoenix1.int:netbios-ns *:*                     root       839029     19138/nmbd
udp        0      0 phoenix.sky:netbios-dgm *:*                     root       839033     19138/nmbd
udp        0      0 phoenix.sky.:netbios-ns *:*                     root       839031     19138/nmbd
udp        0      0 phoenix.sky.net:domain  *:*                     named      835772     18847/
udp        0      0 *:sunrpc                *:*                     root       839094     19208/
```

II-2-5 Lsof

Pour avoir un résultat utilisable, "Lsof" (LiSt Open File) est un outil qui doit se lancer avec les droits root. Il indique quels sont les fichiers ouverts sur le système. Des fichiers ? Mais de quoi parle t'il ? On est ici pour parler de réseau, non ?  Mais oui, on peut s'intéresser bien sûr aux fichiers classiques tel qu'on les connaît dans d'autres OS (comme un fichier texte, une image, une librairie, un programme, etc...) mais on peut aussi utiliser cette commande pour voir l'activité réseau. Comment ? Simple, sous Linux, tout ce que manipule l'OS est vu comme un fichier, les connexions IP entre autre. Dans notre cas, les paramètres intéressants de "lsof" sont :

- -i : Restreint la recherche aux seuls "fichiers" de connexion IP.
- -P : Converti le numéro de port (exemple : 80) en son nom équivalent (exemple : http).

Ainsi, la commande "lsof -Pi" (combinaison des options "-i" et "-P") nous donne :

```
[root@phoenix /]# lsof -Pi | sort
COMMAND   PID    USER   FD   TYPE DEVICE SIZE NODE NAME
dhcpd   19191    root    6u  IPv4 839078       UDP *:67
httpd2  17979    root    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17979    root    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  17987  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17987  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  17988  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17988  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  17989  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17989  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  17990  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17990  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  17991  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  17991  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
httpd2  18591  apache    3u  IPv4 829580       TCP *:443 (LISTEN)
httpd2  18591  apache    4u  IPv4 829583       TCP *:80 (LISTEN)
master  17081    root   11u  IPv4 827316       TCP *:25 (LISTEN)
mysqld  19199   mysql    3u  IPv4 839103       TCP *:3306 (LISTEN)
mysqld  19215   mysql    3u  IPv4 839103       TCP *:3306 (LISTEN)
mysqld  19216   mysql    3u  IPv4 839103       TCP *:3306 (LISTEN)
named   18847   named   10u  IPv4 835771       TCP localhost.sky.net:53 (LISTEN)
named   18847   named   11u  IPv4 835772       UDP phoenix.sky.net:53
named   18847   named   12u  IPv4 835773       TCP phoenix.sky.net:53 (LISTEN)
named   18847   named   13u  IPv4 835774       UDP phoenix1.internet.net:53
named   18847   named   14u  IPv4 835775       TCP phoenix1.internet.net:53 (LISTEN)
named   18847   named   17u  IPv4 835779       TCP localhost.sky.net:953 (LISTEN)
named   18847   named    8u  IPv4 835778       UDP *:32803
named   18847   named    9u  IPv4 835770       UDP localhost.sky.net:53
named   18848   named   10u  IPv4 835771       TCP localhost.sky.net:53 (LISTEN)
named   18848   named   11u  IPv4 835772       UDP phoenix.sky.net:53
named   18848   named   12u  IPv4 835773       TCP phoenix.sky.net:53 (LISTEN)
named   18848   named   13u  IPv4 835774       UDP phoenix1.internet.net:53
named   18848   named   14u  IPv4 835775       TCP phoenix1.internet.net:53 (LISTEN)
named   18848   named   17u  IPv4 835779       TCP localhost.sky.net:953 (LISTEN)
...
```

Le résultat a été tronqué, car trop long. On remarque que par rapport à la précédente commande , la liste des ports est plus longue. Bug ou erreur ? En fait, non : "lsof" affiche les forks et les threads qui utilisent les fichiers, alors que "netstat" n'affiche que les processus pères.

II-2-6 Fuser

"Fuser" (File user) est un peu comme "lsof", à savoir qu'il se base sur les fichiers pour déterminer les connexions IP ouvertes. Pour l'usage qui nous intéresse, nous ne verrons qu'un seul paramètre :

- -v [type de ports] : Affiche les ports ouverts du type [type de ports]. Exemple : "80/tcp" pour les port TCP 80.

On voit que "fuser" n'est adapté qu'à donner des informations sur un nombre limité de ports, et qu'en aucun cas, il n'indique de lui même les ports ouverts. Par rapport à "lsof", il est donc un peu moins intéressant. A noter que tout comme "lsof", "fuser" nécessite les droits root.

La commande "fuser 80/tcp" nous donne :

```
[root@phoenix /]# fuser -v 80/tcp

                     USER        PID ACCESS COMMAND
80/tcp               root      17979 f....  httpd2
                     root      17987 f....  httpd2
                     root      17988 f....  httpd2
                     root      17989 f....  httpd2
                     root      17990 f....  httpd2
                     root      17991 f....  httpd2
                     root      18591 f....  httpd2
```

Et "fuser 67/tcp 67/udp 80/tcp" nous donne :

```
[root@phoenix /]# fuser 67/tcp 67/udp 80/tcp
67/udp:              19191
80/tcp:              17979 17987 17988 17989 17990 17991 18591
```

Ici, les ports 80 en TCP et 67 en UDP sont ouverts. Par contre, le port 67 TCP n'est pas ouvert.
Pour information, le port 67 est celui du DHCP, et il ne fonctionne en général qu'en UDP, et non TCP.

II-2-7 Ping

Il est sans doute inutile de s'étendre sur la commande "ping", tellement elle est connue. "Ping" permet de vérifier la présence d'une machine en lui envoyant un paquet ICMP, auquel la machine cible doit répondre par un paquet ICMP équivalent. La commande affiche alors des informations intéressantes : Adresse IP, durée de transmission, etc...

Utilisé à de mauvaises fins, "ping" peut :

- Saturer une machine, en la noyant sous un flot de paquets, ce qui a pour effet de consommer la bande passante et d'augmenter la charge CPU. Une utilisation particulière basé sur la technique du "ping" est l'attaque DDOS (Distributed Deny Off Service, ou "attaque distribuée par refus de service"), qui peut ralentir énormément une machine, voir un réseau...
- Donner des informations sur la machine cible. Même si une machine n'exécute aucun serveur (voir le paragraphe suivant), "ping" peut indiquer à l'intrus que sa cible est quand même en fonctionnement, ce qui peut être un premier pas pour exciter sa curiosité... Par défaut, les machines ne sont pas configurées pour ne pas répondre aux ping. Dans le chapitre suivant, nous verrons comment configurer la machine pour qu'elle ne réponde pas à "ping", et aux autres paquets ICMP en général.

Utilisation de la commande "ping" sur www.google.fr :

```
[olivier@phoenix /]$ ping www.google.fr
PING www.google.com (216.239.39.99) 56(84) bytes of data.
64 bytes from 216.239.39.99: icmp_seq=1 ttl=49 time=519 ms
64 bytes from 216.239.39.99: icmp_seq=2 ttl=49 time=229 ms
64 bytes from 216.239.39.99: icmp_seq=3 ttl=49 time=229 ms
64 bytes from 216.239.39.99: icmp_seq=4 ttl=49 time=229 ms
64 bytes from 216.239.39.99: icmp_seq=6 ttl=49 time=229 ms
64 bytes from 216.239.39.99: icmp_seq=7 ttl=49 time=229 ms
64 bytes from 216.239.39.99: icmp_seq=8 ttl=49 time=219 ms
64 bytes from 216.239.39.99: icmp_seq=10 ttl=49 time=219 ms

--- www.google.com ping statistics ---
10 packets transmitted, 8 received, 20% packet loss, time 9349ms
rtt min/avg/max/mdev = 219.981/263.552/519.507/96.831 ms
```

Cette commande nous indique que l'adresse IP de "www.google.fr" est "216.239.39.99" (en fait, il y en a plusieurs, et ceci n'est que l'une des adresses de ce site). Par contre, Google doit filtrer une partie des paquets ICMP, ou la connexion doit être mauvaise, car seul 80% des paquets envoyés par Phoenix ont reçu une réponse... Ce sont les aléas d'IP !

II-2-8 Traceroute

Cette commande se base aussi sur ICMP, mais elle est plus intéressante. Elle indique le chemin utilisé par les paquets IP pour aller de votre machine à une machine cible. En plus de cela, elle affiche le temps mis par les paquets pour passer d'un réseau à un autre.

Cette commande est principalement utilisée par les administrateurs réseaux. Elle sert le plus souvent à vérifier la qualité des connexions, ou à déterminer les engorgements dans les réseaux. Cependant, elle peut être aussi utilisée pour remonter la trace d'une machine, et trouver l'origine de sa source.

Du fait de son caractère orienté sur la sécurité, la commande "traceroute" ne peut s'exécuter qu'en temps que root. Cependant, sur certaines OS ou distributions Linux, ce programme est dit "Set-UID root" ("man chmod" pour plus d'information). C'est à dire que lorsque qu'il est lancé, Linux donne au programme les droits de son propriétaire. Concrètement, lorsque vous lancer un programme qui a de telles capacités, vous devenez temporairement root à la place du root ! (is no good ? 😃) L'emplacement de ce type de programme est en général en dehors des emplacements classiques de stockages des programmes utilisateurs. Sous Linux, vous le trouverez généralement en "/usr/sbin/traceroute" :

```
[root@phoenix /]# ls -la /usr/sbin/traceroute
-rwsr-xr-x    1 root     bin         18136 mai  7  2002 /usr/sbin/traceroute*
```

Ici, c'est le caractère "**s**" qui indique que le programme est "Set-UID". Et on voit que le programme appartient au **root**. Donc ce programme est bien "**Set-UID root**".

Exemple d'utilisation de "traceroute" sur la machine "200.165.134.194" qui a tenté aujourd'hui d'accéder illégalement à ma machine :

```
[root@phoenix /]# traceroute 200.165.134.194
traceroute to 200.165.134.194 (200.165.134.194), 30 hops max, 38 byte packets
 1  192.168.254.254 (192.168.254.254)  119.244 ms  110.262 ms  109.510 ms
 2  grenoble-3k-1-a5.routers.proxad.net (213.228.10.30)  109.974 ms  109.811 ms  120.138 ms
 3  cbv-6k-1-a7.routers.proxad.net (213.228.3.120)  129.695 ms  119.787 ms  130.286 ms
 4  prs-b2-geth6-0.telia.net (213.248.71.13)  129.895 ms  129.943 ms  119.906 ms
 5  prs-bb1-pos1-2-0.telia.net (213.248.70.5)  129.947 ms  120.188 ms  119.663 ms
 6  ldn-bb1-pos0-2-0.telia.net (213.248.64.157)  130.058 ms  119.730 ms  129.883 ms
 7  ldn-bb2-pos0-0-0.telia.net (213.248.64.162)  149.989 ms  150.124 ms  139.694 ms
 8  nyk-bb2-pos2-3-0.telia.net (213.248.65.38)  220.042 ms  219.873 ms  219.907 ms
 9  nyk-bb1-pos0-0-0.telia.net (213.248.80.1)  189.920 ms  190.519 ms  199.344 ms
10  sl-gw27-nyc-10-0.sprintlink.net (144.232.230.29)  190.289 ms  190.159 ms  189.732 ms
```

```
11  sl-bb24-nyc-15-0.sprintlink.net (144.232.7.25)  210.080 ms  199.748 ms  199.883 ms
12  sl-bb21-nyc-6-0.sprintlink.net (144.232.13.186)  189.992 ms  209.842 ms  199.948 ms
13  sl-bb21-atl-11-1.sprintlink.net (144.232.18.69)  220.024 ms  209.990 ms  219.963 ms
14  sl-bb20-orl-14-2.sprintlink.net (144.232.19.170)  239.885 ms  229.955 ms  229.944 ms
15  sl-bb21-orl-15-0.sprintlink.net (144.232.2.146)  230.250 ms  239.882 ms  230.041 ms
16  sl-st21-mia-15-1.sprintlink.net (144.232.20.13)  239.983 ms  229.887 ms  239.914 ms
17  sl-splkt1-3-0.sprintlink.net (144.223.244.78)  349.943 ms  349.903 ms  349.912 ms
18  200.187.128.69 (200.187.128.69)  350.908 ms  349.908 ms  349.933 ms
19  200.223.131.213 (200.223.131.213)  389.949 ms  399.927 ms  390.009 ms
20  PO0-0.BOT-RJ-ROTN-01.telemar.net.br (200.223.131.122)  389.862 ms  399.640 ms  389.948 ms
21  PO6-0.ASGS-BA-ROTN-01.telemar.net.br (200.223.131.73)  390.002 ms  379.775 ms  379.940 ms
22  PO4-0.BVG-PE-ROTN-01.telemar.net.br (200.223.131.17)  390.056 ms  389.770 ms  390.063 ms
23  PO10-0.BVG-PE-ROTD-02.telemar.net.br (200.223.131.22)  399.816 ms  389.723 ms  389.966 ms
24  200.164.204.198 (200.164.204.198)  399.975 ms  389.775 ms  390.692 ms
25  200.164.234.34 (200.164.234.34)  1379.447 ms  2529.793 ms  1259.899 ms
26  200.165.134.194 (200.165.134.194)  1809.956 ms  2259.911 ms  2969.929 ms
```

Analysons tout ceci :

- La première ligne est le point de départ de la commande "traceroute", c'est à dire Phoenix. Remarquez qu'il s'agit d'une adresse IP privée, ce qui n'est pas anormal pour une connexion point à point.
- La 2nde ligne montre ma propre adresse IP sur Internet. Comme je n'utilise qu'un modem RTC pour me connecter, cette adresse IP change tout le temps. Donc inutile de vous acharner sur cette adresse afin de pirater ma machine, ce ne sera pas la mienne qui se trouvera à cette adresse IP lorsque vous lirez ces lignes. Au passage, on voit que ma machine se trouve dans la région de Grenoble (en Isère / France / Europe / Terre / Système solaire / Voie lactée), et que mon fournisseur d'accès est Free (Proxad.net <-> Free), mais cela, vous deviez déjà vous en douter...
- La ligne 5 indique que l'on sort du réseau de Free, pour rentrer sur le réseau de TeliaSonera, qui est un groupe Européen de transmissions réseaux.
- En 6 nous sommes à Londre (indicatif "lnd" <-> Londre).
- En 8 nous sommes à New-York (indicatif "nyk" <-> New-York).
- En 10, nous entrons dans le réseau de SprintLink, un autre grand réseau de communications. Les lignes suivantes nous font descendre la coté Est des États-Unis pour Miami. Afin de déterminer le chemin emprunté, on s'aide des règles d'appellation des routeurs de ce réseau .
- En 20, nous sommes maintenant au Brésil !
- En 23, cela se précise, nous sommes dans la région de Pernambuco, sur la pointe Est du Brésil.
- Enfin, la ligne 25 nous indique que nous avons atteint notre but. Il est difficile de déterminer quelle est la nature de cette machine, et si ou non c'est un "script kiddy" qui tue le temps à faire du port scanning, au lieu d'aller sur la plage (quoi que là bas, c'est l'hiver, et l'eau n'est peut-être qu'à seulement 25 degrés...). Mais comme c'est le port 137 de ma machine (le partage de fichiers) qui l'intéressait, j'ose espérer que ce n'était pas une société, en mal de recherche de secrets industriels !

Pour aller plus en avant dans ma recherche, il faudrait que je contacte l'administrateur de ce réseau, et obtenir plus d'informations. Quoiqu'il en soit, que les responsables de ce réseau qui lisent ce document ne se sentent pas l'esprit de justiciers, et n'aillent pas lui casser la tête !! Je me contenterai de la destruction pure et simple de sa machine, ainsi que de l'immolation de ces CD illégaux .

II-2-9 Autres informations

Évidement, ceci n'est qu'une petite liste des outils Linux de surveillance de l'activité réseau. Il en existe beaucoup d'autres, et encore plus qui seront développés dans l'avenir. La recherche sur Internet ou sur des sites spécialisés dans la sécurité informatique vous en apprendra beaucoup plus que ces quelques lignes. Nous avons majoritairement vu ici des outils en mode ligne de commande, mais il en existe des équivalents en mode graphique.

Pour ce qui est de l'utilisation plus complète de ces programmes, je vous invite à consulter leurs documentations respectives. Les commandes "man netstat", "info nmap", etc... sont vos meilleurs amis !

II-3 Démons, serveurs et démons de service

Lorsque vous démarrez votre Linux (oui je sais, c'est long...), tout un tas de programmes sont lancés. Nous allons voir un peu plus en détail de quoi il s'agit.

II-3-1 Démons

Non, nous n'allons pas parler de la Bible, ou d'une aventure de Donjons & Dragons. Sous Linux, un démon est un programme qui se charge en mémoire, attendre certains événements, et réagir en fonction d'eux. On peut citer en vrac :

- cron, qui lance des tâches à intervalles réguliers.
- dm qui gère la sourie en mode texte.
- xfs, le serveur de fontes de X.
- etc...

Bref, tout ces programmes tournent tranquillement, et effectuent leurs tâches sans que vous ne vous en soucier. Pourquoi en parler ? Parce que souvent on parle invariablement de démons et de serveurs, et qu'il est intéressant de connaître les deux appellations.
Pour les utilisateurs de Windows®, un démon est l'équivalent d'un "service".

II-3-2 Serveurs

Un serveur est un programme résident en mémoire, tout comme un démon, à la différence qu'il est tout spécialement intéressé par une activité réseau particulière. Donc un serveur va interagir avec un logiciel à distance, via par exemple le protocole IP. Que le logiciel qui dialogue avec lui soit sur une machine éloignée (connectée via "ethX" ou "pppX"), ou en local (via "lo"), cela n'a pas d'importance.

Lors de la précédente commande " netstat", les programmes répondants au doux nom de "http2", "smbd", "dhcpd", etc..., étaient tous des serveurs.

II-3-3 Démons de service (inetd / xinetd)

Inetd et son "fils" xinetd (pour "eXtended inetd") sont deux serveurs particuliers. "Inetd" étant l'ancêtre et commençant à disparaître des configurations Linux, je ne parlerai que de "Xinetd", qui fait le même travail que "Inetd", mais en mieux...

Le but d'"xinetd" est de se placer entre la couche IP et un ou plusieurs serveurs, afin de les protéger. On peut par exemple vouloir ne laisser l'accès à un serveur que depuis une certaine adresse IP, ou une certaine interface réseau, ou encore uniquement pendant une certaine fenêtre temporelle, etc... Ou alors par une combinaison de quelques unes de ces contraintes. Auquel cas, il faut que le serveur en question ait été suffisamment bien programmé, et qu'il dispose de toutes ces options de configuration. Mais développer autant de paramétrages sur autant de serveurs différents, cela demande du temps, et est la source potentielle de nombreux bugs qui peuvent compromettre la sécurité de la machine.

C'est là qu'intervient "xinetd". Ce meta serveur va se charger d'attendre les requêtes venant de logiciels clients, puis d'appliquer les restrictions qu'on lui a donné, et enfin de décider ou non si la requête est valide. Si c'est le cas, il lancera le logiciel serveur à qui est destiné la requête et laissera ce dernier de débrouiller avec le client. Parmi l'ensemble de logiciels dont nous allons parler un peu plus loin certains peuvent être protégés par xinetd

(pop3, proftpd, telnet,...) mais d'autres non (samba, apache, dhcpd,...). Pour ces derniers, c'est à eux de fournir tous les réglages de sécurité dont l'utilisateur peut avoir besoin. En général, ce sont de gros logiciels serveurs qui les proposent justement, et qui pour des raisons de performance, ne peuvent pas se permettre de gaspiller du temps CPU à faire analyser leur trafic IP par un meta serveur comme xinetd.

Une documentation très intéressante, et en français, sur la configuration de xinetd se trouve ici.

II-4 Risques

Faire tourner un serveur sur sa machine est certes très intéressant, mais il faut avoir conscience des risques que l'on encourt. Et en général, c'est là que le lecteur commence à pâlir et à se dire "M.... c'est ce que j'ai fait. Ca craint....".

- L'utilisateur : Faire tourner un serveur, c'est bien. Mais ce programme là tourne avec quels droits ? Je m'explique : Sous Linux, nous avons différents utilisateurs (par exemple "olivier"), qui ont leur propre compte, et qui utilisent des logiciels. Si l'utilisateur fait un erreur en utilisant un logiciel (ou que celui-ci fasse une erreur), les conséquences ne pourront avoir lieu qu'avec les droits de l'utilisateur.

  Par exemple, la commande "rm -rf /" va avoir pour conséquence de supprimer tous les fichiers et tous les répertoires accessibles par la machine (Non, je ne ferais pas un exemple de cette commande, même pour vous faire plaisir...). Tout les fichiers vont être détruits ? Non, en fait seul ceux dont l'utilisateur "olivier" a les droits d'écriture y passerons, à commencer par les fichiers de "/home/olivier". Mais les principaux fichiers de la machines (les exécutables, les fichiers de configuration, etc...), et les fichiers des autres utilisateurs seront épargnés. Ouf...

  Oui, mais si la commande "rm -rf /" avait été lancé par l'utilisateur root, que ce serait t'il passé ? Et bien, je n'aurais eu plus qu'à m'arracher les cheveux, et tout réinstaller.

  Ce petit exemple a pour but de vous faire prendre conscience des problèmes de droits d'accès. Imaginez maintenant qu'un programme serveur soit lancé avec les droits root, et que d'une manière ou d'une autre un logiciel client puisse arriver à faire exécuter une commande par le logiciel serveur, de type "rm -rf /"... Vous avez compris, ou il faut que je vous fasse un dessin ? Cet exemple est malheureusement très représentatif de la situation d'Internet actuellement. Et si des milliers de machines se font pirater tout les jours par le dernier virus-vers à la mode ("nimda", "kournikova", etc...), c'est souvent parce que le serveur web qui a été touché était lancé avec les droits les plus élevés.

  Heureusement il y a une solution. C'est de lancer le serveur avec les droits d'un utilisateur particulier, qui n'a pas accès à grand chose. Si on reprend les commandes "netstat" ou "lsof" vues plus haut, on voit par exemple que :
  - Le programme qui tourne sur "localhost.sky.:webcache" a été lancé avec l'utilisateur "privoxy".
  - Les différents programmes de type ":domain" sont lancés avec l'utilisateur "named".

  Évidement, ces programmes n'ont que des droits très limités, donc les risques sont faibles. En cas d'attaque un peu sévère, ces programmes ne pourraient supprimer qu'une poignée de fichiers.

  Certains autres programmes ("httpd2", "ftp", etc...) sont lancés avec des droits root. Horreur ?!?!?! En fait non, car ceux-là peuvent être capable de changer d'utilisateur à la volé, juste avant de commencer à analyser les données venant du client. On le voit bien avec le logiciel "http2", qui :
  - Selon "netstat" tourne sous le compte de root ("root 829583 17979/httpd2").
  - Par contre, "lsof" indique que la connexion IP est ouverte par l'utilisateur "apache" ("httpd2 17987 apache").

  Conclusion : Avant de lancer un serveur, bien faire attention à l'utilisateur qui le lance. Cela se paramètre soit dans les fichiers de configuration du serveur (répertoire "/etc/"), ou carrément dans le script qui lance le démon (répertoire "/etc/init.d/").

- La prison : Même si on a bien fait attention à lancer un serveur avec les droits d'un utilisateur non root, il peut arriver 2 choses :
  - Il existe quelque part sur la machine, des fichiers ou des répertoires qui sont en écriture pour tout le monde (Et cela, C'est Mal). Auquel cas cet utilisateur, même si il n'a que des droits limités, pourra quand même supprimer des fichiers.
  - Mais le but d'un pirate n'est pas que de détruire. Il peut aussi vouloir récupérer des informations sur votre machine. Par exemple, l'état de votre compte saisit dans gnucash, vos mots de passe pour vos comptes email (exemple : "~/.fetcmailrc"), ou pire encore... Êtes vous sûr que tous vos fichiers et vos répertoires sont tous interdit en lecture / exécution ("rwxr-x--") ? Nannn, pas sur du tout, hein ? Bon, alors vous êtes mûre pour le chroot, "the jail", bref la prison.

  L'idée de la technique du "chroot", est de faire exécuter une commande en lui faisant croire que le "/" est à un autre endroit. Pour les utilisateurs de Windows®, cela reviendrait à lancer par exemple le "notepad.exe" depuis le "C:\JAIL\WINDOWS\notepad.exe", en lui faisant croire qu'il est en fait en "C:\WINDOWS\notepad.exe". Mais la technique du "chroot" n'existe pas du tout sous Windows®.

  L'intérêt de cette technique est évidente. Le programme s'exécute dans un vrai - faux "/", dans lequel il n'y a rien d'important. Ainsi, si le programme passe sous l'influence de l'intrus, il ne pourra pas sortir de ce faux "/", qui sera donc sa prison...

  Pour mettre en oeuvre un "chroot", il faut que le programme en question ait été conçu un minimum pour cela (afin d'éviter d'avoir besoin d'une grande quantité de programmes externes, de devices, etc...), et que surtout la technique de mise en place du "chroot" soit bien documentée ! Cela ne se voit pas dans les exemples ci-dessus, mais sur la configuration de Phoenix, le démon "named" (c'est un "DNS", un Serveur de Nom de Domaine, quoi !) est lancé depuis un "chroot". Je passerai les détails de la configuration, mais voici le résultat.

  Les fichiers dont "named" à besoin sont ici :

```
[root@phoenix /]# find /var/named/
/var/named/
/var/named/dev
/var/named/dev/log
/var/named/dev/null
/var/named/dev/random
/var/named/dev/zero
/var/named/etc
/var/named/etc/localtime
/var/named/etc/named.conf
/var/named/etc/rndc.conf
/var/named/etc/rndc.key
/var/named/var
/var/named/var/named
/var/named/var/named/localhost
/var/named/var/named/10.0.0
/var/named/var/named/internet.net
/var/named/var/named/192.168.0
/var/named/var/named/127.0.0
/var/named/var/named/sky.net
/var/named/var/run
/var/named/var/run/named.pid
```

  Mais en fait, ce que voit "named" lorsqu'il est lancé c'est :

```
/
/dev
/dev/log
```

```
/dev/null
/dev/random
/dev/zero
/etc
/etc/localtime
/etc/named.conf
/etc/rndc.conf
/etc/rndc.key
/var
/var/named
/var/named/localhost
/var/named/10.0.0
/var/named/internet.net
/var/named/192.168.0
/var/named/127.0.0
/var/named/sky.net
/var/run
/var/run/named.pid
```

Bilan : La technique du chroot est une option intéressante qui permet de limiter grandement les indiscrétions d'un démon qui "serait passé entre les mains" de l'intrus. Malheureusement, cette technique n'est pas fiable à 100%, et si l'intrus est assez fort, il arrivera quand même à sortir de sa prison... C'est dingue, non ?

- Le User-Mode kernel : Bon, là on s'accroche car c'est du costaud ! C'est une nouvelle (*) fonctionnalité du kernel Linux 2.5.x, qui permet de faire fonctionner un kernel Linux dans un autre kernel Linux. Les performances sont clairement réduites, mais c'est l'équivalent à un "chroot" sur le kernel lui-même. Et à priori, on ne peut pas sortir d'une prison comme celle là. Même si cela doit marcher (je ne sais pas, je n'ai pas testé. Un jour peut-être...), c'est quand même sacrément lourd comme technique. Et à moins d'être complètement paranoïaque, ce n'est pas vraiment utilisable pour un particulier. En fait, cette technique a principalement été développée afin de faciliter l'écriture des pilotes de périphériques pour le kernel Linux.

  (*) : Disponible à la fin 2002 (kernel 2.5.35). Il m'est d'avis que dans quelques mois, cela paraîtra excessivement moins nouveau comme fonctionnalité... 😃

Bon, arrêtons nous là pour les risques encourus. Comme on l'a vu, une fois que l'intrus a pu prendre la main sur le serveur, il peut arriver à faire des dégâts. Donc il faut l'arrêter avant qu'il ne dialogue avec le serveur. Comment ? En interdisant tout simplement au serveur de lui répondre, pardi ! Et c'est ce que l'on va voir tout de suite...

II-5 Fermeture des ports

II-5-1 Un peu de bon sens

La première chose à faire, est de lancer un petit audit de votre machine. Pour cela, utilisez " nmap", "netstat" et ses petits copains pour déterminer quels sont les serveurs qui tournent sur votre machine. Rappelez vous que lancer nmap sur les ports UDP est très long, et qu'il est plus rapide d'abord de rechercher la présence des ports UDP avec "netstat", puis de les tester au cas par cas avec nmap.

Bon, une fois que c'est fait, posez vous la question : "Est-ce que j'ai besoin du serveur xxxxx ?". Parfois, la réponse peut être négative, et auquel cas l'action doit être immédiate : fermer au plus tôt ce serveur !! Il n'y a rien de plus dangereux que de laisser un serveur tourner sans l'utiliser, car on fini par oublier jusqu'à sa présence, et on ne fera pas attention à le mettre à jour pour corriger ses défauts. Auquel cas, un intrus peut tomber sur ce serveur et bénéficier d'un trou de sécurité pour pénétrer dans votre machine.

Dans l'exemple qui va suivre, on va chercher à supprimer le serveur Samba / NetBIOS de la machine Phoenix. Samba est un système le partage de fichiers en réseau avec les machines Windows® :

```
[root@phoenix /]# nmap phoenix0.sky.net -p 138,139 && nmap phoenix0.sky.net -p 137,138 -sU

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on phoenix.sky.net (192.168.0.1):
(The 1 port scanned but not shown below is in state: closed)
Port       State       Service
139/tcp    open        netbios-ssn

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on phoenix.sky.net (192.168.0.1):
Port       State       Service
137/udp    open        netbios-ns
138/udp    open        netbios-dgm

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

D'après les commandes nmap, trois ports (139/tcp, 137/udp et 138/udp) sont utilisés pour le service "netbios". Oui, ce serveur est un peu particulier, d'où l'utilisation dans cet exemple... En fait le protocole "NetBIOS" peut utiliser les ports/protocoles suivant: 137/tcp/udp, 138/tcp/udp, et 139/tcp/udp). Afin de supprimer ce serveur, il nous faut d'abord savoir quels sont les programmes qui tournent derrière ces ports :

```
[root@phoenix /]# netstat -taupe | sort
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante Etat   Utilisatr  Inode      PID/Program name
tcp        0      0 phoenix1.in:netbios-ssn *:*              LISTEN root       4583       2157/smbd
tcp        0      0 phoenix.sky:netbios-ssn *:*              LISTEN root       4584       2157/smbd
udp        0      0 phoenix1.in:netbios-dgm *:*                     root       4599       2167/nmbd
udp        0      0 phoenix1.int:netbios-ns *:*                     root       4598       2167/nmbd
udp        0      0 phoenix.sky:netbios-dgm *:*                     root       4601       2167/nmbd
udp        0      0 phoenix.sky.:netbios-ns *:*                     root       4600       2167/nmbd
```

Seul les lignes contenant "netbios" sont affichées. On voit que les programmes qui tournent sont "smbd" et "nmbd". Bien, voila qui est intéressant.

Maintenant, nous allons chercher le programme qui a lancé ces serveurs. Sous Linux, tous les programmes (en fait, ce sont des scripts shell) qui lancent des démons sont situés dans "/etc/init.d/" :

```
[root@phoenix /]# ls -la /etc/init.d/
-rwxr--r--    1 root     root         1838 mar 14 17:11 smb*
```

Seul le script qui nous intéresse est affiché. On voit donc que l'on a un script appelé "/etc/init.d/smb" qui pourrait bien correspondre à ce que nous

voulons. Lançons le, histoire de voir :

```
[root@phoenix /]# /etc/init.d/smb
Utilisation : /etc/init.d/smb {start|stop|restart|status|condrestart}
```

Bon, le programme demande un paramètre. Les scripts de démons doivent en effet être lancés en leur indiquant l'opération à effectuer. Ainsi, on trouve généralement les options suivantes :

- start : Démarre le démon.
- stop : Arrête le démon.
- restart : Redémarre le démon. En fait, c'est exactement la même chose que de faire un "stop" puis un "start".
- reload : Relit le(s) fichier(s) de configuration du démon, afin de prendre en compte un changement de configuration. Cette option ne marche pas toujours très bien, et parfois, il vaut mieux faire un "restart" plutôt qu'un "reload". Mais cela coupe temporairement le serveur, ce qui peut poser problèmes aux clients qui y sont déjà connectés.
- condrestart : Redémarre le démon si celui-ci est déjà démarré. Ne fait rien sinon.
- status : Affiche le status du démon.

Que se passe t' il si l'on utilise "status" pour notre démon Samba ?

```
[root@phoenix /]# /etc/init.d/smb status
smbd (pid 2157) est en cours d'exécution...
nmbd (pid 2167) est en cours d'exécution...
```

Bingo ! Se script contrôle le lancement de "smbd" et "nmdb" et les PID sont les mêmes que ceux de la commande "netstat" précédente. C'est donc bien ce script qui nous intéresse. Arrêtons donc le serveur Samba :

```
[root@phoenix /]# /etc/init.d/smb stop
Arrêt des services SaMBa :                                 [  OK  ]
Arrêt du service NMB :                                     [  OK  ]
[root@phoenix /]# /etc/init.d/smb status
smbd est arrêté
nmbd est arrêté
```

Samba semble arrêté, comme le confirme la 2nd commande. Il nous reste plus qu'à le vérifier :

```
[root@phoenix /]# nmap phoenix0.sky.net -p 139 && nmap phoenix0.sky.net -p 137 -sU

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
The 1 scanned port on phoenix.sky.net (192.168.0.1) is: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
The 1 scanned port on phoenix.sky.net (192.168.0.1) is: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

OK, "nmap" ne trouve plus rien. Et "netstat" ?

```
[root@phoenix /]# netstat -taupe | grep netbios | sort
```

Et "netstat" non plus. Victoire, nous venons de supprimer un serveur et de fermer des ports !!! Bien, plus de risque à craindre pour ce serveur. Si un intrus fait un "nmap" sur ces ports, il ne trouvera rien. Et il pourra bien tenter tout ce qu'il veut comme attaque sur Samba, rien ne se passera.

Mais ce serveur n'est arrêté que tant que votre machine fonctionne. Maintenant, nous pouvons nous débrouiller pour que ce serveur ne soit plus du tout lancé aux prochains démarrages de la machine.

Sous Linux, pour démarrer automatiquement un serveur il suffit de créer un lien symbolique. Suivant que votre machine démarre en mode texte ou en mode graphique, il faut créer un lien symbolique dans respectivement les répertoires "/etc/rc.d/rc3.d/" ou "/etc/rc.d/rc5.d/", vers le script du serveur qui se trouve dans le "/etc/init.d". Concrètement, voila comment cela se passe :

```
[olivier@phoenix /]$ ls -la /etc/rc.d/rc3.d/ /etc/rc.d/rc5.d/
/etc/rc.d/rc3.d/ :
total 8
drwxr-xr-x    2 root     root         4096 jun 28 11:35 ./
drwxr-xr-x   10 root     root         4096 mar 27 20:23 ../
lrwxrwxrwx    1 root     root           20 mar 27 20:32 K15postgresql -> ../init.d/postgresql*
lrwxrwxrwx    1 root     root           16 mar 27 20:28 K55routed -> ../init.d/routed*
lrwxrwxrwx    1 root     root           23 mar 27 20:27 S01switchprofile -> ../init.d/switchprofile*
lrwxrwxrwx    1 root     root           18 mai 27 19:43 S03iptables -> ../init.d/iptables*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S10network -> ../init.d/network*
lrwxrwxrwx    1 root     root           15 mai  4 21:02 S10ulogd -> ../init.d/ulogd*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S11portmap -> ../init.d/portmap*
lrwxrwxrwx    1 root     root           16 mar 27 20:33 S12syslog -> ../init.d/syslog*
lrwxrwxrwx    1 root     root           14 mar 27 20:23 S17alsa -> ../init.d/alsa*
lrwxrwxrwx    1 root     root           15 mar 27 20:33 S18sound -> ../init.d/sound*
lrwxrwxrwx    1 root     root           16 mar 27 20:33 S20random -> ../init.d/random*
lrwxrwxrwx    1 root     root           13 mar 27 20:24 S20xfs -> ../init.d/xfs*
lrwxrwxrwx    1 root     root           12 mar 27 20:23 S30dm -> ../init.d/dm*
lrwxrwxrwx    1 root     root           13 mar 27 20:28 S40atd -> ../init.d/atd*
lrwxrwxrwx    1 root     root           19 mar 27 20:26 S40saslauthd -> ../init.d/saslauthd*
lrwxrwxrwx    1 root     root           15 avr  3 23:23 S55named -> ../init.d/named*
lrwxrwxrwx    1 root     root           16 mar 27 20:32 S56xinetd -> ../init.d/xinetd*
lrwxrwxrwx    1 root     root           15 avr  3 20:02 S65dhcpd -> ../init.d/dhcpd*
lrwxrwxrwx    1 root     root           18 mar 27 20:33 S75keytable -> ../init.d/keytable*
lrwxrwxrwx    1 root     root           17 mar 27 20:26 S80postfix -> ../init.d/postfix*
lrwxrwxrwx    1 root     root           15 mar 27 20:25 S85httpd -> ../init.d/httpd*
lrwxrwxrwx    1 root     root           17 mar 27 20:28 S85numlock -> ../init.d/numlock*
lrwxrwxrwx    1 root     root           15 mar 27 20:33 S90crond -> ../init.d/crond*
lrwxrwxrwx    1 root     root           15 mai 25 17:48 S90mysql -> ../init.d/mysql*
lrwxrwxrwx    1 root     root           13 mar 27 20:30 S91smb -> ../init.d/smb*
lrwxrwxrwx    1 root     root           17 mar 30 22:37 S92privoxy -> ../init.d/privoxy*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S95kheader -> ../init.d/kheader*
lrwxrwxrwx    1 root     root           16 mar 27 20:28 S99devfsd -> ../init.d/devfsd*
lrwxrwxrwx    1 root     root           11 mar 27 20:23 S99local -> ../rc.local*

/etc/rc.d/rc5.d/ :
total 8
```

```
drwxr-xr-x    2 root     root         4096 jun 28 11:35 ./
drwxr-xr-x   10 root     root         4096 mar 27 20:23 ../
lrwxrwxrwx    1 root     root           20 mar 27 20:32 K15postgresql -> ../init.d/postgresql*
lrwxrwxrwx    1 root     root           16 mar 27 20:28 K55routed -> ../init.d/routed*
lrwxrwxrwx    1 root     root           23 mar 27 20:27 S01switchprofile -> ../init.d/switchprofile*
lrwxrwxrwx    1 root     root           18 mai 27 19:43 S03iptables -> ../init.d/iptables*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S10network -> ../init.d/network*
lrwxrwxrwx    1 root     root           15 mai  4 21:02 S10ulogd -> ../init.d/ulogd*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S11portmap -> ../init.d/portmap*
lrwxrwxrwx    1 root     root           16 mar 27 20:33 S12syslog -> ../init.d/syslog*
lrwxrwxrwx    1 root     root           14 mar 27 20:23 S17alsa -> ../init.d/alsa*
lrwxrwxrwx    1 root     root           15 mar 27 20:33 S18sound -> ../init.d/sound*
lrwxrwxrwx    1 root     root           16 mar 27 20:33 S20random -> ../init.d/random*
lrwxrwxrwx    1 root     root           13 mar 27 20:24 S20xfs -> ../init.d/xfs*
lrwxrwxrwx    1 root     root           12 mar 27 20:23 S30dm -> ../init.d/dm*
lrwxrwxrwx    1 root     root           13 mar 27 20:28 S40atd -> ../init.d/atd*
lrwxrwxrwx    1 root     root           19 mar 27 20:26 S40saslauthd -> ../init.d/saslauthd*
lrwxrwxrwx    1 root     root           15 avr  3 23:23 S55named -> ../init.d/named*
lrwxrwxrwx    1 root     root           16 mar 27 20:32 S56xinetd -> ../init.d/xinetd*
lrwxrwxrwx    1 root     root           15 avr  3 20:02 S65dhcpd -> ../init.d/dhcpd*
lrwxrwxrwx    1 root     root           18 mar 27 20:33 S75keytable -> ../init.d/keytable*
lrwxrwxrwx    1 root     root           17 mar 27 20:26 S80postfix -> ../init.d/postfix*
lrwxrwxrwx    1 root     root           15 mar 27 20:25 S85httpd -> ../init.d/httpd*
lrwxrwxrwx    1 root     root           17 mar 27 20:28 S85numlock -> ../init.d/numlock*
lrwxrwxrwx    1 root     root           15 mar 27 20:33 S90crond -> ../init.d/crond*
lrwxrwxrwx    1 root     root           15 mai 25 17:48 S90mysql -> ../init.d/mysql*
lrwxrwxrwx    1 root     root           13 mar 27 20:30 S91smb -> ../init.d/smb*
lrwxrwxrwx    1 root     root           17 mar 30 22:37 S92privoxy -> ../init.d/privoxy*
lrwxrwxrwx    1 root     root           17 mar 27 20:33 S95kheader -> ../init.d/kheader*
lrwxrwxrwx    1 root     root           16 mar 27 20:28 S99devfsd -> ../init.d/devfsd*
lrwxrwxrwx    1 root     root           11 mar 27 20:23 S99local -> ../rc.local*
```

Dans ces 2 répertoires, on voit donc 2 liens "S91smb -> ../init.d/smb*" qui lancent le serveur Samba lors du démarrage de la machine. Nous allons donc les supprimer, et ainsi Samba ne sera plus jamais lancé :

```
[root@phoenix /]# rm /etc/rc.d/rc3.d/S91smb /etc/rc.d/rc5.d/S91smb
rm: détruire lien symbolique ‘/etc/rc.d/rc3.d/S91smb’? y
rm: détruire lien symbolique ‘/etc/rc.d/rc5.d/S91smb’? y
```

Mais d'un autre coté, nous venons aussi d'arrêter le partage de fichiers avec les autres machines Windows® de notre réseau... Hummm, ce n'est n'est pas très bien, car nous voudrions bien que Paradise récupère sur Phoenix la dernière documentation du kernel Linux que nous avons téléchargé.

Conclusion : autant arrêter des serveurs qui ne servent pas est primordiale, autant arrêter des serveurs qui sont utilisés est problématique. Non, décidément dans ce dernier cas ce n'est pas la solution. Il nous faudra faire autre chose. L'idéal serait de dire au serveur "Ne réponds qu'aux machines de sky.net, et ne répond pas aux autres". Oui, cela serait très bien. Mais comment faire ? C'est là qu'il faut plonger dans l'antre les fichiers de configuration, dans le "man" ou le "info", et paramétrer finement ses serveurs.

II-5-2 Restrictions des connexions aux ports : Généralités

Pour les sections qui vont suivre, je rappelle au lecteur que l'emplacement des fichiers de configuration et/ou les options qui sont utilisés ont été testés avec une distribution Mandrake 9.1. Pour les autres distribution Linux (Debian, Red Hat, Suze, Slackware, Gentoo, etc...) ou pour des futures versions de la Mandrake, ces fichiers peuvent éventuellement être placés ailleurs, et certaines options ne sont peut-être pas les mêmes. Par contre, ce qui est important c'est la méthodologie que vous trouverez ici. Elle vous permettra de retrouver par vous mêmes les bonnes options de configuration.

Dans les fichiers de configuration des divers serveurs, les "mots magiques" qui vont permettre de paramétrer le filtrage des demandes de connexion sont :

- Only from : Ce mot permet de restreindre les demandes de connexions à une catégorie d'adresse IP sources. On peut par exemple passer les paramètres paradise.sky.net, 192.168.0.5, 192.168.0.0/255.255.255.0, ou encore 192.168.0.0/24. Ainsi, seul certaines adresses IP / réseaux seront autorisés à lancer des requêtes.
- Allow : Idem que "From"
- Bind : Cette option indique en général l'adresse IP à qui est destinée la demande de connexion. Pour Phoenix par exemple, il y a 2 cartes réseaux, donc 2 adresses IP sur lesquelles peuvent arriver des requêtes externes (comme par exemple celles envoyées par "nmap"). Dans ce cas, on peut donner le paramètre phoenix0.sky.net ou 192.168.0.1 à l'option "bind", ce qui aura pour effet de refuser les demandes de connexions faites à l'adresse IP phoenix1.internet.net. Et donc les requêtes venant de pirate.internet.net seront refusées.
- Listen : Idem que "Bind".
- Interface : Idem que "Bind".

Cette liste n'est pas exhaustive, mais elle vous donne un bon point de départ pour vous aider. A vous maintenant d'éplucher la documentation des serveurs que vous utilisez, afin de trouver les options équivalentes.

Une règle importante pour ce type de paramétrage, c'est que l'on n'est jamais trop paranoïaque. Ainsi, n'hésitez pas à utiliser en même temps les paramètres de types "bind" et "from", afin de doubler la sécurité , et de contrôler à la fois les adresse IP source et destination des requêtes.

Maintenant, regardons pour certains serveurs spécifiques comment cela fonctionne. Au risque de vous déplaire, je ne parlerai que des fichiers de configuration, et non des interfaces graphiques qui permettent de saisir agréablement ces paramètres. Tout simplement parce que je ne les utilise pas, et que je travaille plus vite en configurant à la main les fichiers de configuration. Certains dirons que j'ai des pratiques de papy de l'informatique, auquel je répondrais que moi, je ne me suis pas fait greffé une sourie à la place de la main ... Mais n'entamons pas là un nouveau troll, et passons directement à la suite...

II-5-3 Samba / NetBIOS

On a déjà parlé précédemment de Samba. C'est un serveur qui permet de partager des répertoire de votre disque dur avec des machines Windows®. D'ailleurs, les réseaux Microsoft et Samba utilisent le même protocole. Pour étouffer dans l'oeuf toute confusion : c'est IBM qui a inventé NetBIOS. Puis il a été implémenté dans Windows® 3.11. Et un peu plus tard par Samba.

Fichier de configuration	/etc/samba/smb.conf
Documentation	man smb.conf
Site web	http://samba.org/

Paramétrages	Type de restriction	Option	Restriction sur une adresse IP Exemple : 192.168.0.2	Restriction sur un réseau Exemple : 192.168.0.0/24	Exemples	Remarque

	Source	hosts allow	Oui	Oui	192.168.0.2 192.168.0.0/24	On peut aussi utiliser le terme "EXCEPT"
	Destination	interfaces	Non	Oui	192.168.0.0/24	Utiliser aussi l'option "bind interface only" en temps que 3ème sécurité...
Remarques	L'option "remote announce" est aussi très intéressante					

Exemple :

```
hosts allow = 192.168.0.0/255.255.255.0 10.0.0.0/255.0.0.0
interfaces = 192.168.0.1/26 10.0.0.1/8
bind interfaces only = yes
```

Si vous utilisez un serveur Samba, vous devez à tout prix le sécuriser. C'est la cible privilégiée des intrus sur Internet. En effet, Windows® est fournit en standard avec un serveur NetBIOS, et bon nombre d'Internautes l'utilisent pour partager des fichiers dans leur réseau local (par exemple, avec leur portable professionnel). Et comme ils n'ont pas forcément conscience de ce problème d'ouvertures de ports, ils partagent leur répertoire sur Internet... Ainsi, tout le monde ou presque a accès en lecture, voir en écriture, sur certains de leurs répertoires, voir sur la totalité de leur disque dur...

Pour information, lorsque je suis connecté sur Internet, mon Samba est sollicité environ une fois par minute par des intrus... Ne faites donc pas cette erreur de débutant, et contrôlez au plus vite l'accès à ce port !

II-5-4 Apache

Non, nous n'allons pas jouer aux cowboys et aux indiens. Apache est le nom du serveur web (HTTP) utilisé par environ 50% des serveurs Internet. Il est robuste, en continuel amélioration, mais c'est aussi une cible privilégié des intrus. Je ne parlerai ici que de la version 2.0.

Ce n'est pas très habituel d'avoir un serveur Apache qui tourne sur une machine personnelle, mais qu'est-ce que c'est pratique ! Le mien sert à afficher mes pages HTML en cours de construction, exactement comme si elle étaient sur mon site sur Free. De plus, ce serveur fait tourner une version locale de l'excellent validator HTML, ce qui me permet de tester à tout moment la validité de mes pages (et d'avoir le petit logo en bas de page "Valid XHTML").

Mais ce n'est pas parce que je publie des pages HTML, que je laisse rentrer des intrus sur ma propre machine afin de les visiter avant l'heure. Et puis de toute façon, il y a des pages que je ne désire pas publier... Il convient donc de restreindre utilisation de mon serveur Apache.

Fichiers de configuration	/etc/httpd/conf/httpd2.conf /etc/httpd/conf/httpd2-perl.conf /etc/httpd/conf/common/httpd.conf					
Documentation	man http					
Site web	http://www.apache.org/					
Paramétrages	Type de restriction	Option	Restriction sur une adresse IP Exemple : 192.168.0.2	Restriction sur un réseau Exemple : 192.168.0.0/24	Exemples	Remarque
	Source	Deny from	Oui	Oui	All 192.168.0.2 192.168.0.0/24	Ce paramétrage se fait dans les sections <Directory xxxx> </Directory xxxx>
	Destination	Listen	Oui	Non	192.168.0.1 phoenix0:80	On peut utiliser l'option "Listen" plusieurs fois : Une par adresse IP où l'on veut écouter
Remarques	Apache est un logiciel que l'on peut vraiment beaucoup paramétrer. De ce fait, il faut faire attention à ne pas laisser des "portes ouvertes".					

Exemple :

```
[Fichier : /etc/httpd/conf/httpd2.conf]
Listen phoenix0:80
Listen phoenix1:80

[Fichier : /etc/httpd/conf/common/httpd.conf]
<Directory />
  AllowOverride None
  <IfModule mod_access.c>
    Order deny,allow
    Deny from all
  </IfModule>
</Directory>
```

II-5-5 Xinetd

On a déjà parlé plus haut de "Xinetd". Son fichier de configuration est "/etc/xinetd.conf", mais en général il est écrit "includedir /etc/xinetd.d/" dans ce fichier. Cela veut dire que "Xinetd" lira à la fois "/etc/xinetd.conf" mais aussi les fichiers de "/etc/xinetd.d/*" pour définir sa configuration. L'idée de ce système un peu atypique est :

- De définir un paramétrage global pour tout les serveurs que va surveiller Xinetd (" /etc/xinetd.conf", dans une section "defaults")
- Définir ensuite un paramétrage indépendant par serveur (" /etc/xinetd.d/*")

Cependant, si cette stratégie ne vous plaît pas, vous pouvez écrire tous les paramétrages des serveurs dans "/etc/xinetd.conf".

Fichiers de configuration	/etc/xinetd.conf /etc/xinetd.d/					
Documentation	man xinetd.conf					
Site web	http://www.xinetd.org					
Paramétrages	Type de restriction	Option	Restriction sur une adresse IP Exemple : 192.168.0.2	Restriction sur un réseau Exemple : 192.168.0.0/24	Exemples	Remarque
	Source	only_from	Oui	Oui	192.168.0.2 192.168.0.0/24 192.168.0.0/255.255.255.0	
	Destination	bind	Oui	Non	192.168.0.1 phoenix0	

Remarques	Dans la section "default" de Xinetd, je vous conseille vivement d'utilisation l'option "no_access =" en ne donnant aucun paramètre. Ainsi, à moins de spécifier plus loin un accès à un serveur en particulier, Xinetd refusera par défaut toute tentative de connexion. C'est une 3ème sécurité...

Exemple de configuration pour le serveur proftpd :

```
[Fichier : /etc/xinetd.conf]
defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
# Remarque : On n'écrit rien en paramètre, et c'est VOLONTAIRE ! Ainsi, TOUS les accès sont interdit...
        no_access               =
}
includedir /etc/xinetd.d

[Fichier : /etc/xinetd.d/proftpd-xinetd]
# Service FTP pour phoenix0.sky.net (réseau local)
service ftp
{
        id                      = ftp:0
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/proftpd
        flags                   = REUSE
        log_on_failure          += USERID
        log_on_success          += USERID DURATION
        nice                    = 10
        instances               = 4
        bind                    = phoenix0
        only_from               = 192.168.0.0/24
        disable                 = no
}
```

Commentaire :

- Première protection ("no_access = ") : personne n'a accès aux serveurs protégés par Xinetd
- Seconde protection ("bind = phoenix0") : seul les requêtes arrivant sur phoenix0.sky.net sont autorisées.
- Troisième protection ("only_from = 192.168.0.0/24") : le client doit avoir une adresse IP appartenant au réseau sky.net.

Trois protections en un. Puissant, non ??

II-5-6 X11

Lorsque vous faire un "nmap" sur votre machine, vous remarquez qu'un port est ouvert beaucoup "plus haut" que les autres. Il s'agit du port 6000, qui est associé au système graphique X11. C'est lui qui est responsable de l'affichage des beaux pixels qui s'affichent devant vous pendant que vous lisez ce document.

```
[olivier@phoenix /]$ nmap phoenix0.sky.net

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on phoenix.sky.net (192.168.0.1):
(The 1591 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
110/tcp    open        pop-3
111/tcp    open        sunrpc
443/tcp    open        https
6000/tcp   open        X11
```

Pourquoi un logiciel d'affichage a t'il un port IP d'ouvert ? En fait c'est là que Linux (et les Unix/BSD en général) est génial. Avec Linux, vous pouvez lancer un programme, et afficher son résultat non pas sur votre écran, mais sur l'écran d'un autre PC, en utilisant le réseau IP. Rien d'exceptionnel me diriez vous, Windows® le fait depuis sa version 2000. Oui, mais les Unix ont eu cette technologie plus de 20 ans avant les OS de Microsoft...

Il est important de fermer ce port, car des intrusions peuvent venir plus ou moins facilement dessus. Et même si ce n'est pas le cas, ce port ouvert indique très clairement à l'intrus que vous avez un système Linux / Unix / BSD. Et ce n'est pas la peine de le mettre trop rapidement sur la voie.

Fichier de configuration (exemple pour xdm)	/etc/X11/xdm/Xservers
Documentation	man Xserver
Site web	http://www.xfree86.org/

La méthode protection de X11 est beaucoup plus primitive que celles des autres logiciels dont nous parlons ici. En fait, elle n'a que deux options : Le port 6000 est ouvert ou fermé. En général, sur un réseau de particulier on n'utilise pas ce système d'affichage exporté ("export display" en anglais et c'est d'ailleurs le nom d'un variable d'environnement qui est justement utilisé pour cela). C'est pourquoi on peut fermer ce port en toute quiétude.

C'est là que cela devient un peu plus compliqué. Le manuel ("man X11") indique que la commande "X" ("/usr/X11R6/bin/X" en fait) peut recevoir le paramètre "-nolisten tcp". Comme cela l'indique, X11 ne va plus écouter le port TCP, et donc le port sera fermé. Oui, mais en général, on ne lance pas "X" directement, car il y a d'autres couches logiciels à initialiser d'abord, comme le "Window Manager" ("gestionnaire de fenêtres" en français), le "Desktop" ("bureau" en français), etc... Ainsi :

- Si vous démarrez votre machine en ligne de commande ("init 3"), vous avez l'habitude de taper la commande "startx" pour lancer l'interface graphique. Lancez donc plutôt : "/usr/X11R6/bin/startx -- -nolisten tcp"

- Si au contraire vous démarrez directement en mode graphique ("init 5"), cela va dépendre du gestionnaire de fenêtres qui est utilisé. Si il s'agit de "xdm" ("/usr/X11R6/bin/xdm"), rajoutez "-nolisten tcp" à la ligne lançant "X", dans le fichier "/etc/X11/xdm/Xservers". Ainsi,

  [Fichier : /etc/X11/xdm/Xservers]
  ```
  :0 local /bin/nice -n -10 /usr/X11R6/bin/X -deferglyphs 16
  ```

  devient :

  [Fichier : /etc/X11/xdm/Xservers]
  ```
  :0 local /bin/nice -n -10 /usr/X11R6/bin/X -nolisten tcp -deferglyphs 16
  ```

  Pour les autres gestionnaires de fenêtres, je vous laisse chercher par vous même !

  Pour les utilisateurs de distributions Mandrake, prenez bien garde de mettre le "-nolisten tcp" juste après le "/usr/X11R6/bin/X". En effet, sur cette distribution tourne un logiciel ("/usr/sbin/msec") qui surveille à heure régulière les changements de "/etc/X11/xdm/Xservers". Il supprimera l'option "-nolisten tcp" si elle n'est pas mise à l'emplacement décrit. "Msec" considère que tout changement non référencé à ce fichier, et à tout un tas d'autres, sont des atteintes potentielles à la sécurité de la machine, ce en quoi il a raison... Pour les détails, je vous laisse regarder le fichier de configuration de msec : "/usr/share/msec/libmsec.py"

Enfin, si vous avez lancé plusieurs serveurs X11 sur votre machine (c'est assez rare quand même...), un port situé au-delà de 6000 sera ouvert par serveur X11 supplémentaire : 6001, 6002, 6003, etc...

II-5-7 Bind / Named

On a déjà parlé de named, donc je ne m'étendrai pas dessus. Le nom du produit s'appelle "Bind", mais l'exécutable se nomme "named", c'est un peu une source de confusion... L'important si vous avez un serveur DNS sur votre réseau interne, c'est que personne de l'extérieur ne s'amuse à l'interroger, et ne devine la topologie de votre réseau (le nom et les adresses IP des machines à l'intérieur). Avec une configuration non sécurisée, c'est beaucoup plus facile à faire qu'on le croît. D'un autre coté, il est assez rare d'avoir besoin d'un serveur DNS dans un réseau personnel, sauf si on fait du NAT. Mais n'anticipons pas.

Fichier de configuration	/etc/named.conf					
Documentation	man named.conf					
Site web	http://www.isc.org/products/BIND/					
Paramétrages	Type de restriction	Options	Restriction sur une adresse IP Exemple : 192.168.0.2	Restriction sur un réseau Exemple : 192.168.0.0/24	Exemples	Remarque
	Source	allow-transfer allow-query allow-update allow-recursion	Oui	Oui	192.168.0.2 192.168.0.0/24	Voir la doc pour l'explication des différentes options
	Destination	listen-on	Oui	Oui	192.168.0.1 192.168.0.0/24	
Remarques	Bind fait parti de ces logiciels qui sont soumis à des attaques continuels, car fort utilisés. Il convient donc de multiplier les protections lorsqu'on l'utilise, et de le mettre à jour régulièrement.					

Exemple :

```
allow-transfer {
  192.168.0.0/24;
};

allow-query {
  192.168.0.0/24;
};

listen-on {
  192.168.0.0/24;
};
```

II-5-8 Postfix

"Postfix" est un serveur de mails, tout comme le célèbre "sendmail". Il est plus facile à configurer, et peut servir aussi bien à l'intérieur de réseaux modestes ou au sein des grands comptes.

Fichier de configuration	/etc/postfix/main.cf					
Documentation	man postfix					
Site web	http://www.postfix.org/					
Paramétrages	Type de restriction	Option	Restriction sur une adresse IP Exemple : 192.168.0.2	Restriction sur un réseau Exemple : 192.168.0.0/24	Exemples	Remarque
	Source	mynetworks	Non	Oui	192.168.0.0/24	
	Destination	inet_interfaces	Oui	Non	192.168.0.1 phoenix0.sky.net	
Remarques	Sur Internet sévi depuis un bon bout de temps une activité à peine plus inconvenante que l'intrusion. Il s'agit du "spam", "courrier non sollicité" ou "pourriciel". En temps qu'utilisateur, vous devez garantir que votre machine ne servira pas à envoyer des mails non sollicités de ce type, c'est à dire que ce ne soit pas un "open relay".					

Exemple :

```
mynetworks_style = subnet
mynetworks = 192.168.0.0/24
inet_interfaces = phoenix0.sky.net
```

II-6 Bilan

A la fin de paramétrage de chaque serveur, n'hésitez pas à utiliser "nmap" afin de vérifier que seuls les machines ou les réseaux en qui vous pouvez avoir confiance (les réseaux locaux, quoi !!) accèdent à vos ports.

Remarquez bien que ces options de configurations n'assurent pas toute la sécurité de votre serveur, et encore moins celle des données qui y sont dessus. Par exemple, si vous partagez votre "/" avec Samba, NFS, ou quoi que ce soit d'autre, c'est vous qui prenez la responsabilité que l'on accède à plus d'informations que vous ne le vouliez : Il vaut mieux en effet ne partager qu'un seul petit répertoire de votre disque dur, par exemple "/home/olivier/partage". Autre exemple : Si votre serveur Apache est autorisé à exécuter des CGI-BIN, c'est encore vous qui prenez la responsabilité de ce que ces CGI-BIN peuvent faire... Donc la règle est simple : Lorsque vous lancez un serveur sur votre machine, ayez bien conscience de ce qu'il peut faire, et paramétrez le en conséquence.

Bien, maintenant que nous avons bien configuré nos serveurs, et qu'un "nmap pheonix1.internet.net" n'affiche aucun port ouvert, nous nous sentons en sécurité. Oui, mais, est-ce vraiment justifié ? Oui ? Vraiment ? Et bien non !!! Dommage, hein ? Sinon, il n'y aurait pas de chapitre suivant, et ce document ne commencerait pas par "Firewall ...". 😀

Bon, trêve de palabres. Refaites le plein de boissons non alcoolisées, prenez quelques aspirines, et passez au chapitre suivant. Nous allons voir comment domestiquer les flammes de l'enfer...

III NETFILTER / IPTABLES

PLAN :

III-1 Introduction

Dans le précédent chapitre, nous avons réussi à fermer tous les ports de notre machine, mais est-ce à cela que se résume la sécurité en informatique ? Même si ce point est très important, il reste insuffisant. Pourquoi ?

- Le ping : A aucun moment nous n'avons vu comment interdire à notre machine de répondre aux commandes de type "ping" ou "traceroute". En effet, ce n'est pas un serveur qui est responsable de répondre à ce type de sollicitation, mais le kernel lui-même via la couche IP. On peut donc à tout moment savoir que votre machine est allumée, ou effondrer son CPU sous une attaque DOS / DDOS.
- Ports fermés : Dans de rares cas, et bien que vos ports soient fermés, l'intrus peut soupçonner la présence de "quelque chose" derrière l'un de vos ports. J'en ai fait l'expérience il y a un bout de temps, lorsque j'ai voulu faire tester en ligne la sécurité de ma machine par le site http://www.pcflank.com/ . J'ignore comment ce serveur a fait, mais il a réussi à trouver pour l'un ou l'autre de mes serveurs une trace d'activité. Et donc il a indiqué qu'il y avait probablement "quelque chose" derrière ce port, ce en quoi il avait raison. Mais avec les informations que vous trouverez dans ce chapitre, c'est le genre de chose qui n'arrivera plus !
- L'IP spoofing : Dans les différents paramétrages que nous avons vu précédemment , nous avons configuré les serveurs pour qu'ils ne fassent confiance qu'à certaines adresses IP sources. Oui, mais que se passe t'il si l'intrus ment délibérément sur son adresse IP source, et envoie une demande de connexion semblant venir de votre propre réseau local ? C'est tout à fait faisable, et c'est ce que l'on appelle l'IP spoofing... Cette technique a un inconvénient, c'est que l'intrus ne pourra pas recevoir les réponses de la machine cible (car celle-ci renverra l'information là où elle semble venir, c'est à dire sur le réseau local). Mais il n'empêche que cela peut suffit au pirate à envoyer des paquets d'informations qui peuvent corrompre le serveur, via la technique du "buffer overflow" ("Dépassement de tampon" en français). Internet fourmille d'exemples réussis de ce type d'attaque, et seul une mise à jour régulière de ses logiciels serveur peut réussir à les stopper.

En fait, ce qui ne va pas avec le système de sécurité que nous avons vu jusqu'à présent, c'est que :

- Nous n'avons aucun contrôle sur l'origine précise des requêtes IP. Nos serveurs ne peuvent en général pas savoir si une trame IP arrive ou part d'une interface où d'une autre.
- Nous ne pouvons pas empêcher notre machine de répondre à certaines requêtes.
- Nous aimerions avoir une trace des activités réseaux suspectes de notre machine, et donner l'alerte le cas échéant.

Tous ces points trouvent leur solution avec un seul mot : Netfilter, le firewall du kernel Linux 2.4 et supérieur.

Netfilter est le nom d'une partie du kernel Linux qui est destinée à assurer la surveillance de tous les transferts de données réseaux. Sa tache est de faire du "Network Packet Filtering", c'est à dire du "Filtrage de Paquets Réseaux". Grosso modo, il se place entre la couche réseau du kernel Linux, et la couche applicative (c'est plus compliqué que cela en faite, mais pour les besoins de ce document, cette approximation n'est pas fausse 😀). Netfilter supporte actuellement les protocoles IPv4, IPv6, DECnet, et ARP, et en partie IPX via des patchs expérimentaux. Nous ne nous intéresserons ici qu'à IPv4.

Comme Netfilter est un élément implanté profondément dans le kernel Linux (le "kernel space", ou "l'espace du coeur" en français), on ne peut pas le configurer aussi simplement qu'avec un jolie interface graphique doté de pleins de couleurs. On ne peut pas non plus le paramétrer directement via un fichier de configuration du "/etc/". Non, l'unique moyen que nous ayons de dialoguer avec lui est un programme appelé "iptables", que l'on lance dans le "user space" ("espace utilisateur" en français) avec les droits root. Nous verrons par la suite qu'il existe un certain nombre d'autres applications, graphiques ou non, qui servent d'interface à "iptables".

Enfin, et au risque de me répéter, Netfilter / Iptables ne sont que des éléments fonctionnant avec un kernel Linux 2.4 et supérieur. Si vous utilisez un kernel Linux de type 2.2 ou inférieur, vous devez utiliser une applications appelée " ipchains". Mais cela sort malheureusement du cadre de cette documentation. Linux évolue très vite, et ses outils aussi. Pour moi, le seul fait que les kernels 2.4 et supérieurs supportent le suivi de connexion justifient totalement leur utilisation.

Donc pour les utilisateurs du kernel 2.2 ou moins, mettez à jour votre kernel si vous voulez exploiter le contenu de ce chapitre. Mais la mise à jour du kernel Linux est une opération lourde. Je ne suis donc absolument pas responsable des conséquences qui pourraient en découdre !

III-2 Pré-requis

Si vous utilisez une distribution Mandrake récente, tous les outils dont vous avez besoin sont disponibles. Vérifiez simplement que vous le package "iptables" est bien installé sur votre machine.

```
[olivier@phoenix /]$ rpm -q iptables
iptables-1.2.7a-2mdk
```

Le cas échéant, installez-le :

```
[root@phoenix /]# urpmi iptables
```

Pour les autres distributions, vous pouvez utiliser la technique habituelle d'installation de paquets de votre système : "rpm", "urpmi", "apt", interface graphique, etc... Mais vous devez en savoir plus que moi dans ce domaine.

Pour ceux qui veulent compiler eux même leur kernel à partir des sources (Hummm, vous n'êtes pas si néophyte que cela alors !), vous devez utiliser les options suivantes :

- Enable loadable module support (CONFIG_MODULES) : Y
- Packet socket (CONFIG_PACKET) : Y
- Network packet filtering (replaces ipchains) (CONFIG_NETFILTER) : Y
- Unix domain sockets (CONFIG_UNIX) : Y
- TCP/IP networking (CONFIG_INET) : Y
- IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) : Y
- Connection tracking (required for masq/NAT) (CONFIG_IP_NF_CONNTRACK) : M
- FTP protocol support (CONFIG_IP_NF_FTP) : M
- IRC protocol support (CONFIG_IP_NF_IRC) : M
- IP tables support (required for filtering/masq/NAT) (CONFIG_IP_NF_IPTABLES) : M
- Packet filtering (CONFIG_IP_NF_FILTER) : M
- Full NAT (CONFIG_IP_NF_NAT) : M
- Limit match support (CONFIG_IP_NF_MATCH_LIMIT) : M
- Connection state match support (CONFIG_IP_NF_MATCH_STATE) : M
- MASQUERADE target support (CONFIG_IP_NF_TARGET_MASQUERADE) : M
- REDIRECT target support (CONFIG_IP_NF_TARGET_REDIRECT) : M
- LOG target support (CONFIG_IP_NF_TARGET_LOG) : M
- ULOG target support (CONFIG_IP_NF_TARGET_ULOG) : M

Les options indiquées ici sont destinées à un kernel 2.4.20. Pour d'autres kernels de type 2.4, cela doit être équivalent.

Suivant vos habitudes, vous pouvez compiler certains élément dans le corps du kernel Linux (option "Y"), plutôt qu'en module ("M").

Une fois vos options sélectionnées, à vous le "make deps", "make vmlinux", "make modules", "make modules_install" et "make install", puis le reboot. Mais vous devez connaître la chanson, non ? Si ce n'est pas le cas, recherchez sur Internet comment on compile un kernel Linux.

Enfin, vous devez avoir l' outil "iptables" qui soit en adéquation avec la version du kernel Linux que vous utilisez. Si vous avez tout, lancez "iptables -L" en temps que root. Vous devez avoir quelque chose ressemblant à ceci :

```
[root@phoenix /]# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

La commande "iptables -L" affiche l'état des tables de Netfilter. Késako tout cela ? Le paragraphe suivant va l'expliquer...

III-3 Vue générale de Netfilter

Je pense qu'à ce niveau là de la documentation, les utilisateurs du monde Windows® (y en a t'ils qui lisent cette documentation ?) doivent se demander dans quel monde de fous ils sont tombés ... Et bien oui, sous Windows® c'est beaucoup plus simple d'installer un firewall de typeZoneAlarm® Tiny Firewall® ou autre... Un double clique sur le "setup.exe", et hop cela marche du premier coup. Cela marche, oui, mais est-ce que c'est fiable ? Parce qu'à la porte de votre PC, il y a un bon nombre de virus-vers qui n'attendent qu'une erreur de votre part pour désactiver vos firewall, anti-virus et autres programmes de ce protection.

En fait, on ne peut pas comparer le mode de fonctionnement des firewalls sous Windows® et sous Linux tellement ils sont différents.

- Sous Windows® c'est l'OS qui met à disposition les paquets IP qu'il désire(*) dans l'espace utilisateur ("user space"), où un programme externe ("ZoneAlarm®", "Tiny Firewall®", etc...) décide si oui ou non ce paquet doit être accepter pas le système. Tout cela est bien joli, mais que se passe t'il si un autre programme de la machine, un virus par exemple, attaque le firewall et le force à s'éteindre ? La machine perd tout simplement sa protection... Ne souriez pas, ce type d'attaque est très en vogue à l'heure où j'écris ces lignes. En effet c'est parfaitement faisable, car la plus grande partie des utilisateurs de Windows® utilisent leur machine avec les droits administrateurs (même si ils ne sont pas connectés spécifiquement sous ce nom). Ou pire encore, ils utilisent les versions 95, 98, ou Millenium, qui n'ont aucune notion de restriction de droits. Enfin, si c'est le même utilisateur qui utilise la machine et qui a démarré le firewall, alors tout programme lancé, par inadvertance ou non, par cet utilisateur pourra arrêter le firewall...
  Quand au fonctionnement du firewall intégré dans la version XP de Windows®, même si il *semble* intégré dans l'OS, il est similaire aux logiciels de Firewall dont nous venons de parler. Donc c'est un produit insuffisant et à éviter...
- Sous Linux par contre, tout reste au niveau de l'espace kernel ("kernel space"), c'est à dire qu'à l'exception d'"iptables", aucun programme ne peut interférer avec Netfilter. Comme il faut impérativement avoir les droits de root pour lancer "iptables", et que bien entendu vous n'utilisez que rarement les droits root, vous pouvez être tranquilles.

(*) : C'est juste une petite remarque auquel j'invite l'utilisateur de Windows® à réfléchir quelques secondes. Qu'est-ce qui vous prouve que toutes les trames IP qui rentrent ou qui sortent de votre Windows® passent bien par la validation du firewall que vous avez installé ? Le code source de Windows® ? Vous l'avez vu ? La belle affaire : Même si c'est le cas, je vous rappelle que vous avez un accord de non divulgation qui vous interdit de parler de ce que vous avez pu y trouver...

Il existe plusieurs manières d'aborder l'explication de "Nerfilter" : Soit rentrer dans les détails de sa structure interne, et dans ce cas là il vaut mieux prévoir quelques boîtes d'aspirines. Soit, et c'est ce que je vais tenter de faire, expliquer les principaux composants dont l'utilisateur a besoin, puis aborder au fur et à mesure les aspects plus techniques. Pour ceux qui veulent en savoir un peu plus sur le fonctionnement de Netfilter, je les invite à consulter la documentation du développeur de Netfilter (traduite en français).

Bien, êtes vous prêt pour la grande explication de Netfilter ? Oui ? OK, allons y alors.

Netfilter peut intervenir en 5 endroits du système de gestion de la pile IP. Pour chacune de ces étapes (que l'on appelle des "hook", pour "crochet" en français), Netfilter peut :

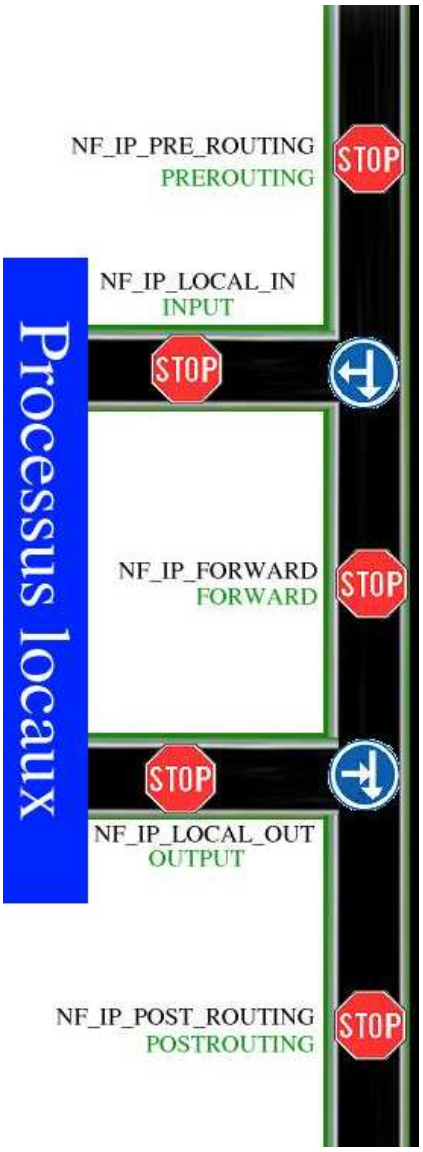
- Imposer au kernel de supprimer le paquet ("drop" en anglais). Auquel cas, il est jeté aux oubliettes, et c'est comme si le paquet n'avait jamais existé.
- Indiquer au kernel que le paquet est accepté. Dans ce cas là, le kernel peut continuer à travailler dessus.
- Modifier le paquet, puis le rendre au kernel.

Pour un programmeur, il est facile de comprendre ce qu'est un "hook". C'est l'équivalent dans la couche IP d'une interruption (matérielle ou logicielle) : Lorsque qu'un événement arrive, le système d'exploitation arrête les tâches en cours, et exécute un morceau de code particulier. Une fois que ce code est fini, le système continue son travail comme si rien ne s'était passé.

Pour un non-informaticien, le hook peut s'expliquer comme suit : Lorsque vous roulez en voiture, et qu'il y a par exemple un accident sur la route, les gendarmes vont bloquer le passage, et vous faire dévier sur un itinéraire de secours. A la sortie de cet itinéraire, et si tout se passe bien, vous réintégrerez votre route initiale. Et vous aurez effectué ce que l'on appelle vulgairement "faire un crochet". 😀

Pour chacun de ces "hook", Netfilter associe une chaîne. Mais qu'est-ce qu'une chaîne ? Une chaîne est un ensemble de règles (du type "si quelque chose alors je fais ceci") concernant les paquets IP : leur origine, leur destination, leur taille, etc... En fonction des différentes règles de la chaîne, Netfilter pourra décider quoi fait du paquet IP : Le laisser passer, le supprimer ou le modifier.

Reprenons notre analogie avec la route : Les paquets IP sont des voitures, qui circulent sur des routes. Si Netfilter n'est pas configuré spécialement, les voitures se déplacerons à travers les différentes routes de la pile IP, emprunteront les ronds-points de routage, puis finirons par sortir de la route IP. Par contre, si Netfilter est activé, il y aura des barrages de police en certains endroits du trajet des paquets (les fameux "hook"). En fonction des règles contenus dans ces "hook" (les fameuses chaînes, c'est à dire les signes "**STOP**"), le barrage décidera ou non de laisser passer le véhicule IP, voir de le modifier... D'ailleurs, c'est un peu ce qui se passe sur nos routes à nous. Il y a parfois des contrôles de police ou de gendarmerie : En fonctions de certaines règles (contrôle des papiers, de l'alcoolémie, de l'état du véhicule, du délit de "sale gueule", ...) l'officier pourra ou non décider de laisser passer le véhicule... 😀



Voyons maintenant les 5 "hook" et les 5 chaînes qui y sont associées :

Hook	Chaîne	Description
NF_IP_PRE_ROUTING	PREROUTING	A ce stade, le paquet est "brut de forme", c'est à dire qu'il n'a subit aucune modification par rapport à ce que l'interface réseau a reçu. On reparlera de ce hook un peu plus tard, donc ce n'est pas la peine de s'y intéresser tout de suite.
NF_IP_LOCAL_IN	INPUT	Ce "hook" est très intéressant, car à ce stade, le paquet est prêt à être envoyé aux couches applicatives, c'est à dire aux serveurs et aux clients qui tournent sur la machine. C'est un des points principaux sur lequel nous allons travailler.

NF_IP_FORWARD	FORWARD	"Forward" ("faire suivre" en français) est assez particulier. Ce "hook" voit passer des paquets IP qui vont transiter d'une interface réseau à une autre, sans passer par la couche applicative. Pourquoi diantre faire suivre un paquet entre 2 interfaces réseaux, comme par exemple entre "eth0" et "ppp0" ? En fait, c'est afin de permettre à Linux de se transformer en passerelle : Vous vous souvenez, c'est l'histoire du garde-barrière que nous avons vu dans le 1er chapitre. Comme pour "NF_IP_PRE_ROUTING", nous n'en reparlerons que plus tard.
NF_IP_LOCAL_OUT	OUPUT	Ce "hook" est l'équivalent du "NF_IP_LOCAL_IN", sauf qu'il est exécuté après que les couches applicatives aient traités, ou générés, un paquet IP. Tout comme "NF_IP_LOCAL_IN", c'est un point que nous allons voir en détail.
NF_IP_POSTROUTING	POSTROUTING	C'est l'équivalent du "NF_IP_PRE_ROUTING" pour les paquets IP sortants de la couche IP. A ce stade, les paquets sont prêt à être envoyés sur l'interface réseau. Comme pour "NF_IP_PRE_ROUTING" et "NF_IP_FORWARD", nous en parlerons plus tard.

Le graphique ci-dessus nous montre l'implantation des différents hooks. Le sujet de ce document étant la sécurité réseau, ce qui nous intéresse le plus est la possibilité de contrôler les paquets arrivant et sortant des différents logiciels de notre machine. C'est donc au niveau des "hooks" "NF_IP_LOCAL_IN" et "NF_IP_LOCAL_OUT" que nous allons nous pencher plus précisément, via la notion de filtrage ("filter" ou "filtering" en anglais).

La partie que nous venons de voir ici est une des plus technique de ce document. J'espère que vous êtes toujours là, frais et dispo pour le paragraphe suivant...

III-4 Les tables

Nous allons maintenant voir ce qu'est une table. Une table permet de définir un comportement précis de Netfilter. Une table est en fait un ensemble de chaînes, elles-mêmes composées de règles. Bref, une table va nous permettre de manipuler Netfilter, afin de lui faire faire des choses intéressantes.

Mais comment manipuler ces tables ? Je vous le donnes en 1000 : Avec le programme "iptables" dont je vous parles depuis le début de ce document pardi !

Il existe pour l'instant 3 tables (Filter, NAT et Mangle), d'autres pouvant être rajoutées à l'avenir. Nous allons nous intéresser principalement à la table "Filter", puis dans un 2nd temps à la table "NAT". Nous ne ferons qu'évoquer la table "Mangle", car elle n'a pas beaucoup d'intérêt ici.

III-4-1 La table Filter

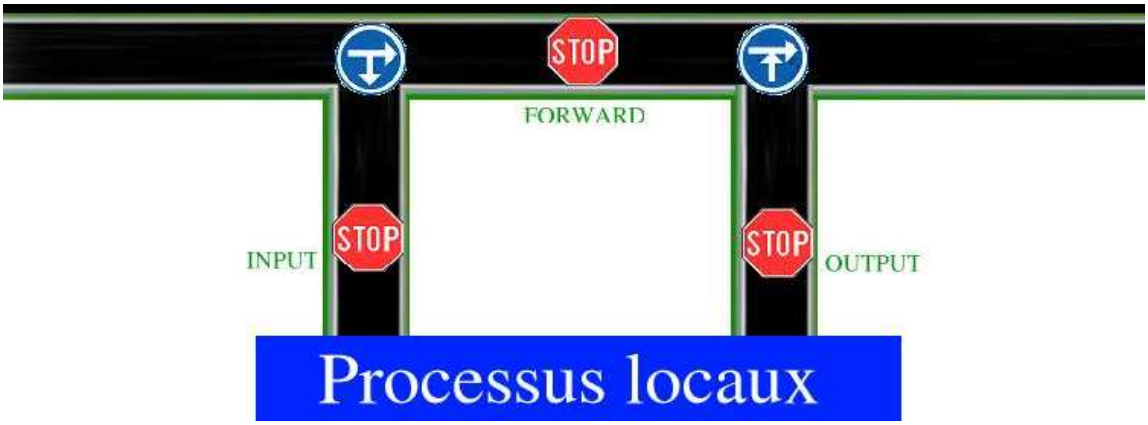
Comme son nom l'indique, cette table sert à filtrer les paquets réseaux. C'est à dire que nous allons pouvoir trier les paquets qui passent à travers le réseau, et supprimer ceux qui ne nous intéressent pas, ou que nous trouvons dangereux. Pour cela, la table "Filter" n'utilise que 3 chaînes :

- INPUT : Cette chaîne contrôle les paquets à destination des applications.
- OUTPUT : Elle analyse les paquets qui sortent des applications.
- FORWARD : Elle filtre les paquets qui passent d'une interface réseau à l'autre. Notez au passage que les paquets de ce type ne passent jamais par les chaînes INPUT et OUTPUT.

La philosophie du filtrage est très simple : Tout ce qui n'est pas explicitement autorisé est strictement interdit. C'est plutôt autoritaire comme système non ?

Pour cela, nous allons travailler en deux temps :

- Premièrement, interdire par défaut tous les paquets. C'est facile à faire, car les 3 chaînes que nous utilisons (INPUT, OUTPUT et FORWARD) ont une valeur par défaut. Donc par défaut, nous allons supprimer toutes les trames (on utilisera par la suite le terme de "DROP").
- Dans un second temps, nous n'allons autoriser que certains flux bien particuliers. Ce sera un juste équilibre entre la sécurité du système et les fonctionnalités dont nous avons besoin.



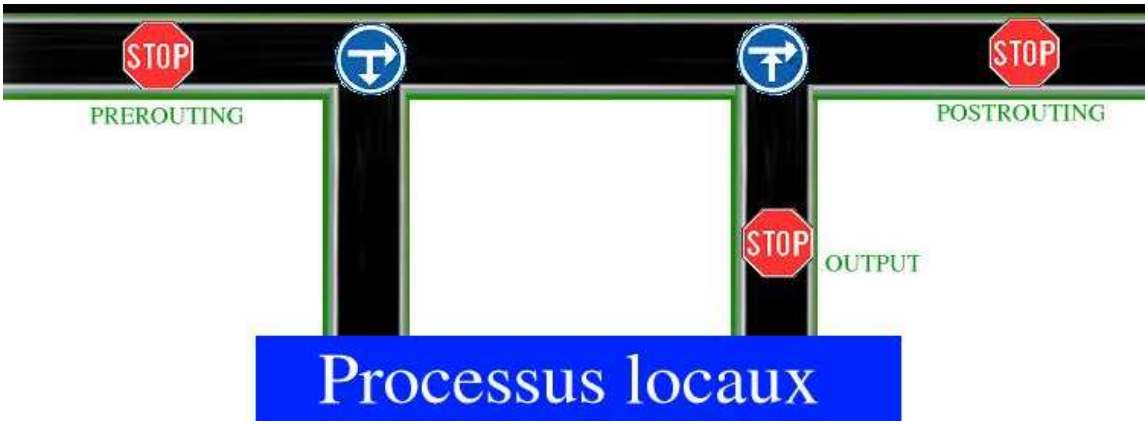
La table Filter : Le filtrage des paquets

III-4-2 La table NAT

La table NAT (Network Adress Translation, ou Traduction d'Adresses Réseau) nous fait sortir du cadre orienté strictement sur la sécurité de ce document. Mais ce qu'elle permet est tout bonnement essentiel pour la machine Paradise. En fait, elle va transformer notre machine Linux (Phoenix) en une passerelle Internet, ce qui permettra à Paradise de surfer sur Internet. Et dans un second temps, nous verrons comment faire suivre certains paquets arrivant sur Phoenix, pour les transmettre à Paradise.

Pour faire tout ceci, nous avons besoin là encore de 3 chaînes :

- PREROUTING : Les paquets vont être modifiés à l'entrée de la pile réseaux, et ce, qu'ils soient à destination des processus locaux où d'une autre interface.
- OUTPUT : Les paquets sortant des processus locaux sont modifiés.
- POSTROUTING : les paquets qui sont près à être envoyés aux interfaces réseaux sont modifiés.



La table NAT : Le passage de trames d'une interface à une autre

III-4-3 La table Mangle

Nous allons nous transformer en boucher, avide de bits à découper et à reconditionner... En effet, le terme "mangle" en anglais veut dire "mutilation"...

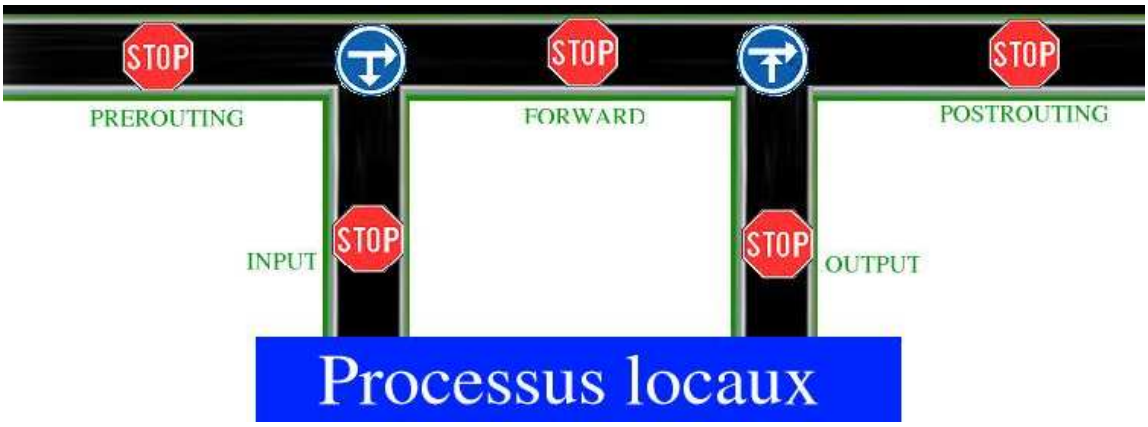
Que diable voulons nous faire à nos braves paquets réseaux ? En fait, il s'agit de les marquer en entrée de la couche réseau, afin que d'autres programmes de l'espace kernel ("kernel space") puissent en faire quelque chose. L'idée de cette technique est par exemple de fournir à Linux la possibilité d'avoir un contrôle sur les débits des flux de données entrants et sortants de la machine, afin de rendre certains flux plus prioritaires que d'autres. Du sectarisme au niveau réseau ? Oui, c'est tout à fait cela : Certains flux vont être volontairement privilégiés, afin de garantir un meilleur confort d'utilisation à l'utilisateur. Cette technique s'appelle le QoS ("Quality of Service" ou "Qualité de Service" en français).

Un exemple d'intérêt du QoS est le suivant : Pendant que vous téléchargez en FTP un gros fichier, comme une image ISO de la dernière distribution Linux à la mode, vous pouvez désirer continuer à surfer tranquillement sur Internet, sans être tout le temps pénalisé par votre gros téléchargement. Ceci sera possible avec le QoS, et rendant le contenu HTTP (port source 80) prioritaire par rapport à celui de votre téléchargement (port source 20).

Mais l'application de cette technique est assez complexe, et sort complètement du cadre de cette documentation. Cela sera, peut-être, l'occasion pour moi de faire une nouvelle documentation ? Qui sait, le sujet n'est pas dénué d'intérêt...

Dans les premiers kernels de la série 2.4, la table Mangle n'utilisait que 2 chaînes (PREROUTING et OUTPUT). Mais depuis le kernel 2.4.18 elle utilise toutes les chaînes de Netfilter :

- PREROUTING : Les paquets vont être marqués en entrée de la couche réseau, en fonction de certains critères, de type de service (grâce aux numéros de ports source et/ou de destination), d'adresses IP de source et/ou de destination, de taille des paquets, etc. Ces informations seront utilisés par un programme fonctionnant dans l'espace kernel.
- INPUT : Les paquets sont marqués juste avant d'être envoyés aux processus locaux.
- FORWARD : Les paquets passant d'une interface réseau à l'autre sont marqués.
- OUTPUT : Là, ce sont les paquets générés par les applications locales (un client web par exemple) qui vont être marqués, tout comme les paquets entrant dans la couche réseau.
- POSTROUTING : Les paquets prêt à être envoyés sur le réseau sont marqués. L'utilisation de cette chaîne dans la table Mangle n'est cependant pas très évident.



La table Mangle : Le marquage des paquets

Bien c'est fini pour les parties vraiment compliquées. Maintenant que nous en avons fini avec la théorie, passons à la pratique. Voyons comment nous pouvons remplir les chaînes, afin de paramétrer ces tables, et enfin sécuriser notre réseau.

III-5 Chaînes, règles et iptables

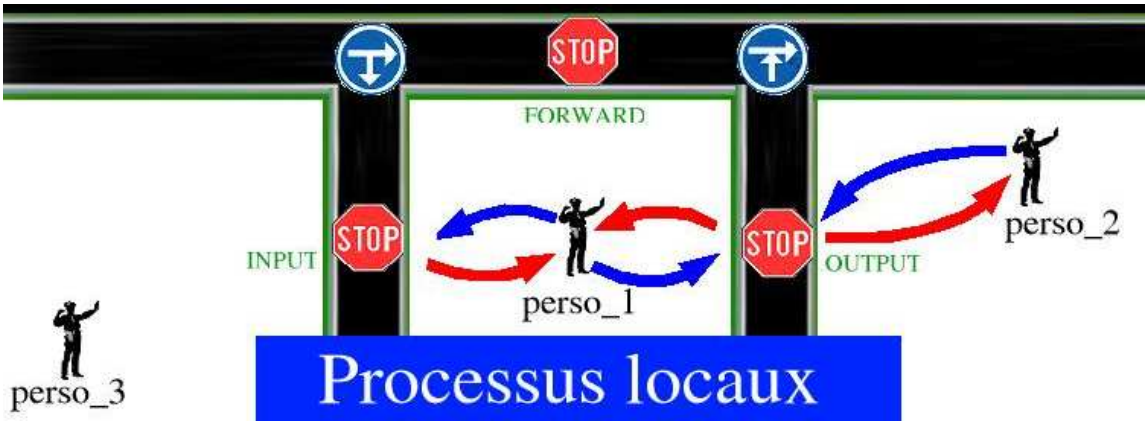
III-5-1 Les chaînes utilisateurs

Nous avons vu précédemment qu'il existe 5 principales chaînes, appelées aussi chaînes pré-définies , (PREROUTING, INPUT, FORWARD, OUTPUT et POSTROUTING). Ce sont les fameux signes "**STOP**" de nos illustrations. Et nous avons aussi vu que ces chaînes sont constituées d'un certain nombre de règles, qui forment une "check list" que Netfilter parcourra afin de décider quoi faire des paquets.

Ce que nous n'avons pas vu par contre, c'est qu'il est possible à l'utilisateur, au root pour être précis, de créer ses propres chaînes. Ce sont les chaînes utilisateurs. Pourquoi diable faire cela ? Nous ne sommes pas tous des programmeurs avides de milliers de ligne de code ?!?! Ne vous inquiétez pas, il ne s'agit pas là de programmer quoi que ce soit. Mais simplement de rajouter votre propre "check list" lors de l'un ou l'autre des "contrôles de gendarmerie". Ouf, vous avez eu chaud, pas vrai ?

Sur le dessin ci-dessous, on voit quelques chaînes utilisateurs qui sont crées pour la table "Filter". Comme vous le voyez, les chaînes utilisateurs peuvent :

- Êtres utilisées par une chaîne en particulier : perso_2
- Êtres appelées par plusieurs chaînes : perso_1
- Ne pas êtres appelées du tout (perso_3). A quoi cela sert il alors ? A rien tout simplement : le root qui a écrit ces chaînes là à oublié pourquoi il les as créées, et il a du s'endormir sur son clavier avant de faire le ménage...



Chaînes utilisateur : Des agents de sécurité privés

III-5-2 Règles et cibles

Les règles, comme leur nom l'indique, sont une série de critères auquel doivent ou non répondre les paquets. En fait, on peut associer une règle à un "délit de sale gueule". Si le paquet réseau ressemble à l'un ou l'autre des critères, alors la règle est appliquée. Les différentes règles d'une chaînes sont appliquées les unes à la suite des autres.

Les critères peuvent être multiples :

- Interface source ou destination.
- Adresse IP source ou de destination.
- Port source ou de destination.
- Type de trame.
- Nombre de paquets.
- Paquet marqué par la table Mangle.
- Etc.

Il peut y avoir autant de règles que l'on veut dans une chaîne, mais il est intéressant de limiter au maximum leur nombre, afin d'avoir une vue claire et précise de notre système de filtrage.

Enfin, à chaque règle est associée une action (ou "CIBLE" dans la nomenclature de Netfilter) à effectuer si la règle doit s'appliquer. C'est là que Netfilter agit, qu'il fait (enfin) quelque chose avec le paquet réseau. Les principales actions sont :

- DROP : Le paquet est détruit purement et simplement. C'est typique du "délit de sale gueule"...
- ACCEPT : Le paquet a une "bonne tête", il est donc autorisé à continuer à passer. Mais une autre règle située après la règle qui a accepté ce paquet peut très bien finalement décider de le supprimer.
- LOG / ULOG : Le paquet est autorisé à continuer de passer, mais ses caractéristiques sont notées au passage. En général, c'est qu'on estime que le paquet est "louche", et que l'on veut en prendre note. Mais plus souvent encore, on décidera de le supprimer. Car, en informatique, tout ce qui est louche est anormal, et tout ce qui est anormal doit être supprimé !
- MASQUERADE : Le paquet va être modifié, afin de dissimuler (de masquer en fait) certaines informations concernant son origine. Cette technique sera utilisée un peu plus loin.
- MARK : le paquet est marqué en y attachant une information. Ceci est principalement utilisé avec les tables "Mangle", donc nous n'y reviendrons pas dessus.
- Une chaîne utilisateur : Nous avons vu un peu plus haut qu'il existe des chaînes utilisateurs. Dans le cas de cette action, le paquet est envoyé à une chaîne définie par l'utilisateur, où il passera à travers de nouvelles règles. Ceci est illustré par les **flèches rouges** dans ce schéma ci. Si aucune décision n'est prise à la fin de la chaîne utilisateur, le paquet revient dans la chaîne courante, et passe à la règle suivante. C'est ce qu'indiquent les **flèches bleues** de ce même schéma .

III-5-3 Iptables

Iptables est donc une commande que seul le root peut lancer. Son but est de dialoguer avec Netfilter, afin de contrôler les règles des chaînes, dans le but de configurer les tables.

Iptables est la boîte à tout faire de Netfilter. Cette commande va pouvoir :

- Rajouter des règles / chaînes.
- Supprimer des règles / chaînes.
- Modifier des règles / chaînes.
- Afficher les règles / chaînes

Comme toute commande Linux qui se respecte, "iptables" est un programme qui se lance en ligne de commande, et qui attend de nombreux paramètres. Les lister tous n'est pas forcément très intéressant (et puis la commande " man iptables" doit bien servir à quelque chose, non ?), aussi ne verrons nous que les options les plus intéressantes.

Option	Paramètre	Paramètre optionnel ?	Explication	Exemple
-t (table)	"filter", "nat", "mangle"	Oui	Indique sur quelle table nous voulons travailler. Si aucun paramètre n'est fourni, c'est la table filter qui est sélectionnée par défaut. **Par la suite, si aucun paramètre "-t" n'est utilisé dans une commande "iptables",**	"iptables -t nat ..." permet de travailler sur ta table "NAT".

			c'est que cette commande est destinée à la table filter.	
-F (Flush)	"filter", "nat", "mangle"	Oui	Supprime toutes les règles d'une chaîne prédéfinie (tel "PREROUTING", "INPUT", "FORWARD", "OUTPUT" et "POSTROUTING"). Si aucun paramètre n'est donné, toutes les chaînes prédéfinies sont supprimés.	"iptables -F" supprime toutes les chaînes prédéfinies de la table "filter".
-X (eXclude)	[Chaîne utilisateur]	Oui	Supprime une chaîne utilisateur. Si il n'y a aucun paramètre, toutes les chaînes utilisateurs sont supprimées.	"iptables -X perso_3" supprime la chaîne utilisateur "perso_3".
-P (Policy)	"filter", "nat", "mangle"	Non	Définie la politique (ou cible) par défaut d'une chaîne. Seules les chaînes prédéfinies peuvent avoir un comportement par défaut. Cette cible ne sera appliquée qu'après l'exécution de la dernière règle de la chaîne.	"iptables -P INPUT DROP" supprime par défaut tout les trames dans la chaîne "INPUT". Si aucune règle plus "souple" n'est définie, aucun paquet réseau n'arrivera aux processus utilisateurs de notre machine. C'est efficace comme technique, mais peu fonctionnel.
-N (New)	[Chaîne utilisateur]	Non	Cette option crée une nouvelle chaîne utilisateur. Par la suite, des règles doivent êtres créées, afin de remplir cette chaîne. Sinon, elle ne sera pas très utile !	"iptables -N LogAndDrop" crée une chaîne utilisateur dont le nom laisse supposer que l'on va logger les paquets, puis les supprimer.
-A (Append)	[Chaîne]	Non	Ajoute une règle à une chaîne prédéfinie ou utilisateur. Nous allons utiliser majoritairement cette commande.	"iptables -A INPUT ..." ajoute une règle à la table des paquets entrant dans l'espace des programmes.
-D (Delete)	[Chaîne] [Numéro de règle]	Non	Supprime une règle dans une chaîne particulière. Le numéro de la règle peut être retrouvé avec la commande "iptables -L" ou "iptables -L -n"	"iptables -D OUTPUT 1 -t mangle" supprime la 1ère règle de la chaîne "OUTPUT" de la table "Mangle".
-L (Liste)	[Chaîne]	Oui	Affiche la liste des règles pour la chaîne indiquée. Ou, si aucune chaîne n'est indiquée, cela affichera les règles pour toutes les chaînes de la table indiquée. Note : Rajouter à "-L" les options "-n" et "-v" est très utile.	"iptables -L -n -v" affiche le maximum d'informations sur les règles de la table "filter".
-j (jump)	[Cible]	Non	Défini l'action à prendre si un paquet répond aux critères de cette règle. Les principales valeurs sont : ACCEPT, DROP, REJECT, LOG	"iptables -A OUPUT -j DROP" supprime tous les paquets sortant des processus locaux. Ca, c'est de la censure !!
-i (input)	[Interface réseau]	Non	Critère sur l'interface réseau dont provient le paquet	"iptables -A INPUT -i eth0 ..." filtre les paquets arrivant aux programmes et venant de l'interface réseau eth0.
-o (output)	[Interface réseau]	Non	Critère sur l'interface réseau d'où les paquets vont sortir	"iptables -A OUTPUT -o ppp0" filtre les paquets créés par les programmes et sortant sur Internet.
-s (source)	[!] [Adresse IP ou réseau]	Non	Critère sur l'adresse IP source du paquet	"iptables -t nat -A PREROUTING -i 192.168.0.0/24 ..." travaille sur le routage des paquets du réseau interne sky.net.
-d (destination)	[!] [Adresse IP ou réseau]	Non	Critère sur l'adresse IP de destination du paquet	"iptables -A OUTPUT -d 192.168.0.0/24 ..." filtre les paquets sortant de l'espace utilisateur, à destination du réseau sky.net.
-p (protocole)	[!] "all", "tcp", "udp", "icmp", ou un numéro	Non	Critère sur le type de trames utilisé dans le paquet. D'autres numéros de protocoles peuvent être utilisés. Voir votre fichier "/etc/procoles"	"iptables -A INPUT -p udp ..." filtre uniquement les paquets de type UDP.
--sport	[!] [Port ou service]	Non	Critère sur le port source des paquets IP	"iptables -A INPUT -sport 80 ... " filtre tout ce qui vient du port HTTP (80).
--dport	[!] [Port ou service]	Non	Critère sur le port de destination des paquets IP	"iptables -O OUPUT -dport ! 21 ... " filtre tout ce qui n'est pas à destination du FTP (21).
-m (module)	[Nom d'un module]	Non	Demande d'utiliser un module particulier	"iptables ... -m state ..." utilise le module de suivi de connexion ("state" veut dire "statut" en français).
--state	[!] "NEW", "ESTABLISHED", "RELATED", "INVALID"	Non	Cette option ne s'utilise que cumulée avec l'option "-m state", afin d'être utilisé pour le suivi de connexion	"iptables ... --state NEW,ESTABLISHED ..." filtre les paquets de nouvelles connexions, ou de connexions déjà établies via une " poignée de main ".
--log-prefix --ulog-prefix	[Un mot]	Non	Rajoute un commentaire pour les cibles LOG et ULOG	"iptables ... -j LOG --log-prefix toto" rajoutera le mot "toto" au log de cette règle.

Remarque : On trouve parfois dans la colonne "Paramètre" du tableau le paramètre " [!]". Ce paramètre peut se rajouter aux autres paramètres de l'option, afin d'indiquer la négation . Exemples :

- "iptables -I INPUT -p tcp -sport 80 -j DROP" : Supprime toutes les trames HTTP rentrant dans l'espace utilisateur.
- "iptables -I INPUT -p tcp -sport ! 80 -j DROP" : Supprime toutes les trames rentrant dans l'espace utilisateur, excepté celles de HTTP.

Dans la suite de ce document, je vous conseille de revenir régulièrement sur ce tableau, afin de bien comprendre le mécanisme des règles que nous allons utiliser. Lorsque ce tableau aura répondu à toutes vos questions, et qu'il ne suffira plus à répondre à votre soif de connaissances, je vous invite à taper un petit "man iptables" !! Place maintenant à la pratique d'iptables, et lançons nous dans les scripts !

III-6 Un premier script simple

Comme vous vous devez vous en douter à ce niveau du document, le paramétrage de notre firewall va se résumer à taper un certain nombre de commandes "iptables". Et comme les règles de Netfilter sont perdues à chaque arrêt de la machine, il faudra tout recommencer à chaque fois que vous démarrerez votre ordinateur. Hein ? Quoi ? Il est fou ce type ? Nannn, je vous rassure, il y a un moyen simple de d'éviter cette galère. Il s'agit des scripts. Si vous savez déjà ce qu'est un script, passez au paragraphe suivant.

III-6-1 Un script ? Késako ?

Un script est un ensemble de commandes que l'ordinateur va exécuter à votre place. Il s'écrit tout simplement en copiant dans un fichier texte, les commandes que vous auriez tapé. D'ordinaire, un script est un fichier ne comportant pas d'extension. Mais pour les distinguer des fichiers classiques de votre disque, il est parfois utile de leur donner un extension. Contrairement à DOS® / Windows® et leurs fichiers ".bat" ou ".cmd", n'importe quelle extension peut être utilisée. Dans ce qui va suivre, je vais donc utiliser l'extension " .sh".

Voici un exemple de script très très simple :

```
[olivier@phoenix /]$ cat archives/scripts/exemple-01.sh
################################################################################
# NOM : exemple-01.sh
#
# COMMENTAIRE : Simple exemple de script sh/bash
#
# Créé le : 2003/07/01                    Dernière modification le : 2003/07/01
################################################################################

# Les lignes commençant par un "#" sont des commentaires, elles ne sont donc pas
# affichées

echo "+ Bonjour lecteur"
echo "+ Nous somme le ‘date +%Y/%m/%d‘ (aaaa/mm/jj) et il est ‘date +%H:%M:%S‘"
echo "+ Vous aimez les scripts ?"
```

Pour l'exécuter, il suffit de lancer la commande "sh exemple-01.sh", où "sh" est le nom du programme (on parle aussi du terme d'interpréteur), qui va exécuter ce script.

```
[olivier@phoenix /]$ sh archives/scripts/exemple-01.sh
+ Bonjour lecteur
+ Nous somme le 2003/07/01 (aaaa/mm/jj) et il est 15:51:32
+ Vous aimez les scripts ?
```

Mais taper la commande "sh ..." est un peu pénible, non ? Donc nous pouvons changer tout ceci en faisant quelques améliorations à notre script :

- D'abord, rajoutons "#!/bin/sh" puis "#" aux deux premières lignes de notre script, comme dans l'exemple-02.sh.
- Puis, rendons exécutable le script via la commande "chmod +x exemple-02.sh" (lire "man chmod" afin de comprendre en détail comment on rend exécutable un script).
- Enfin, lançons le : ./exemple-02.sh . Notez bien le "./" en début de commande. Ou tapez le chemin absolue ou relatif de ce script ("archives/scripts/exemple-02.sh") dans notre exemple.

```
[olivier@phoenix /]$ cat archives/scripts/exemple-02.sh
#!/bin/sh
#
################################################################################
# NOM : exemple-02.sh
#
# COMMENTAIRE : Simple exemple de script sh/bash
#
# Créé le : 2003/07/01                    Dernière modification le : 2003/07/01
################################################################################

# Les lignes commençant par un "#" sont des commentaires, elles ne sont donc pas
# affichées

echo "+ Bonjour lecteur"
echo "+ Nous somme le ‘date +%Y/%m/%d‘ (aaaa/mm/jj) et il est ‘date +%H:%M:%S‘"
echo "+ Vous aimez les scripts ?"
[olivier@phoenix /]$ chmod +x archives/scripts/exemple-02.sh
[olivier@phoenix /]$ archives/scripts/exemple-02.sh
+ Bonjour lecteur
+ Nous somme le 2003/07/01 (aaaa/mm/jj) et il est 16:19:04
+ Vous aimez les scripts ?
```

Remarquez que, une fois que les bits d'exécution du fichier ont été activés (via la commande "chmod"), vous pouvez fort bien le lancer en double-cliquant dessus via votre gestionnaire de fichiers favoris : Konqueror, Nautilus, ou quoi que soit d'autre. Cependant, bon nombre de scripts rendent la main immédiatement une fois exécuté, et le gestionnaire de fichiers ferme en général le programme tout de suite. Donc vous n'aurez pas le temps de voir ce qui se passe, mis à part une fenêtre noire s'affichant furtivement. Bref, la meilleure solution reste le bon vieux Xterm, la Konsole, etc... Personnellement, j'ai toujours une Konsole d'ouverte, et vous ? 😃

Reprenons notre sérieux : Le fait de changer le bit d'exécution est une opération qui peut être lourde de conséquence, surtout si vous utiliser le compte root pour lancer le script. D'une manière générale, je ne fais pas confiance aux scripts que je trouve sur Internet, à moins :

- que le programme soit fait par quelqu'un de sérieux, ou par une personne que je connaisse.
- ou que j'ai analysé le code, à la recherche de commandes endommageant mon système.

Et de toute façon, je redouble de prudence quand il s'agit d'exécuter un script en temps que root.

Bref, comme par la suite vous allez devoir lancer en temps que root des scripts que vous trouverez sur cette documentation, je vous encourage à vérifier auparavant qu'ils sont bien ce qu'ils sont supposés être. Si cela peut vous rassurer, j'ai lancé moi-même ces scripts sur ma machine en temps que root. Mais si vous ne voulez pas prendre le risque, je comprendrais parfaitement.

Bon, je ne vais pas rentrer plus dans les détails des scripts. Les capacités des scripts sont énormes, car se sont en fait des mini programmes. Vous pouvez même en trouver sur votre Linux qui sont de belle taille (ceux de "/etc/init.d/" sont pas mal) et qui ont énormément de possibilités. J'en propose moi même un dans cette documentation, dont l'explication se trouve dans le chapitre suivant.

Si vous voulez plus d'information sur les scripts, je vous conseille de rechercher sur Internet, ou tout simplement d'utiliser le "man sh" ou "man bash". Cependant, comme tout "man", la manière donc cela est écrit est un peu particulière, voir carrément pénible. Alors refaites votre stock d'aspirines ! Décidément, à force de lire ce document cela va finir par vous coûter cher, et faire la fortune de votre pharmacien ! 😃

III-6-2 Iptables basique

Nous allons commencer par travailler exclusivement sur la table filter.

Bon, petit rappel sur LA règle de filtrage universellement reconnue :

- Premièrement, vider toutes les chaînes de toutes les tables de Netfilter, afin de savoir exactement ce que l'on a dans notre firewall. Mais il ne faudra pas rester trop longtemps dans cette situation, car la machine sera sans aucune protection.
- Deuxièmement, interdire par défaut tous les paquets. Pour cela, nous allons utiliser l'option "-P" ("Politique par défaut) des chaînes INPUT, FORWARD et OUTPUT de la table filter.
- Dans un dernier temps, nous n'allons autoriser que certains flux bien particuliers.
- Concernant l'ordre des règles : Lorsque nous appelons la commande "iptables" pour rajouter/supprimer des règles, l'ordre d'insertion de celles-ci à une certaine importance. Si une première règle supprime un certain type de paquets, une seconde règle écrite un peu plus loin dans votre script ne verra jamais ces paquets. Donc inutile de les accepter, ou de les supprimer de nouveau. Mais en général, comme on écrit uniquement des règles pour accepter des paquets ("-j ACCEPT"), il n'y pas d'importance dans l'ordre des règles.

Facile non ? Bon, gardez en mémoire le tableau des options d'iptables, mettons nous au boulot :

- Suppression de toutes les chaînes : C'est facile, il faut supprimer les tables pré-définies (option " -F') et toutes les tables utilisateurs (option "-X"). Que nous en ayons déjà ou pas écrite n'a aucune importance, on supprime tout :

```
[root@phoenix /]# iptables -t filter -F
[root@phoenix /]# iptables -t filter -X
```

- Définition de la politique (cibles) par défaut : La table filter possède 3 chaînes, donc elles sont toutes les trois à initialiser. Par défaut, on décide donc de tout détruire (DROP) :

```
[root@phoenix /]# iptables -t filter -P INPUT DROP
[root@phoenix /]# iptables -t filter -P OUTPUT DROP
[root@phoenix /]# iptables -t filter -P FORWARD DROP
```

  A ce stade là, vous avez court-circuité tout votre système de réseau. Toutes vos connexions réseaux sont hors service. Pas un logiciel que vous utilisez ne peut accéder au réseau, ou à vous propres serveur. J'en veux pour preuve, la commande "ping localhost" ne marche même plus, et pourtant il lui en faut pour ne plus marcher !

```
[olivier@phoenix /]$ ping localhost
PING localhost.sky.net (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- localhost.sky.net ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2011ms
```

  Hahhh bravo, c'est du propre !

  Mais cela tombe bien, car c'est exactement ce que nous voulions faire. D'un autre coté, c'est la protection absolue du réseau, qui rivalise presque avec la déconnexion physique des câbles... Mais quand à l'intérêt de la chose, c'est plutôt limité !

- Autoriser quelques connexions : Dans notre réseau, nous avons 2 cartes réseaux ("eth0" et "eth1"), ainsi que l'interface de loopback ("lo"). Occupons nous donc de chacune des 3 interfaces :
  - lo (réseau virtuel localhost) : C'est le plus facile.
    Nous pouvons avoir toute confiance en ce réseau, car il est interne à la mémoire de notre machine. Nous allons donc autoriser ("-j ACCEPT") toutes les connexions sortantes des processus locaux ("-A OUTPUT") par cette interface virtuelle ("-o lo") ayant une adresse de loopback ("-s 127.0.0.0/8"), et à destination des machines de ce réseau virtuel (-d "127.0.0.0/8) :

    ```
    [root@phoenix /]# iptables -t filter -A OUTPUT -o lo -s 127.0.0.0/8 -d 127.0.0.0/8 -j ACCEPT
    ```

    Puis nous allons faire l'inverse, c'est à dire autoriser ("-j ACCEPT") toutes les connexions rentrantes dans les processus locaux ("-A INPUT"), par cette interface virtuelle ("-i lo"), venant des machines de ce réseau virtuel (-s "127.0.0.0/8) et à destination des adresses de loopback ("-d 127.0.0.0/8") :

    ```
    [root@phoenix /]# iptables -t filter -A INPUT  -i lo -s 127.0.0.0/8 -d 127.0.0.0/8 -j ACCEPT
    ```

    Et nous pouvons immédiatement vérifier que le "ping localhost" fonctionne à nouveau :

    ```
    [olivier@phoenix /]$ ping localhost
    PING localhost.sky.net (127.0.0.1) 56(84) bytes of data.
    64 bytes from localhost.sky.net (127.0.0.1): icmp_seq=1 ttl=64 time=0.134 ms
    64 bytes from localhost.sky.net (127.0.0.1): icmp_seq=2 ttl=64 time=0.091 ms

    --- localhost.sky.net ping statistics ---
    2 packets transmitted, 2 received, 0% packet loss, time 999ms
    rtt min/avg/max/mdev = 0.091/0.112/0.134/0.023 ms
    ```

    Ouuuuiiiii ! Nous sommes très fort ! Bon, une interface de réglée passons aux autres.

  - eth0 (réseau interne "sky.net") : Cela reste encore facile. Nous désirons dans un premier temps permettre à paradise.sky.net de dialoguer sans limite avec phoenix0.sky.net. Quoi ? Sans limite ? Mais ce n'est pas un peu dangereux, non ? Certes cela peu se discuter, mais comme à priori nous travaillons dans un réseau local et qui plus est dans le réseau local d'un particulier, nous pouvons avoir une certaine confiance dans les machines de notre propre réseau, non ? Ah, vous utilisez Windows® sur les autres machines de votre réseau ? Et vous avez peur que ce Windows® agresse vos Linux ? Chacun son problème mon (ma) cher (chère) ! En allant un peu plus loin dans ce document, vous comprendrez comment on peut mettre un filtrage un peu plus restrictif, donc n'anticipons pas.

    Donc nous avons fait comme pour l'interface "lo", et autoriser les processus locaux à dialoguer en entrée et en sortie avec le réseau local :

    ```
    [root@phoenix /]# iptables -t filter -A OUTPUT -o eth0 -s 192.168.0.0/24 -d 192.168.0.0/24 -j ACCEPT
    [root@phoenix /]# iptables -t filter -A INPUT  -i eth0 -s 192.168.0.0/24 -d 192.168.0.0/24 -j ACCEPT
    ```

    Bien, et que donne le "ping paradise.sky.net" ?

    ```
    [olivier@phoenix /]$ ping paradise.sky.net
    PING paradise.sky.net (192.168.0.2) 56(84) bytes of data.
    ```

```
64 bytes from paradise.sky.net (192.168.0.2): icmp_seq=1 ttl=64 time=0.134 ms
64 bytes from paradise.sky.net (192.168.0.2): icmp_seq=2 ttl=64 time=0.091 ms

--- localhost.sky.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.091/0.112/0.134/0.023 ms
```

Très bien ! 2 de passées, reste la 3ème...

- eth1 (réseau Internet "internet.net") : évidement, c'est la dernière, et donc la plus compliquée...
  Que vous nous faire exactement en fait ? Interdire l'accès à pirate.internet.net, mais autoriser celui de web.internet.net. C'est facile alors, il suffit de mettre uniquement des règles "-j ACCEPT" en direction et depuis l'adresse IP de web.internet.net ? En théorie oui... Mais est-ce que vous connaissez l'adresse IP de tous les intrus qui veulent vous ennuyer ? Non hein ? Ce n'est pas le genre d'informations qui se trouve sous le sabot d'un cheval ! Et puis en plus, quelqu'un sur web.internet.net pourrait héberger sans que vous le sachiez un autre intrus... Que la vie est compliquée, non ?

  En fait, c'est très simple : nous n'allons autoriser que les connexions vers les services web qui nous intéresse, à savoir le HTTP (port 80 en TCP), et le HTTPS (443 en TCP), et cela pour toutes les machines. Seul les requêtes en direction et venant des ports seront autorisées. Hop, 4 règles d'"iptables" est c'est fait :

```
[root@phoenix /]# iptables -t filter -A OUTPUT -o eth1 -s 10.0.0.0/8 -p tcp --dport 80  -j ACCEPT
[root@phoenix /]# iptables -t filter -A OUTPUT -o eth1 -s 10.0.0.0/8 -p tcp --dport 443 -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i eth1 -d 10.0.0.0/8 -p tcp --sport 80  -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i eth1 -d 10.0.0.0/8 -p tcp --sport 443 -j ACCEPT
```

  Il ne nous reste plus qu'à faire ouvrir une page web sur web.internet.net, ou de faire un petit "nmap" sur ses ports 80 et 443, afin de voir si tout marche bien.

```
[olivier@phoenix /]$ nmap web.internet.net -p 80,443

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on web.0.0.10.in-addr.arpa (10.0.0.200):
Port       State       Service
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

  Bingo ! Ca marche ! Faites péter les biscuits apéros ! Ce soir, c'est fête ! Et en plus, ce premier script "iptables" est téléchargeable ici .

Nannnnn désolé de vous avoir donné un faux espoir, mais en fait il y a un truc important qui ne marche pas...

Depuis Phoenix, essayez de faire un "ping phoenix.sky.net", un "ping phoenix0.sky.net", ou un "ping phoenix1.sky.net" :

```
[olivier@phoenix /]$ ping phoenix.sky.net
PING phoenix.sky.net (192.168.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- phoenix.sky.net ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1010ms

[olivier@phoenix /]$ ping phoenix0.sky.net
PING phoenix0.sky.net (192.168.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- phoenix0.sky.net ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1019ms

[olivier@phoenix /]$ ping phoenix1.internet.net
PING phoenix1.internet.net (10.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted

--- phoenix1.internet.net ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Pourquoi cela ne marche t'il pas ? Aurions nous oublié quelque chose ? De toute évidence oui. Souvenez vous (recherchez le lien "plus tard") lors du 1er chapitre, nous en avions rapidement parlé. Que vous ayez sauté ce chapitre ou que vous somnoliez à ce moment n'a pas d'importance .

En fait, c'est très simple : dans nos règles "iptables" de configuration de localhost, nous avons indiqué que nous autorisions uniquement les échanges sur l'interface localhost, pour les machines du réseau 127.0.0.0/8. Hors, et c'est ce que nous avions vu lors du 1er chapitre, lorsque qu'une machine accède à ses propres ressources elle ne passent pas par l'interface physique, mais uniquement par l'interface loopback. Donc au lieu de passer par les règles de l'interface eth0, ces paquets sont passés par ceux de l'interface lo, et se sont fait détruire par la règle DROP par défaut... C'est très fort, non ?

Comment résoudre ce problème alors ? La solution est très simple, il suffit d'être un peu plus "tolérant" sur les 2 règles de l'interface "localhost", et retirer les options "-d 127.0.0.0/8" et "-s 127.0.0.0/8". Pour cela, nous allons supprimer les précédentes règles, et les recréer :

```
[root@phoenix /]# iptables -t filter -L -n -v
Chain INPUT (policy DROP 463 packets, 35458 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       127.0.0.0/8          127.0.0.0/8 <-- Ceci est la règle 1
    0     0 ACCEPT     all  --  eth0   *       192.168.0.0/24       0.0.0.0/0
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp spt:80
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp spt:443

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

Chain OUTPUT (policy DROP 71 packets, 10712 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      lo      127.0.0.0/8          127.0.0.0/8 <-- Ceci est la règle 1
    0     0 ACCEPT     all  --  *      eth0    0.0.0.0/0            192.168.0.0/24
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           tcp dpt:443
[root@phoenix /]# iptables -t filter -D OUTPUT 1
```

```
[root@phoenix /]# iptables -t filter -D INPUT  1
[root@phoenix /]# iptables -t filter -A OUTPUT -o lo -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i lo -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
[root@phoenix /]# iptables -t filter -L -n -v
Chain INPUT (policy DROP 475 packets, 37048 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1   248 ACCEPT     all  --  eth0   *       192.168.0.0/24       0.0.0.0/0
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp spt:80
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp spt:443
  274  166K ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0

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

Chain OUTPUT (policy DROP 84 packets, 12080 bytes)
 pkts bytes target     prot opt in     out     source               destination
    9   872 ACCEPT     all  --  *      eth0    0.0.0.0/0            192.168.0.0/24
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           tcp dpt:443
  274  166K ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
```

Cette fois ci, un "ping phoenix0.sky.net" et des autres adresses IP de Phoenix marche sans problème. Ce second script corrigé est téléchargeable ici .

Vous vous demandez sûrement pourquoi j'ai pris un (malin ?) plaisir à introduire cette "erreur" dans mon premier script ? C'est simple :

- Il sera très important par la suite de comprendre cette histoire d'auto connexion.
- Je ne suis pas sûr que vous vous en seriez souvenu si je ne vous avais pas fait pointer le doigt dessus.
- Au passage, nous avons vu une utilisation de deux commandes que nous ne connaissions pas : "iptables -t filter -L -n -v" pour afficher la liste des chaînes d'une table, et "iptables -t filter -D ...." qui permet de supprimer une règle. Comme quoi, les erreurs sont parfois constructives !

Bon, pendant que vous prenez un biscuit apéro bien mérité, je ferais quelques remarques sur ces scripts :

- Dans toutes nos commandes, nous avons utilisé l'option "-t filter", pour indiquer que nous voulions travailler sur cette table. C'est très bien, mais c'est un peu inutile. En effet, par défaut "iptables" considère que c'est la table "filter" qui est utilisée. Donc par la suite, on pourra économiser de la ligne de commande en n'utilisant plus le "-t filter" pour ces commandes iptables là.
- Dans nos commandes iptables ayant pour cible "ACCEPT", nous avons cherché à rajouter le maximum de paramètre, en combinant par exemple les options "-i" avec "-s". C'est une bonne chose, car les règles d'acceptation des paquets doivent toujours être les plus restrictives possible. Ne rentre pas sur notre ordinateur qui veut ! Même si au passage, un peu trop de rigueur amène à des erreurs du type du "localhost"...
- Même si nos règles sur le réseau " internet.net" sont satisfaisantes, car bien strictes, elle sont un peu lourdes à gérer. 4 règles "iptables" uniquement pour autoriser la visite de sites web, sécurisés ou non, cela fait beaucoup... D'autant que nous n'avons pas parlé de FTP, de IRC, de "chat" ("discussion" en français), et que sais je encore.
- Enfin, au point où nous en somme, nous avons l'équivalent d'un petit firewall de type matériel, que l'on peut trouver dans le commerce, voir même équivalent au firewall intégré dans Windows® XP (car au jour où j'écris ces lignes, il ne gère pas le conntrack).

Cependant, nous pouvons être satisfait de notre travail. Sans protection de la part de netfilter, l'intrus pouvait voir tout nos ports ouverts :

```
[intrus@pirate /]$ nmap phoenix1.internet.net

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on phoenix1.internet.net (10.0.0.1):
(The 1590 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
110/tcp    open        pop-3
111/tcp    open        sunrpc
139/tcp    open        netbios-ssn
443/tcp    open        https
3306/tcp   open        mysql
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Maintenant, avec notre second script netfilter, non seulement il ne voit plus rien, mais en plus il suppose que Phoenix est arrêté :

```
[intrus@pirate /]# nmap phoenix1.internet.net

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds
```

Alors que nous, nous pouvons surfer en toute tranquillité. He he he, nous avons berné l'intrus ! La vie n'est elle pas belle ? Hummmm, si si...

Ahhh, que c'est beau l'insouciance de ces jeunes internautes... Mais à votre avis, pourquoi il y a encore quelques paragraphes après celui-ci ? Humm ? Je vous le donne en mille : c'est à cause du retour de l'intrus masqué ! Il revient, et il n'est pas du tout content que vous ayez essayé de l'empêcher de rentrer sur votre machine... En fait, nous avons perdu cette bataille face à lui, et nous ne le savons même pas encore...

Bon, prenez fissa de quoi vous remonter, et préparez vous à vous remettre une nouvelle couche d'aspirines...

III-7 Le suivi de connexion (conntrack)

III-7-1 Comment leurrer un firewall en une leçon...

Phoenix se trouve toujours dans la configuration Netfilter définie par le script "iptables-basic-2.sh", et Pirate ne nous voit toujours pas en utilisant la commande "nmap". Oui, mais notre intrus n'est pas un petit plaisantin. Comme nous, il sait lire le "man nmap", et il connaît l'option "-g"...

Notre intrus va donc se connecter en temps que root sur sa propre machine, et lancer la commande "nmap phoenix1.internet.net -g 80" :

```
[root@pirate /]# nmap phoenix1.internet.net -g 80

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
```

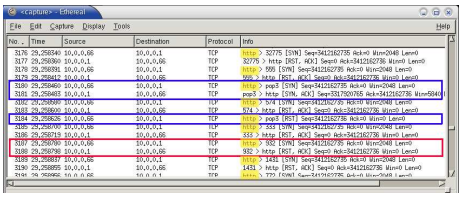
```
Interesting ports on phoenix1.internet.net (10.0.0.1):
(The 1585 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
110/tcp    open        pop-3
111/tcp    open        sunrpc
139/tcp    open        netbios-ssn
307/tcp    filtered    unknown
404/tcp    filtered    nced
443/tcp    open        https
511/tcp    filtered    passgo
578/tcp    filtered    ipdd
607/tcp    filtered    nqs
3306/tcp   open        mysql
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Horreur sans nom ! Contrairement à la précédente commande "nmap" , non seulement notre intrus a réussi à nous retrouver, mais en plus il voit tout nos ports ouverts ! Argggggg, tout ce travail pour rien ! Netfilter ne serait il qu'un gadget inutile ?

Non, en fait c'est tout à fait naturel et cela prouve la protection complètement insuffisante d'un firewall basé uniquement sur l'ouverture des ports. L'image ci-contre le montre bien. C'est une capture des connexions réseaux faite par " Ethereal" sur phoenix1.internet.net. On y voit :

- En rouge : Pirate ouvre une connexion venant du port 80, et à destination d'un port de Phoenix sur lequel il suppose que tourne un démon. Il commence une " handshake" tout à fait classique (SYN), auquel répond Phoenix par un "RESET" (RST, ACK), disant que ce port (932) n'est pas ouvert.
- En bleu : là encore, Pirate ouvre une connexion venant du port 80. Mais cette fois ci, Phoenix l'informe que le port ("pop3" <=> 110) est ouvert. Poliment, Pirate reforme la connexion (RST) afin de ne pas donner l'éveil à Phoenix. Mais au passage, il a pu noter que le port en question était ouvert... Victoire sur tout la ligne pour l'intrus...



Analyse d'un 'nmap -g 80'

Comme on le voit, l'utilisation du paramètre " -g 80" permet de configurer "nmap" en lui disant de se faire passer pour un serveur web ("HTTP" <=> 80). Cette option n'est utilisable que parce que l'intrus a lancé la commande en temps que root. Mais qu'importe, l'intrus est forcément root sur sa machine. Et dans notre configuration de Netfilter, comme nous avons explicitement autorisé les connexions rentrantes venant du port 80, Netfilter ne peut pas interdire ce port scanning... Quant à interdire les connexions venant du port 80, c'est tout bonnement impossible, car sinon nous ne pourrions plus accéder à des sites web...

On peut se demander si un tel port scanning a un réel intérêt. Bon, l'intrus sait que nos ports sont ouverts. Oui, et alors ? La belle affaire ! Notre Netfilter est là pour nous protéger, non ? Non, malheureusement pas avec une configuration si pitoyables...

Bon, notre intrus n'est pas née de la dernière pluie, et il connaît aussi le programme "nc" ("nc - TCP/IP swiss army knife", "man nc" pour plus d'informations). Avec ce programme, il va pouvoir ouvrir une connexion TCP/IP depuis son port 80, vers le port 80 de Phoenix, et envoyer toutes les commandes qu'il veut. Comme part exemple un "GET /index.html" qui permet de télécharger la page "/index.html". Cette commande est la base du protocole HTTP, et votre navigateur le fait pour ainsi dire tout le temps.

```
[root@pirate /]# nc -p 80 phoenix1.internet.net 80
GET /
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.61 [en] (X11; I; Linux 2.2.9-23mdk i686) [Netscape]">
   <meta name="Author" content="MandrakeSoft">
   <title>Welcome to the Advanced Extranet Server, ADVX!</title>
```

La fin de la réponse a été coupée, car inutile. On voit bien ici la réponse du serveur web de Phoenix, qui affiche tout naturellement une page web de notre réseau local... Que le lecteur continue de s'inquiéter, ce qui est fait ici très simplement avec un serveur web, peut l'être tout autant avec un serveur un peu plus sensible, comme par exemple le partage de fichier "Samba" ou "NFS". Il ne faut pas grand chose en fait, juste un peu modifier les sources de programmes comme "smbmount" ou "mount"...

Bon, si à ce stade je n'ai pas réussi à capter toute votre attention sur ce problème, vous pouvez définitivement arrêter la lecture de ce document, et continuer de vous faire allégrement pirater ! 😀

III-7-2 ... Et comment renvoyer l'intrus dans sa niche

Toujours là ? Parfais, passons à la solution de ce problème :

Bien, le problème ici est que notre intrus peut rentrer comme il veut par les différents ports que nous aurions laissé ouverts afin que nous, nous puissions aller sur Internet. En fait, il faudrait que notre firewall soit semi-perméable, c'est à dire qu'il ne laisse entrer des connexions venant du port 80, que si elles viennent en réponse de connexions qui auraient été initialisées par Phoenix. C'est justement ce que permet le module de suivi de connexion (conntrack) de Netfilter.

Cette technique est vraiment une avancée en matière de firewall. Netfilter se base sur la poignée de main ("handshake", que nous avons vu au 1er chapitre), afin de déterminer si une connexion a été initialisée par Phoenix ou non. Pour toute nouvelle connexion sortante, il associe l'état "NEW" ("nouveau" en anglais) à la connexion, et stocke cette information en mémoire. Quand il verra arriver d'autres connexions venant de l'extérieur en réponse à cette connexion "NEW", il leur attribuera alors l'état "ESTABLISHED" ("établie" en anglais). Ainsi, pour une connexion rentrante et qui ne se trouve pas déjà dans la mémoire de Netfilter, celui-ci pourra en déduire qu'elle a été initialisée par l'extérieur, et lui attribuera l'état " INVALID". Dans ce cas, et si il a été configuré pour, Netfilter pourra refuser ("DROP") ou rejeter ("REJECT") cette connexion.

Il existe un dernier statut intéressant pour le suivi de connexion, il s'agit du statut "RELATED" : Il existe certains protocoles, comme le FTP ou l'IRC par exemple, qui ne répondent pas toujours sur le port qui a initialisé la connexion. Au lieu de cela, ils initialisent eux-mêmes une connexion sur un autre port de la machine demandant l'information. Vous pouvez trouver une excellente explication sur le fonctionnement de ce mode particulier du protocole FTP sur ce site (la partie sur le "mode passif"). Pour certains de ces protocoles particuliers, Netfilter est capable de comprendre que cette nouvelle connexion rentrante est en fait en relation avec une autre connexion précédemment établie par la machine elle-même. Pour ce type de connexion, elle leur attribue le statut de "RELATED" (pour "relative" en anglais).

Voyons maintenant comment mettre en oeuvre cette technique. Le système de suivi de connexion de Netfilter n'est pas forcément compilé dans le kernel, donc si ce n'est pas le cas, vous devez d'abord charger le module "ip_conntrack" :

```
[root@phoenix /]# modprobe ip_conntrack
```

Si vous envisagez d'utiliser vos clients FTP en mode passif, ou que vous compter utiliser l'IRC, vous avez la possibilité de charger les modules "ip_conntrack" qui supportent ces protocoles :

```
[root@phoenix /]# modprobe ip_conntrack_ftp
[root@phoenix /]# modprobe ip_conntrack_irc
```

Enfin, pour voir la listes des modules "ip_conntrack" qui sont installés sur votre machine :

```
[root@phoenix /]# uname -a
Linux phoenix.sky.net 2.4.21-0.13mdksmp #1 SMP Fri Mar 14 13:41:18 EST 2003 i686 unknown unknown GNU/Linux
[root@phoenix /]# find /lib/modules/2.4.21-0.13mdksmp/ -iname "ip_conntrack_*"
/lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ip_conntrack_ftp.o.gz
/lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ip_conntrack_proto_gre.o.gz
/lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ip_conntrack_h323.o.gz
/lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ip_conntrack_irc.o.gz
/lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ip_conntrack_pptp.o.gz
```

Maintenant, comment indiquer à Netfilter que nous ne désirons laisser passer que les connexions sortantes initialisées par la machine, mais aussi de ne laisser passer que celles qui sont en réponse avec les premières ? C'est tout simplement aussi facile que ce que nous venons de dire :

```
[root@phoenix /]# iptables -A OUTPUT -o eth1 -s 10.0.0.1  -d 0.0.0.0/0
                           -p all -m state --state ! INVALID           -j ACCEPT
[root@phoenix /]# iptables -A INPUT  -i eth1 -s 0.0.0.0/0 -d 10.0.0.1
                           -p all -m state --state RELATED,ESTABLISHED -j ACCEPT
```

- Le paramètre "-m state" signifie que nous avons besoin du module "state" (celui du suivi de connexion), et de ses options particulières.
- "--state" indique les états qui vont être utilisés pour nos règles.
- "! INVALID" : comme vu plus haut, le module de suivi de connexion gère 4 états, NEW, ESTABLISHED, RELATED et INVALID. Ce paramètre indique que l'on accepte uniquement les connexions qui ne sont pas invalides, donc qui sont "NEW, ESTABLISHED, RELATED". Ceci est uniquement un manière plus rapide d'écrire qu'il faut que les connexions soient d'un des 3 états "NEW, ESTABLISHED, RELATED".

Voila, nous venons de terminer notre configuration de Netfilter en utilisant le suivi de connexion. Le script iptables de tout ceci se trouve ici.

Mais est-ce vraiment efficace ? On va le voir tout de suite :

```
[root@pirate /]# nmap phoenix1.internet.net -g 80 -P0

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
All 1601 scanned ports on phoenix1.internet.net (10.0.0.1) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 1721 seconds
```

On utilise l'option -P0 pour forcer la connexion par "nmap", vu que Phoenix refuse de répondre aux demandes les plus simples. On remarque que le scan est excessivement long (1721 secondes, soit près d'une demi heure...), car Pirate fait son scan en testant une dizaine de ports en parallèle, mais attends une dizaine de secondes pour chaque groupe de ports testés.
Dans ces conditions, Nmap ne peut déterminer si Phoenix est joignable ou pas.

Sur Phoenix on peut voir par contre qu'un grand nombre de paquets entrants ont été "dropés". Ce sont les tentatives de connexions de "nmap" :

```
[root@phoenix /]# iptables -L -v -n
Chain INPUT (policy DROP 6571 packets, 268K bytes)
 pkts bytes target     prot opt in     out  source         destination
 3480 2235K ACCEPT     all  --  lo     *    0.0.0.0/0      0.0.0.0/0
    9  2232 ACCEPT     all  --  eth0   *    192.168.0.0/24 192.168.0.0/24
    0     0 ACCEPT     all  --  eth1   *    0.0.0.0/0      10.0.0.1       state RELATED,ESTABLISHED

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

Chain OUTPUT (policy DROP 22 packets, 1716 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
```

```
 3480 2235K ACCEPT     all  --  *      lo   0.0.0.0/0      0.0.0.0/0
    9  2232 ACCEPT     all  --  *      eth0 192.168.0.0/24 192.168.0.0/24
   97  9096 ACCEPT     all  --  *      eth1 10.0.0.1       0.0.0.0/0      state NEW,RELATED,ESTABLISHED
```

La conclusion est sans appel : notre machine est un "trou noir" qui se refuse de répondre aux sollicitations extérieures . Par contre, Phoenix arrive toujours à se connecter sur internet.net :

```
[olivier@phoenix /]$ ping web.internet.net
PING web.internet.net (10.0.0.200) 56(84) bytes of data.
64 bytes from web.0.0.10.in-addr.arpa (10.0.0.200): icmp_seq=1 ttl=64 time=2.22 ms
64 bytes from web.0.0.10.in-addr.arpa (10.0.0.200): icmp_seq=2 ttl=64 time=0.300 ms

--- web.internet.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.300/1.263/2.227/0.964 ms
[olivier@phoenix /]$ nmap web.internet.net -p 80,443

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on web.0.0.10.in-addr.arpa (10.0.0.200):
Port       State       Service
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Victoire ? Cette fois ci, OUI !!! L'intrus ne peut nullement accéder de lui-même à notre machine, toutes ses tentatives de "nmap" échoues. Votre machine a donc été en grande partie sécurisée. Comparé à la situation au début de la lecture de ce document, vous pouvez donc être un peu plus tranquille, et vous avez un peu moins à craindre des intrus. Mais, et comme nous le verrons en conclusion de ce chapitre, ce n'est pas pour autant que vous ne risquez rien.

Bien, maintenant que nous avons sécurisé l'accès à notre machine grâce à la table " Filter", passons aux autres fonctionnalités intéressantes de Netfilter : passons à l'exploitation de la table "NAT".

III-8 IP masquerading / Port forwarding

Si Netfilter est très efficace pour filtrer les connexions entrantes et sortantes des processus locaux, il peut servir aussi à d'autres fonctionnalités, comme le masquage IP et le suivi de port.

III-8-1 IP masquerading

Jusqu'à présent, la machine Paradise de notre réseau n'a pas été très utilisée. Son activité réseau est restreinte à celle du réseau sky.net. Alors que Phoenix, lui, peut se connecter au réseau internet.net. Tout cela n'est pas très juste, aussi allons nous y remédier. Comment ? En transformant tout simplement Phoenix en une passerelle entre les réseaux sky.net et internet.net.

Dans tout ce qui suit, nous allons nous arranger pour que paradise.sky.net puisse se connecter au serveur web web.internet.net.

Sous Windows®, ce que nous allons faire s'appelle du "partage de connexion Internet", ce qui est tout à fait vrai : Phoenix va partager sa connexion à Internet avec les machines du réseaux sky.net.

Pour réaliser ceci, il nous faut réaliser plusieurs opérations :

- La première, c'est de charger les modules dont nous allons avoir besoin. En premier, nous avons besoin du module de NAT, c'est à dire "iptables_nat". Comme nous voulons aussi faire du suivi de connexion sur les paquets NAT, nous chargerons de même les modules NAT FTP et IRC, "ip_nat_ftp" et "ip_nat_irc" :

  ```
  [root@phoenix /]# modprobe iptable_nat
  [root@phoenix /]# modprobe ip_nat_ftp
  [root@phoenix /]# modprobe ip_nat_irc
  ```

- Bien entendu, tout comme nous initialisons la table "Filter" nous devons initialiser la tables NAT :

  ```
  [root@phoenix /]# iptables -t nat -F
  [root@phoenix /]# iptables -t nat -X
  ```

- Pour ce qui est des cibles par défaut des chaînes de la table NAT, nous acceptons toutes les connexions. Il n'est pas nécessaire de faire pointer ces cibles sur "DROP", car la sécurité est établie au niveau de la table "Filter", par le "DROP" par défaut de la chaîne FORWARD :

  ```
  [root@phoenix /]# iptables -t filter -P FORWARD     DROP
  [root@phoenix /]# iptables -t nat    -P PREROUTING  ACCEPT
  [root@phoenix /]# iptables -t nat    -P POSTROUTING ACCEPT
  [root@phoenix /]# iptables -t nat    -P OUTPUT      ACCEPT
  ```

- Bien, maintenant nous allons faire suivre sur le réseau internet.net, les connexions issues du réseau sky.net. Bien entendu, nous ne ferons suivre que les connexions qui sont à destination du réseau internet.net, et non celles destinées à la machine Phoenix elle-même. Pour cela, nous allons avoir besoin d'utiliser la table "Filter" (he oui, encore !), afin de faire suivre les paquets venant de la carte eth0 (phoenix0.sky.net) à la carte eth1 (phoenix1.internet.net), et vice versa. En plus de cela, comme ce sont uniquement les connexions initialisées par le réseau interne que nous désirons faire sortir, nous rajouterons un peu de suivi de connexion.

  ```
  [root@phoenix ]# iptables -t filter -A FORWARD -i eth0 -o eth1 -s 192.168.0.0/24 -d 0.0.0.0/0 \
                           -m state --state ! INVALID           -j ACCEPT
  [root@phoenix ]# iptables -t filter -A FORWARD -i eth1 -o eth0 -s 0.0.0.0/0 -d 192.168.0.0/24 \
                           -m state --state ESTABLISHED,RELATED -j ACCEPT
  ```

  L'accumulation des paramètres "-i", "-o", "-s", "-d" et "--state" permettent de garantir que seul les connexions initialisées par le domaine sky.net seront autorisées à passer, et non l'inverse.

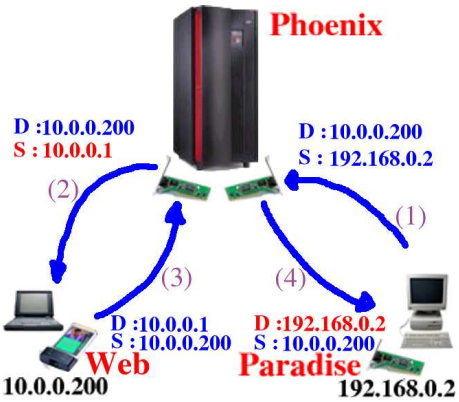
- Au point où nous en somme, les paquets venant de sky.net et sortant par l'interface eth1, ont :
  - pour adresse de destination, une adresse du réseau internet.net, ce qui est parfaitement normale. Celle de web.internet.net par exemple
  - pour adresse source, l'adresse d'une machine du domaine sky.net, comme par exemple celle de paradise.sky.net (voir lien "(1)"). Oui, mais ce domaine est un réseau privé, dont l'adresse IP n'est pas du tout routable sur Internet. Conclusion : lorsque web.internet.net recevra la requête, il ne saura absolument pas où envoyer la réponse... C'est assez ennuyeux, non ?

  Il va donc falloir que Phoenix subtilise l'adresse IP source des trames qui le traverse, et la remplace par la sienne (voir lien "(2)"). C'est là qu'intervient (enfin !) la table NAT, avec la cible "MASQUERADE" (pour "masquage" en anglais) :

  ```
  [root@phoenix /]# iptables -t nat -A POSTROUTING -o eth1 \
                             -s 192.168.0.0/24 -j MASQUERADE
  ```

  Vous pouvez cependant vous poser la question suivante : que deviennent les paquets revenant du réseau internet.net sur l'interface eth1 ? Ils ont pour adresse source web.internet.net, et pour adresse cible phoenix1.internet.net (voir lien "(3)"). Donc si ils sont recopiés sur le réseau sky.net, ils ne saurons pas qu'ils doivent aller sur paradise.sky.net, non ? En théorie, ceci est exacte, et demanderait la mise en place d'une seconde règle de de "MASQUERADE". Mais en fait, c'est inutile, car avec la première règle, Netfilter gère une table en mémoire qui suit ces masquage d'adresses IP, et elle réassigne aux connexions rentrantes la bonnes adresse IP cibles (voir lien "(4)").

  En fait, ce serait même terriblement dangereux de mettre en place une telle règle de ce type. En effet, cela pourrait faire passer pour des connexions de Phoenix, des connexions initialisées par les machines de internet.net ! C'est comme si vous poussiez vous-même le loup dans la bergerie ! Donc l'utilisation de la table NAT et de la cible "MASQUERADE" doivent se faire avec le plus rigueur possible, souvenez vous en bien.



- Enfin, maintenant que tout le NAT est configuré, il ne reste plus qu'à autoriser dûment votre Linux à faire jouer son rôle de gateway. Pour cela, il faut utiliser simplement écrire un "1" dans le "/proc/sys/net/ipv4/ip_forward". Ce fichier est en fait un fichier virtuel, qui permet de dialoguer directement avec le kernel. Nous allons donc utiliser la commande suivante :

  ```
  [root@phoenix /]# echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

Le NAT étant maintenant configuré et activé, il reste à configurer les machines du réseau sky.net, afin de leur indiquer quelle est la passerelle. C'est chose faite en lançant par exemple sur paradise.sky.net :

```
[root@paradise /]# route add default gw phoenix0.sky.net
[root@paradise /]# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         phoenix0.sky.ne 0.0.0.0         UG    0      0        0 eth0
```

Bien, tout est en place. Voyons ce que cela donne. Depuis paradise.sky.net, tentons une connexion sur web.internet.net:

```
[olivier@paradise /]$ nmap web.internet.net -p 80,443

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on web.0.0.10.in-addr.arpa (10.0.0.200):
Port       State       Service
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Bien ! Comme on le voit, la connexion se fait d'un réseau à l'autre, à travers Phoenix. Opération réussie donc ! Cela qui nous donne l'occasion de trouver ici le script iptables de cet exemple.

Remarques : Dans tout nos exemples, nous avons systématiquement autorisé tout le réseau sky.net à accéder à internet.net. Cependant, nous aurions très bien pu limiter cet accès à uniquement quelques machines de notre réseau interne. Pour cela, il suffit de changer les options "-s" et "-d" de nos règles de "FORWARD", et d'indiquer l'adresse IP de la machine qui est autorisée à passer d'un réseau à un autre.
Exemple :

```
[root@phoenix ]# iptables -t filter -A FORWARD -i eth0 -o eth1 -s 192.168.0.2 -d 0.0.0.0/0 \
                          -m state --state ! INVALID           -j ACCEPT
[root@phoenix ]# iptables -t filter -A FORWARD -i eth1 -o eth0 -s 0.0.0.0/0 -d 192.168.0.2 \
                          -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Et qu'en dit la sécurité ? Est-ce que par hasard pirate.internet.net peut accéder à paradise.sky.net ? Est-ce possible ? Techniquement oui, il suffit que pirate.internet.net déclare phoenix1.internet.net comme étant sa passerelle pour le réseau internet.net :

```
[root@pirate /]# route add default gw phoenix1.internet.net
[root@pirate /]# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
10.0.0.0        *               255.0.0.0       U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         phoenix1.intern 0.0.0.0         UG    0      0        0 eth0
```

Puis, de lancer par exemple un "ping" sur l'adresse IP de paradise.sky.net :

```
[root@pirate /]# ping -c 3 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.

--- 192.168.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2014ms
```

Bien, notre machine est correctement sécurisée. Tout va donc pour le mieux ? Indéniablement, oui. Mais tel un funambule entrain de traverser le vide

sur une corde tendue, il ne faut pas grand chose pour que cette état idéal se transforme en un cauchemar sans nom...

En effet, regardons les traces qu'on laissé la dernière connexion de pirate.internet.net dans les statistiques de Netfilter :

```
[root@phoenix /]# iptables -L -v -n -t nat
Chain PREROUTING (policy ACCEPT 3 packets, 252 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      eth1    192.168.0.0/24       0.0.0.0/0

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

Cette commande nous indique que 3 paquets ont été acceptés par le comportement par défaut de la chaîne "PREROUTING" de la table "NAT". C'est tout à fait normal, ce sont nos 3 paquets de ping qui arrivent de pirate.internet.net et qui s'apprêtent à passer à travers Phoenix. Mais qui va les arrêter ?

```
[root@phoenix /]# iptables -L -v -n -t filter
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     all  --  lo     *    0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  eth0   *    192.168.0.0/24 192.168.0.0/24

Chain FORWARD (policy DROP 3 packets, 252 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     all  --  eth0   eth1 192.168.0.0/24 0.0.0.0/0      state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  eth1   eth0 0.0.0.0/0      192.168.0.0/24 state RELATED,ESTABLISHED

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     all  --  *      lo   0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  *      eth0 192.168.0.0/24 192.168.0.0/24
```

Cette commande là nous indique que 3 paquets ont été détruits par le comportement par défaut de la chaîne "FORWARD" de la table "Filter". Là encore, c'est ce à quoi nous attendions. Pourquoi ? Parce qu'aucune des autres règles de la chaîne "FORWARD" de cette table ne les ont laissé passer.

Maintenant, changeons seulement le comportement par défaut de la table "FORWARD" de la table "Filter" :

```
[root@phoenix /]# iptables -t filter -P FORWARD ACCEPT
```

Juste pour information, on pourra télécharger ce méchant script iptables ici.

Et relançons notre "ping" depuis pirate.internet.net :

```
[intrus@pirate /]$ ping -c 3 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=254 time=0.957 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=254 time=0.987 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=254 time=0.957 ms

--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
rtt min/avg/max/mdev = 0.957/0.967/0.987/0.014 ms
```

QUOI !?!? Notre intrus peut accéder sans problème à l'intérieur de notre réseau ? Diantre, il y a un bug, et un méchant ! Et tout cela à cause d'un malheureuse règle par défaut ?

Oui, exactement. Regardons les statistiques d'iptables pour la table "Filter" :

```
[root@phoenix /]# iptables -L -v -n -t filter
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp dpt:6000
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp dpt:6000
    0     0 ACCEPT     all  --  lo     *    0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  eth0   *    192.168.0.0/24 192.168.0.0/24

Chain FORWARD (policy ACCEPT 3 packets, 252 bytes)
 pkts bytes target     prot opt in     out  source         destination
    3   252 ACCEPT     all  --  eth0   eth1 192.168.0.0/24 0.0.0.0/0      state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  eth1   eth0 0.0.0.0/0      192.168.0.0/24 state RELATED,ESTABLISHED

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
    0     0 ACCEPT     all  --  *      lo   0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  *      eth0 192.168.0.0/24 192.168.0.0/24
```

Voilà quelque chose de très intéressant :

- 3 paquets sont passés par la cible "ACCEPT" par défaut de la chaîne "Filter" . Se sont les 3 pings venant de pirate.internet.netà destination de paradise.sky.net. Notre trou de sécurité se trouve indéniablement ici.
- 3 autres paquets ont utilisés la règle basée sur du suivi de connexion , partant du réseau sky.net et allant sur internet.net. Bien sûr, c'est tout à fait normal. Car pour Netfilter, une connexion sky.net -> internet.net est bien initialisée ou en relation avec une autre...

Regardons maintenant les statistiques de la table "NAT" :

```
[root@phoenix /]# iptables -L -v -n -t nat
Chain PREROUTING (policy ACCEPT 3 packets, 252 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 3 packets, 252 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
    0     0 MASQUERADE  all  --  *      eth1    192.168.0.0/24       0.0.0.0/0

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

3 paquets sont passés par les chaînes "PREROUTING" puis "POSTROUTING" afin de sortir du réseau internet.net et atteindre le réseau sky.net. Par contre, il n'y a pas de traces des paquets de réponse de paradise.sky.net à pirate.internet.net. Ceci est dû au mécanisme interne de Netfilter qui identifie les connexions en réponse, et qui ne les fait pas passer par la table NAT. C'est pour cela, et comme vu plus haut, qu'il n'y a pas besoin d'écrire des règles spécifiques pour le "MASQUERADING rentrant".

De cette (longue) explication nous pourrons en tirer 3 leçons :

- Il faut toujours initialiser les cibles par défaut , même pour les chaînes des tables que nous ne pensons peut-être pas utiliser. Il faut en toute situation connaître l'état de de ces cibles. Un script Netfilter devrait donc toujours commencer par :

```
# Initialisation de la table FILTER
iptables -t filter -F
iptables -t filter -X
iptables -t filter -P INPUT   DROP
iptables -t filter -P OUTPUT  DROP
iptables -t filter -P FORWARD DROP

# Initialisation de la table NAT
iptables -t nat -F
iptables -t nat -X
iptables -t nat -P PREROUTING  ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT      ACCEPT

# Initialisation de la table MANGLE
iptables -t mangle -F
iptables -t mangle -X
iptables -t mangle -P PREROUTING  ACCEPT
iptables -t mangle -P INPUT       ACCEPT
iptables -t mangle -P OUTPUT      ACCEPT
iptables -t mangle -P FORWARD     ACCEPT
iptables -t mangle -P POSTROUTING ACCEPT
```

- Avant de mettre en place un système de NAT, il faut bien prendre en compte tous les paramètres, et se rendre compte de l'impact que cela peut avoir sur la sécurité de votre réseau. Il n'y a rien de pire que de copier/coller un mauvais morceau de script "iptables", et de se dire que cela doit bien marcher tout seul. Voir de bidouiller affreusement le script dans son coin, et de jouer à l'apprenti sorcier ! 😃
- En cas de doute, ou de problème sur l'utilisation de Netfilter, il ne fait pas hésiter à utiliser la commande " iptables -L -v -t [table]" qui donne plein d'informations quand aux nombre de paquets passant par une règle ou un comportement par défaut.

Bien, l'IP masquerading étant vu, passons au port forwarding.

III-8-2 Port forwarding

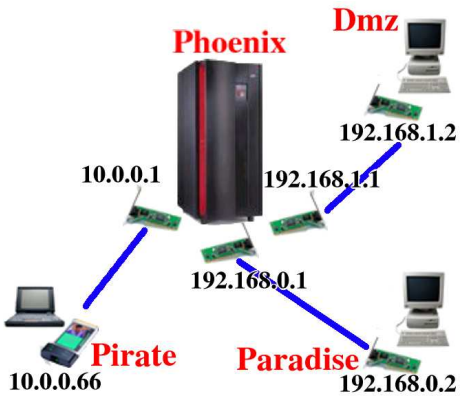
Bien caché derrière notre firewall Phoenix, notre réseau sky.net ne craint plus grand chose. De plus, avec ce que nous avons vu au chapitre précédent , les machines de notre réseau local peuvent rejoindre le réseau internet.net.

Maintenant, nous voudrions avoir un serveur (HTTP, FTP, IRC, peer-to-peer, etc...) qui puisse proposer des informations et des services non plus sur le réseau sky.net, mais carrément sur le réseau internet.net. L'emplacement d'un tel service devrait se faire en toute logique sur la machine Phoenix, qui est le seul accès que nous ayons sur le réseau internet.net. Mais ceci serait une très mauvaise idée . Pourquoi ? Parce qu'un tel accès est inévitablement une faille dans notre système de sécurité. Grâce à ce service que nous proposons, l'intrus pourrait essayer de se l'accaparer, puis d'en profiter pour prendre la main sur Phoenix, et d'enfin de court-circuiter toutes nos règles Netfilter. Notre machine se trouverait alors sans aucune protection vis à vis de l'extérieur... Inquiétant, non ? Au minimum, il faudrait configurer ce service pour fonctionner sous un utilisateur n'ayant que peu de droits, et en plus dans un chroot. Mais même avec cela, la sécurité ne pourrait pas être garantie.

Alors, comment faire ? C'est là qu'intervient le suivi de port ("port fowarding" en anglais). L'idée d'un tel système est que toute connexion entrante sur un certain port de phoenix1.internet.net (par exemple le port 80, pour du HTTP), soit automatiquement redirigée vers une autre machine de sky.net : paradise.sky.net par exemple. Vu de l'extérieur, on aurait l'impression que le port 80 de phoenix1.internet.net serait ouvert, mais en fait, ce serait le port 80 de paradise.sky.net qui serait réellement ouvert, et sur lequel tournerait un serveur Apache.

Est-ce réellement une "solution miracle" ? Presque, mais à condition de prendre quelques précautions :

- Dans cette configuration, nous ne faisons que déplacer le problème de sécurité sur Paradise. Il faut donc que cette machine soit très bien sécurisée, avec les paramétrages vus dans la section "risque" du chapitre II. Mais ce ne sera pas suffisant.
- Paradise devenant donc un serveur web à part entière, et du fait des risques énoncés, Phoenix devra donc s'en méfier comme de la peste. Fini les gentilles règles Netfilter laissant paradise.sky.net pleinement accéder à phoenix0.sky.net. Il faudra durcir tout cela, et utiliser le conntrack ! Dans notre esprit, paradise.sky.net devra devenir aussi soupçonnable que pirate.internet.net.
- Enfin, dans le cas où un intrus prendrait le contrôle de Paradise, et même si Phoenix à des règles Netfilter très strictes, rien n'empêcherait l'intrus de s'attaquer à une autre machine du réseau sky.net ! Voir dans le pire des cas, de s'approprier le contrôle d'une autre machine de sky.net, puis d'attaquer Phoenix par l'arrière, en profitant d'éventuelles règles Netfilter plus souples vis à vis de ce 2nd hôte...



DMZ pour port fowarding

En terme de sécurité informatique, la paranoïa peut devenir un jeu d'esprit où il faut systématiquement envisager les pires cas, et les hypothèses les plus tordues. Dans le cas de notre réseau, les 3 problèmes ci-dessus peuvent êtres réglés par une technique plutôt efficace, appelée DMZ ("DeMilitarized Zone ou "Zone Dé Militarisée" en français). La mise en place d'une telle DMZ se fait en installant une 3ème carte réseau sur Phoenix ("eth2"), sur laquelle on ne reliera que une seule machine qui ne servira qu'au seul usage de serveur web. On appellera cette machine DMZ, tout simplement. Voir à ce sujet l'illustration ci-contre. Mais ceci sort du cadre de ce document, et d'un réseau personnel, je n'en parlerai donc pas plus longuement.

Mais revenons à un réseau un peu plus simple, et baissons d'un cran notre paranoïa. Nous allons supposer que la machine qui hébergera notre serveur HTTP est fiable, et que personne ne peut en prendre la main (ceci n'est qu'une hypothèse bien sûr !). Nous la laisserons donc dans le réseau sky.net, en compagnie des autres machines de notre réseau interne.

Que devons nous faire ?

- Tout d'abord, nous devons initialiser la table NAT, tout comme nous l'avons fait pour le IP masquerading. Et aussi bien sûr charger le module

"iptables_nat" :

```
[root@phoenix /]# modprobe iptable_nat
[root@phoenix /]# iptables -t nat -F
[root@phoenix /]# iptables -t nat -X
[root@phoenix /]# iptables -t filter -P FORWARD     DROP
[root@phoenix /]# iptables -t nat    -P PREROUTING  ACCEPT
[root@phoenix /]# iptables -t nat    -P POSTROUTING ACCEPT
[root@phoenix /]# iptables -t nat    -P OUTPUT      ACCEPT
```

- Comme nous voulons partager un serveur HTTP, il peut être intéressant d'autoriser les machines de internet.net de pouvoir utiliser le "ping" sur notre serveur. Ce n'est absolument pas une obligation, mais cela peut-être pratique pour les Internautes qui, avant de lancer la connexion HTTP, vérifient que notre machine est bien active. Ce choix relève plus d'une bonne manière qu'autre chose, et n'est donc pas obligatoire. Notre "port forwarding" pourra très bien marcher sans cela. On utilise quand même du "suivi de connexion", afin de ne pas accepter les paquets ICMP volontairement mal formatés :

```
[root@phoenix /]# iptables -t filter -A INPUT  -i eth1 -s 0.0.0.0/0 -d 10.0.0.1 \
                           -p icmp -m state --state ! INVALID           -j ACCEPT
[root@phoenix /]# iptables -t filter -A OUTPUT -o eth1 -s 10.0.0.1  -d 0.0.0.0/0 \
                           -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT
```

  Au passage, vous noterez qu'il n'est pas nécessaire d'ouvrir d'autres accès. Et notamment via les chaînes "INPUT" et "OUTPUT" afin autoriser l'échange avec le port 80. Pourquoi ? Simplement parce que dans le cas du port forwarding, les trames à destination de paradise.sky.net ne sont pas destinées aux processus locaux, et donc elle ne passent pas à travers les chaînes "INPUT" et "OUTPUT". En fait, comme on le verra tout de suite, seul la chaîne "FORWARD" de la table "Filter" devra être modifiée.
- Nous allons maintenant faire suivre un certain type de paquets de l'interface "eth1" à l'interface "eth0" : Ce sera les paquets à destination du serveur HTTP. De même que nous laisserons passer les paquets en réponse. Là encore, nous utilisons du "conntrack" afin d'améliorer la sécurité de ce suivi de port :

```
[root@phoenix /]# iptables -t filter -A FORWARD -i eth1 -o eth0 -s 0.0.0.0/0   \
                           -d 192.168.0.2 -p tcp --dport 80                    \
                           -m state --state ! INVALID           -j ACCEPT
[root@phoenix /]# iptables -t filter -A FORWARD -i eth0 -o eth1 -s 192.168.0.2 \
                           -d 0.0.0.0/0   -p tcp --sport 80                    \
                           -m state --state RELATED,ESTABLISHED -j ACCEPT
```

- Nous arrivons à la 1er spécificité du "port forwarding". Pour les paquets entrant sur phoenix1.internet.net et à destination du port 80, nous allons modifier l'adresse de destination. Ce ne sera plus phoenix1.internet.net, mais paradise.sky.net, c'est à dire notre réel serveur HTTP. Pour cela, nous allons utiliser dans la table NAT, la chaîne "PREROUTING" et la cible "DNAT" (pour "Destination Network Adress Translation" en anglais).

```
[root@phoenix /]# iptables -t nat -A PREROUTING  -i eth1 -s 0.0.0.0/0   \
                           -d 10.0.0.1  -p tcp --dport 80               \
                           -m state --state ! INVALID -j DNAT --to-destination 192.168.0.2:80
```

  Au passage, on voit que l'on peut aussi modifier le port de destination, et mettre autre chose que 80. Cela peut-être utile si le serveur HTTP sur paradise.sky.net tourne sur autre chose que le port 80.
- La seconde spécificité du "port forwarding", c'est de modifier aussi l'adresse source de la requête. En effet, si nous laissons l'adresse source actuelle (une adresse de 10.0.0.0/8 par exemple), paradise.sky.net sera bien ennuyé pour répondre, car ce sera une adresse d'un réseau qu'il ne peut pas joindre. Évidement, nous pourrions indiquer à paradise.sky.net que phoenix0.sky.net est une passerelle, et dans ce cas là les paquets retrouveraient tout de suite la sortie du réseau sky.net. Mais, et même si cette solution marche effectivement très bien, c'est une solution particulièrement mal propre, comme nous le verrons un peu plus loin. Donc, modifions cette adresse source :

```
[root@phoenix /]# iptables -t nat -A POSTROUTING -o eth0 -s 0.0.0.0/0 \
                           -d 192.168.0.2 -p tcp --dport 80            \
                           -m state --state ! INVALID -j SNAT --to-source 192.168.0.1
```

- Bien, au niveau de Netfilter, tout est configuré. Il nous reste qu'à activer le NAT par la commande que nous connaissons déjà :

```
[root@phoenix /]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Le script iptables décrit ici est téléchargeable ici.

Il ne nous reste plus qu'à vérifier que pirate.internet.net a bien accès à phoenix1.internet.net :

```
[intrus@pirate /]$ telnet phoenix1.internet.net 80
Escape character is '^]'.
GET /
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.61 [en] (X11; I; Linux 2.2.9-23mdk i686) [Netscape]">
   <meta name="Author" content="MandrakeSoft">
   <title>Welcome to the Advanced Extranet Server, ADVX!</title>
   <LINK REL="SHORTCUT ICON" HREF="/favicon.ico">
<!-- Background white, links blue (unvisited), navy (visited), red (active) --> </head>

<body text="#000000" bgcolor="#FFFFFF" link="#0000FF" vlink="#000080" alink="#FF0000">
<table border=0>
<tr>
<td valign=top><a href=http://www.advx.org><img border=0 src=/icons/advx.png
ALT="Powered by ADVX.org software" height=47 width=102 align=LEFT></a></td>
<td valign=top width=100%><h2><center>Welcome to paradise.sky.net
</center></h2>
</td></tr>
```

La fin de la réponse a été tronquée. Bien que l'on accède à phoenix1.internet.net, c'est paradise.sky.net qui répond. Regardons maintenant le log du serveur Apache de paradise.sky.net :

```
[root@paradise /]# tail -1 /var/log/httpd/access_log
192.168.0.1 - - [06/Jul/2003:21:02:21 +0200] "GET /" 200 6988 "-" "-"
```

On voit la connexion faite par pirate.internet.net, et le téléchargement de la page web. Cependant, on notera que l'adresse IP qui est logée n'est pas celle de pirate.internet.net, mais bien celle de phoenix1.internet.net, ce qui est tout à fait normal, du fait de l'utilisation de la cible "SNAT".

Enfin, regardons ce que nous donne les statistiques de Netfilter sur Phoenix :

```
[root@phoenix /]# iptables -L -v -n -t filter
```

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp dpt:6000
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp dpt:6000
    0     0 ACCEPT     all  --  lo     *    0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  eth0   *    192.168.0.0/24 0.0.0.0/0
    0     0 ACCEPT     icmp --  eth1   *    0.0.0.0/0      10.0.0.1       state NEW,RELATED,
                                                                           ESTABLISHED

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source      destination
    8   430 ACCEPT     tcp  --  eth1   eth0 0.0.0.0/0      192.168.0.2    tcp dpt:80 state
                                                                           NEW,RELATED,ESTABLISHED
    8  7412 ACCEPT     tcp  --  eth0   eth1 192.168.0.2    0.0.0.0/0      tcp spt:80 state
                                                                           RELATED,ESTABLISHED

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
    0     0 ACCEPT     tcp  --  *      *    0.0.0.0/0      0.0.0.0/0      tcp spt:6000
    0     0 ACCEPT     all  --  *      lo   0.0.0.0/0      0.0.0.0/0
    0     0 ACCEPT     all  --  *      eth0 0.0.0.0/0      192.168.0.0/24
    0     0 ACCEPT     icmp --  *      eth1 10.0.0.1       0.0.0.0/0      state RELATED,
                                                                           ESTABLISHED
[root@phoenix /]# iptables -L -v -n -t nat
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    1    60 DNAT       tcp  --  eth1   *    0.0.0.0/0      10.0.0.1       tcp dpt:80 state NEW,RELATED,
                                                                           ESTABLISHED to:192.168.0.2:80

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out  source         destination
    1    60 SNAT       tcp  --  *      eth0 0.0.0.0/0      192.168.0.2    tcp dpt:80 state NEW,RELATED,
                                                                           ESTABLISHED to:192.168.0.1
```

La requête et l'envoi de la page HTML ont demandés au total 2 fois 8 paquets, qui sont passés à travers les chaînes "FORWARD".

On voit bien que les paquets entrant dans phoenix1.internet.net sont bien passés par les cibles "DNAT" et "SNAT" des chaînes "PREROUTING" et "POSTROUTING". Par contre ces statistiques ne montre pas les paquets en réponses à la demande de connexion. C'est dû à la même routine de suivi de connexion de Netfilter que pour l'IP masquerading.

Nous pourrions nous arrêter ici pour le port fowarding, mais un dernier point est intéressant à soulever : que se passe t'il si nous n'incluons pas la règles SNAT, ce qui est indiqué plus haut comme étant une " méthode sale " ?

Commençons par retirer cette règle :

```
[root@phoenix /]# iptables -D POSTROUTING 1 -t nat
```

Puis, indiquons à Paradise que sa passerelle est phoenix0.sky.net

```
[root@paradise /]# route add default gw phoenix0.sky.net
[root@paradise /]# route

Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         phoenix0.sky.ne 0.0.0.0         UG    0      0        0 eth0
```

Et maintenant, demandons à nouveau notre page HTML depuis pirate.internet.net. Suite à cette connexion, les statistiques d'"iptables" nous donne :

```
[root@phoenix /]# iptables -L -v -n -t nat
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out source       destination
    1    60 DNAT       tcp  --  eth1   *   0.0.0.0/0    10.0.0.1    tcp dpt:80 state NEW,RELATED,
                                                                     ESTABLISHED to:192.168.0.2:80

Chain POSTROUTING (policy ACCEPT 1 packets, 60 bytes)
 pkts bytes target     prot opt in     out source       destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
```

Comme précédemment, un paquet est passé par le chaîne "PREROUTING", mais aucun n'est passé par de règles "POSTROUTING". Et pour cause, il n'y en a plus. Par contre, le paquet de retour est passé par la règle par défaut de la chaîne POSTROUTING... C'est parce que les paquets n'empruntent pas les mêmes canaux en entrée et en sortie que j'estime cette méthode comme étant "sale".

Enfin ce qui est le plus intéressant c'est le log d'Apache sur paradise.sky.net :

```
[root@paradise /]# tail -1 /var/log/httpd/access_log
10.0.0.66 - - [06/Jul/2003:22:27:43 +0200] "GET /" 200 6988 "-" "-"
```

Cette fois ci, c'est bien l'adresse IP de la machine à distance qui est stockée, forcément, puisque Phoenix ne la dissimule plus.

Vous trouverez le script de cet exemple ici, mais comme indiqué, plus haut c'est la méthode "sale" pour effectuer du port forwarding...

Ceci conclue donc l'utilisation du port forwarding. Dans le cadre d'un usage personnel, cette technique est assez peu utile : qui chez lui héberge son propre serveur HTTP, FTP, IRC, etc...

Pas utile donc ? En fait non, il y a bien un usage personnel fort utile où l'on peut utiliser le port forwarding, et qui est particulièrement à la mode au jour où j'écris ces lignes : il s'agit du peer-to-peer (P2P). C'est un système d'échange de fichiers à travers Internet, où toutes les machines connectées partagent des fichiers sur un morceau de son disque dur. Chaque machine agit donc à la fois comme un client et un serveur. Une grand partie des clients pour ce type d'échange se trouvant sous Windows®, et comme c'est un OS dont l'usage courant est peu orienté sur la sécurité, on peut utiliser le port forwarding de Linux pour le protéger.

Avec les explications ci-dessus, vous pouvez sans difficulté écrire vos propres scripts iptables afin de renvoyer vos connexions entrantes vers la machine sur lequel tourne votre client P2P. Cependant, souvenez vous que vous ne faites que déplacer le problème de sécurité sur cette machine ci, et donc que c'est à vous d'en assurer la responsabilité. Si cette machine est par exemple sous Windows®, la présence de Netfilter en tête de votre réseau ne vous dispense pas de mettre en place un firewall logiciel sur cette machine, afin par exemple de surveiller ce que fait votre Windows®. Ceci étant, je

me lave les mains de ce que vous pourriez faire de l'usage du port forwarding et du P2P. Et si je ne fournis aucun script d'exemple, c'est entre autre parce que je n'utilise pas ce type de logiciel... 😀

Enfin, je dirais que la mise en place du port forwarding n'est pas forcément ce qui se fait le plus facilement, bien qu'en fait les règles ne soit pas beaucoup plus complexes que pour l'IP masquerading. Sur Internet, vous trouverez beaucoup de bribes d'explications sur cette techniques, et beaucoup de scripts plus où moins bien fait à ce sujet. Méfiez vous donc de ce que vous trouverez. Le mieux est sans aucun doute de tester votre configuration par étape, et de regarder systématiquement les statistiques d'iptables afin de voir par où passent vos paquets. D'une manière générale, si les paquets utilisent les cibles par défaut de PREROUTING ou de POSTROUTING, c'est qu'il y a probablement une mauvaise configuration de votre script. Enfin, comme indiqué dans les 2 scripts "iptables-portforwarding-*.sh" que vous pouvez télécharger, vous avez la possibilité d'utiliser la cible LOG afin d'analyser quelles sont les paquets qui utilisent ces cibles par défaut.

III-9 Log (LOG / ULOG)

Jusqu'à présent, nous avons vu comment configurer nos règles avec Netfilter : définir les cibles par défaut, et supprimer des paquets. Mais à force de tout supprimer, il nous est difficile de connaître l'efficacité de notre firewall, et si il est intéressant de rajouter ou de supprimer des règles. D'où l'intérêt de logger certaines informations.

Dans ce qui suis, nous allons utiliser l'affreux anglicisme "logger" qui décrit l'action de stocker dans un "log" une certaine information.

III-9-1 LOG

C'est la méthode la plus standard pour logger des trames. Il s'agit simplement de créer une règle "iptables" dont la cible ("-j [cible]") n'est pas "ACCEPT" ou "DROP", mais tout simplement "LOG". Si nous reprenons le script "iptables-conntrack-1.sh", nous pouvons par exemple rajouter une dernière règle qui va logger tout ce qui na pas été accepté par la chaîne "INPUT". C'est très pratique, car ainsi nous serons averti de tout ce qui tente d'accéder aux processus utilisateurs de notre système. Pour cela, nous rajoutons cette règle en temps que dernière règle de la chaîne "INPUT" :

```
[root@phoenix /]# iptables -t filter -A INPUT -j LOG
```

Comme Netfilter est un élément du Kernel, ses logs sont donc des logs de ce dernier. Et comme tout bon log Kernel, ils se retrouve dans le fichier "/var/log/messages". Ainsi, si pirate.internet.net fait un "nmap" sur les ports HTTP et HTTPS de Phoenix, nous aurons :

```
[intrus@pirate /]# nmap phoenix1.internet.net -p 80,443

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds

[root@phoenix /]# tail -14 /var/log/messages
Jul  8 23:04:59 phoenix nmbd[2178]: [2003/07/08 23:04:59, 0] nmbd/nmbd_packets.c:send_netbios_packet(172)
Jul  8 23:04:59 phoenix nmbd[2178]:   send_netbios_packet: send_packet() to IP 10.255.255.255 port 137 failed
Jul  8 23:04:59 phoenix nmbd[2178]: [2003/07/08 23:04:59, 0] nmbd/nmbd_namequery.c:query_name(256)
Jul  8 23:04:59 phoenix nmbd[2178]:   query_name: Failed to send packet trying to query name WORKGROUP
Jul  8 23:07:03 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=28 TOS=0x00 PREC=0x00
                                TTL=38 ID=41593 PROTO=ICMP TYPE=8 CODE=0 ID=58020 SEQ=0
Jul  8 23:07:03 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00
                                TTL=39 ID=51459 PROTO=TCP SPT=53120 DPT=80 WINDOW=4096 RES=0x00 ACK URGP=0
Jul  8 23:07:09 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=28 TOS=0x00 PREC=0x00
                                TTL=38 ID=2581 PROTO=ICMP TYPE=8 CODE=0 ID=58020 SEQ=256
Jul  8 23:07:09 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00
                                TTL=39 ID=27444 PROTO=TCP SPT=53121 DPT=80 WINDOW=4096 RES=0x00 ACK URGP=0
Jul  8 23:07:15 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=28 TOS=0x00 PREC=0x00
                                TTL=38 ID=29887 PROTO=ICMP TYPE=8 CODE=0 ID=58020 SEQ=512
Jul  8 23:07:15 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00
                                TTL=39 ID=48409 PROTO=TCP SPT=53122 DPT=80 WINDOW=4096 RES=0x00 ACK URGP=0
Jul  8 23:07:21 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=28 TOS=0x00 PREC=0x00
                                TTL=38 ID=37813 PROTO=ICMP TYPE=8 CODE=0 ID=58020 SEQ=768
Jul  8 23:07:21 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00
                                TTL=39 ID=17449 PROTO=TCP SPT=53123 DPT=80 WINDOW=4096 RES=0x00 ACK URGP=0
Jul  8 23:07:27 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=28 TOS=0x00 PREC=0x00
                                TTL=38 ID=57294 PROTO=ICMP TYPE=8 CODE=0 ID=58020 SEQ=1024
Jul  8 23:07:27 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00
                                TTL=39 ID=64622 PROTO=TCP SPT=53124 DPT=80 WINDOW=4096 RES=0x00 ACK URGP=0
```

Grâce au log, on voit que phoenix1.internet.net reçoit une série de commandes "ping" et de requêtes sur son port 80 de la part de pirate.internet.net. Évidement, comme Phoenix ne répond pas à ces requêtes, Pirate fait plusieurs essais, et finalement ne testera même pas le port 443.

Un moyen pratique de suivre ses logs en temps réel est la commande, lancé en temps que root : " tail -f /var/log/messages". Cela affichera en permanence la fin de ce fichier de log. Pour l'arrêter, il suffit d'appuyer sur CTRL+C.

Mais ce fichier sert aussi (surtout !) à stocker tout les messages d'informations ou d'erreurs du Kernel. Il nous faut donc un moyen de retrouver ces messages de log Netfilter parmi tout les autres messages. On peut configurer avec le paramètre " --log-prefix=[Message de log]" la règle de log afin qu'elle rajoute systématiquement des messages en début de log. Ainsi :

```
[root@phoenix /]# iptables -t filter -A INPUT -s 10.0.0.66 -j LOG --log-prefix="AttackPirate"
[root@phoenix /]# iptables -t filter -A INPUT -s 10.0.0.200 -j LOG --log-prefix="AttackWeb"
```

indiquera clairement les connexions faites par les machines pirate.internet.net et web.internet.net :

```
[intrus@pirate /]$ nmap -p 80 phoenix1.internet.net
[intrus@web /]$ nmap -p 80 phoenix1.internet.net
[root@phoenix /]# tail -f /var/log/messages
Jul  8 23:26:06 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:26:06 phoenix kernel: AttackWeb IN=eth1 OUT= SRC=10.0.0.200 DST=10.0.0.1 ...
Jul  8 23:26:12 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:26:12 phoenix kernel: AttackWeb IN=eth1 OUT= SRC=10.0.0.200 DST=10.0.0.1 ...
Jul  8 23:26:12 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:26:12 phoenix kernel: AttackWeb IN=eth1 OUT= SRC=10.0.0.200 DST=10.0.0.1 ...
Jul  8 23:26:25 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:26:28 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:26:34 phoenix kernel: AttackPirate IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
```

Cependant, cette méthode de log a l'inconvénient que même avec des "--log-prefix", il reste difficile de séparer les messages de log de Netfilter des autres messages du Kernel. Comme en témoigne d'ailleurs le premier affichage du /var/log/message, où des messages de Samba (en début de log) sont mélangés aux messages de Netfilter.

Une autre méthode est de définir un "niveau de log" ("log level" en anglais), qui rajoute une autre information aux log, ce qui permet au demon de log (un programme appelé "syslogd") de stocker ces log dans un autre fichier. Par exemple, sur une Mandrake, on peut utiliser (voir "man syslogd", "man syslog.conf", "less /usr/include/sys/syslog.h" pour avoir plus d'informations sur ces commandes) :

```
[root@phoenix /]# iptables -t filter -A INPUT -j LOG --log-level=4
[root@phoenix /]# less /usr/include/sys/syslog.h
#define LOG_WARNING     4       /* warning conditions */
[root@phoenix /]# less /etc/syslog.conf
kern.=warn                      -/var/log/kernel/warnings
[intrus@pirate /]$ nmap -p 80 phoenix1.internet.net

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 60 seconds
[root@phoenix /]# tail -10 /var/log/kernel/warnings
Jul  8 18:55:43 phoenix kernel: MSDOS FS: Using codepage 850
Jul  8 19:26:34 phoenix kernel: i2c-amd756.o version 2.7.0 (20021208)
Jul  8 19:26:34 phoenix kernel: i2c-amd756.o: AMD768 bus detected and initialized
Jul  8 19:28:37 phoenix kernel: UDF-fs: No VRS found
Jul  8 23:53:58 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:54:01 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:54:07 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:54:10 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:54:13 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
Jul  8 23:54:19 phoenix kernel: IN=eth1 OUT= SRC=10.0.0.66 DST=10.0.0.1 ...
```

Comme on le voit, bien que les log de Netfilter soient isolés dans un fichiers à part (/var/log/warnings), ce fichier n'en est pas moins partagé avec d'autres modules du Kernel, qui utilisent eux aussi le log Kernel de niveau 4 <=> Warning. Dans le précédent log, on voit notamment des logs de "MSDOS FS" et "i2c-amd756.0" qui n'ont rien à voir avec Netfilter.

On pourrait s'imaginer utiliser une autre entrée du syslog, afin de vraiment séparer ces log. Le seul problème, c'est que les log de Netfilter sont des log du Kernel, et qu'à ce titre, on sera toujours obligé de les déclarer en temps que " kern." dans le "/etc/syslog.conf".

Alors ? Stocker ces logs à part est vraiment une chose impossible sous Linux ? Non, heureusement que non ! La solution s'appelle ULOG, et nous allons la voir tout de suite.

III-9-2 ULOG

ULOG est module du Kernel dont le développement a été fait par http://www.gnumonks.org/projects/. Il a été spécialement conçu pour recevoir les log de Netfilter. Il y a certaines (petites) contraintes à son utilisation :

- Kernel récent : il faut un Kernel >= 2.4.18-pre8 pour pouvoir utiliser ce module.
- Option de compilation : il faut que votre kernel soit compilé avec l'option CONFIG_IP_NF_TARGET_ULOG=m.
- Une fois compilé, le module ipt_ULOG.o doit se trouver sur votre disque dur (/lib/modules/[version du kernel]/kernel/net/ipv4/netfilter/ipt_ULOG.o.gz par exemple) :

  ```
  [root@phoenix /]# find /lib/modules/`uname -r`/ -iname "*ULOG*"
  /lib/modules/2.4.21-0.13mdksmp/kernel/net/ipv4/netfilter/ipt_ULOG.o.gz
  ```

  Sur une distribution "Mandrake 9.1", tout ceci est fait par défaut.
- Vous devez récupérer et installer le demon "ulogd" sur votre machine. Personnellement, j'ai téléchargé la version 1.0, et je l'ai compilé et installé dans mon "/usr/local/sbin/ulogd".
- Il faut configurer le demon ulogd. Pour cela, il faut éditer son fichier de configuration. Vous pouvez utiliser le mien. Les 2 informations importantes sont de spécifier qu'ULOG doit stocker ses logs sous forme de fichier texte, dans "/var/log/ulogd.syslogemu" par exemple.
- Le demon ULOG ("ulogd") doit tourner sur votre machine. L'idéal est de le lancer au démarrage, en même temps que les autres demons de votre machine. A toute fin utile, voici le script de démarrage du demon ULOG que j'ai écris. Il se place dans le "/etc/init.d/". Pour le démarrer systématiquement, il faut créer un lien symbolique du "/etc/rc.d/rc3.d/S[un nombre]ulogd" ou du "/etc/rc.d/rc5.d/S[un nombre]ulogd" vers "/etc/init.d/ulog". Exemple :

  ```
  [root@phoenix scripts]# ls -la /etc/init.d/ulogd
  -rwxr--r--    1 root     root         1821 jui  9 00:30 /etc/init.d/ulogd
  [root@phoenix scripts]# ls -la /etc/rc.d/rc5.d/S10ulogd
  lrwxrwxrwx    1 root     root           15 mai  4 21:02 /etc/rc.d/rc5.d/S10ulogd -> ../init.d/ulogd
  ```

Une fois que tout ceci est mis en place :

- Démarrez le demon ulog . Par exemple :

  ```
  [root@phoenix /]# /etc/rc.d/rc5.d/S10ulogd start
  ```

- Vérifiez que le démon fonctionne correctement :

  ```
  [root@phoenix /]# cat /var/log/ulogd.log
  Wed Jul  9 00:50:27 2003 <3> ulogd.c:474 ulogd Version 1.00 starting
  Wed Jul  9 00:50:27 2003 <5> ulogd.c:688 initialization finished, entering main loop
  ```

- Configurez Netfilter pour qu'il utilise la cible ULOG au lieu de la cible LOG. Ici, on log tout ce qui va être supprimé par la chaîne "INPUT" :

  ```
  [root@phoenix /]# iptables -t filter -A INPUT -p all -j ULOG --ulog-prefix=DefaultDrop
  ```

- Et attendez qu'un intrus vienne se frotter à votre Netfiler :

  ```
  [root@phoenix /]# tail -f /var/log/ulogd.syslogemu
  Jul  8 20:24:22 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21 LEN=61
                  TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32796 LEN=41
  Jul  8 20:24:22 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21 LEN=40
                  TOS=00 PREC=0x00 TTL=251 ID=0 DF PROTO=TCP SPT=110 DPT=33108 SEQ=0
                  ACK=1355878989 WINDOW=0 ACK RST URGP=0
  Jul  8 20:24:32 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21 LEN=61
                  TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32797 LEN=41
  Jul  8 20:24:54 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
                  LEN=61 TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32799 LEN=41
  Jul  8 20:25:04 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
                  LEN=61 TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32800 LEN=41
  Jul  8 20:25:22 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
                  LEN=78 TOS=00 PREC=0x00 TTL=104 ID=36090 PROTO=UDP SPT=1034 DPT=137 LEN=58
  Jul  8 20:25:28 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
  ```

```
                LEN=40 TOS=00 PREC=0x00 TTL=251 ID=0 DF PROTO=TCP SPT=110 DPT=33220 SEQ=0
                ACK=1428948021 WINDOW=0 ACK RST URGP=0
Jul  8 20:25:34 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
                LEN=61 TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32802 LEN=41
Jul  8 20:25:38 phoenix DefaultDrop IN=ppp0 OUT= MAC= SRC=xx.xx.xx.xx DST=62.147.74.21
                LEN=61 TOS=00 PREC=0x00 TTL=60 ID=0 DF PROTO=UDP SPT=53 DPT=32803 LEN=41
```

Ceci n'est qu'une simple portion de mon log de Netfilter, lors d'une connexion en RTC le 8 Juillet 2003. Remarquez le temps plutôt faible entre les accès. Il s'agit en fait de plusieurs "port scan" effectués par plusieurs machines... Et encore, mon Netfilter est configuré pour ne pas afficher les logs des requêtes P2P qui, bien que je n'ai aucun client P2P, inondent régulièrement ma machine...

Tout comme la cible LOG, la cible ULOG a des options intéressantes :

- --ulog-prefix : préfixe des log. Même chose que pour "--log-prefix" de la cible LOG
- --ulog-nlgroup : similaire à "--log-level" de la cible LOG

Conclusion : si vous voulez avoir des logs bien exploitables de votre Netfilter, utilisez ULOG plutôt que LOG. Ce n'est finalement pas si compliqué à installer, et c'est très pratique. Enfin, pour les maniaques de la sécurité qui ont plein de place disque et du CPU à revendre, vous pouvez spécifier au demon "ulogd" de stocker les paquets non pas dans un fichier, mais dans une base de donnés MySQL ou PostgreSQL.

III-10 Autres astuces

III-10-1 Règles par défaut ("Policy")

Jusqu'à présent, nous avons vu que nous devions initialiser et définir les règles par défaut de ta table "Filter", et parfois celles de la table "NAT". En fait, ce n'est pas tout à fait la bonne manière de faire. A supposer par exemple que vous voulez faire uniquement du filtrage (option "-t filter"), vous pouvez très bien avoir laissé quelques règles de PREROUTING ou de POSTROUTING activées dans la table NAT. Dans ce cas, vous ne savez pas forcément ce que fait Netfilter, et il se peut qu'avec de telles configurations, il laisse passez des trames sans que vous ne vous en doutiez.

Donc il est est primordial que la première chose que vous fassiez dans un script Netfilter, c'est d'initialiser toutes les tables ("Filter", "NAT" et "Mangle") :

```
[root@phoenix /]# iptables -t filter -F
[root@phoenix /]# iptables -t filter -X
[root@phoenix /]# iptables -t filter -P INPUT   DROP
[root@phoenix /]# iptables -t filter -P FORWARD DROP
[root@phoenix /]# iptables -t filter -P OUTPUT  DROP
[root@phoenix /]# iptables -t nat -F
[root@phoenix /]# iptables -t nat -X
[root@phoenix /]# iptables -t nat -P PREROUTING  ACCEPT
[root@phoenix /]# iptables -t nat -P OUTPUT      ACCEPT
[root@phoenix /]# iptables -t nat -P POSTROUTING ACCEPT
[root@phoenix /]# iptables -t mangle -F
[root@phoenix /]# iptables -t mangle -X
[root@phoenix /]# iptables -t mangle -P PREROUTING  ACCEPT
[root@phoenix /]# iptables -t mangle -P INPUT       ACCEPT
[root@phoenix /]# iptables -t mangle -P FORWARD     ACCEPT
[root@phoenix /]# iptables -t mangle -P OUTPUT      ACCEPT
[root@phoenix /]# iptables -t mangle -P POSTROUTING ACCEPT
```

Vous remarquerez que l'on définie à "ACCEPT" les règles par défaut des tables "NAT" et "Mangle". Cela n'a pas trop d'influence sur la sécurité, du moment que vos règles de FORWARD sont bien écrites. Sinon, vous pouvez les mettre à "DROP", mais cela rallongera énormément le code et le temps de développement de votre script Netfilter.

Ce n'est pas non plus une mauvaise chose que de désactiver en début de script , au moins temporairement, le NAT (on ne sait jamais, des fois que vous l'ayez oublié) :

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

III-10-2 Chaînes utilisateurs

Nous avons peu parlé des chaînes utilisateurs, aussi nous allons y jeter un oeil. Lorsque vous écrivez vos règles Netfilter, il y a parfois des morceaux de code que vous aimeriez mettre en commun. Par exemple, supposons que vous voudriez interdire le "ping" de certaines machines du réseau local ainsi que du réseau externe, mais que vous vouliez aussi "logger" toute utilisation du "ping". Vous auriez alors à écrire quelque chose comme :

```
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.2 -p icmp -j LOG
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.2 -p icmp -j DROP
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.3 -p icmp -j LOG
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.3 -p icmp -j DROP
[root@phoenix /]# iptables -t filter -A INPUT -i eth1 -p icmp -j LOG
[root@phoenix /]# iptables -t filter -A INPUT -i eth1 -p icmp -j DROP
```

Dans ces cas, les règles utilisateurs sont là pour simplifier la vie. Commençons par écrire notre propre règle "LogDrop" qui comme son nom l'indique, va "logger" les trames, puis les supprimer :

```
[root@phoenix /]# iptables -t filter -N LogDrop
[root@phoenix /]# iptables -t filter -A LogDrop -j LOG --log-prefix LogDrop
[root@phoenix /]# iptables -t filter -A LogDrop -j DROP
```

Puis, appelons la pour nos pings sur les réseaux locaux :

```
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.2 -p icmp -j LogDrop
[root@phoenix /]# iptables -t filter -A INPUT -i eth0 -s 192.168.0.3 -p icmp -j LogDrop
```

Et les réseaux externes :

```
[root@phoenix /]# iptables -t filter -A INPUT -i eth1 -p icmp -j LogDrop
```

On note au passage que notre chaîne "LogDrop" peut être appelée à n'importe quelle occasion, avec une règle "parente" ayant ou non beaucoup d'options. Le script vu ci-dessus se trouve ici.

III-10-3 Script final d'exemple

Nous avons vu jusqu'à présent un bon nombre de fonctionnalités de Netfilter grâce aux commandes "iptables". J'ai donc écrit un script qui regroupe toutes ces fonctionnalités en un seul script que vous pouvez utiliser chez vous. Ce script est téléchargeable ici.

Avant toute chose, il faudra que vous le configuriez à votre usage. Pour cela, le début du script contient un certain nombre de variables globales, qui définissent le comportement du script :

- LAN_* (Local Area Network) : ces variables définissent le réseau local. Dans notre exemple, il s'agit du réseau " sky.net".
- WAN_* (Wan Area Network) : là, il s'agit des variables définissant le réseau Internet. Elles sont spécialement conçues pour un réseau connecté par l'intermédiaire d'une passerelle. Or, vous devez plutôt avoir un modem RTC/RNIS/ADSL pour vous connecter, non ? Dans ce cas, il vous faudra donc décommenter le 2nd paquet de lignes "WAN_*", qui définirons vos paramètres de connexions Internet. La seule variable qu'il faudra bien vérifier est "WAN_INTERFACE=ppp0", mais à priori, vous ne vous connectez à Internet que par l'interface "ppp0". Dans le doute, et une fois que votre connexion Internet est activé, les commandes "/sbin/ifconfig" et "/sbin/route" vous donnerons plus d'informations.
- NAT (Network Adress Translation) : cette variable définie si ou non vous voulez autorisez les machines de votre réseau interne à se connecter à Internet.
- PF_* (Port Forwarding) : ces variables définissent si ou non vous voulez faire du port forwarding. Notez que j'ai pris ici l'exemple le plus simple : le HTTP ne demande en effet qu'un seul port omnidirectionnel. Si vous voulez faire du port forwarding sur du FTP, ce sera un peu plus compliqué, car il faut laisser passer le canal de données (FTP-DATA) qui utilise le port 20 du coté client. Pour le P2P, c'est votre port de connexion (par exemple, "4660") qu'il faudra laisser passer en entrée.

Quelques remarques à propos de ce script :

- Toutes les règles Netfilter qui contrôlent l'accès à Internet utilisent l'adresse IP externe de la machine ("WAN_IP"). C'est une manière de faire, qui je l'admets est contestable. Beaucoup de scripts que vous trouverez sur Internet n'utilisent pas ces options ("-d $WAN_IP" pour les règles "INPUT" et "-s $WAN_IP" pour les règles "OUTPUT"), et ne restreignent tout simplement pas les connexions externes à l'adresse IP externe. Par exemple, au lieu d'écrire :

```
[root@phoenix /]# iptables -t filter -A OUTPUT -o $WAN_INTERFACE -s $WAN_IP -d $WAN_NETWORK \
                           -p all -m state --state ! INVALID          -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i $WAN_INTERFACE -s $WAN_NETWORK -d $WAN_IP \
                           -p all -m state --state RELATED,ESTABLISHED -j ACCEPT
```

  Ces autres scripts écriraient :

```
[root@phoenix /]# iptables -t filter -A OUTPUT -o $WAN_INTERFACE -d $WAN_NETWORK \
                           -p all -m state --state ! INVALID          -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i $WAN_INTERFACE -s $WAN_NETWORK \
                           -p all -m state --state RELATED,ESTABLISHED -j ACCEPT
```

  C'est effectivement plus simple à écrire, mais personnellement je n'ai pas confiance en ce type d'écriture "courte". Et ce, pour 2 raisons :
  - On peut toujours craindre une attaque malicieuse où un intrus nous enverrait un paquet mal formé, avec une adresse IP qui n'est pas la nôtre. C'est spécialement faisable si l'intrus se trouve être chez notre propre fournisseur d'accès. Ce paquet n'étant pas du tout normal, il n'a aucune raison de rentrer dans notre machine. On peut aussi imaginer qu'il indique comme adresse de destination une adresse de notre réseau interne. Ce type de paquet n'est pas du tout supposé pouvoir pénétrer dans notre machine, mais les techniques d'intrusion évoluant sans cesse, qui sait si un jour une telle technique ne sera pas capable de contourner les protections de Netfilter ?
  - "Iptables" permet d'être très "fin" au niveau de ses règles de filtrages, je trouve donc tout simplement regrettable de ne pas exploiter au maximum ses possibilités, et donc de ne pas faire des règles "ACCEPT" les plus restrictives possibles.

  Conclusion : dans le doute, je m'abstiens, et j'utilise la "WAN_IP"... 😀

  Mais ce choix a un désavantage important : pour que ce script soit fonctionnel, il faut le lancer à chaque fois que la configuration réseau change, c'est à dire à chaque connexion à Internet. Et c'est la commande (barbare ?) "**/sbin/ifconfig | grep "P-t-P" | sed "s/^[: a-z]*\([.0-9]*\).*/\1/g"**" qui va se charger de trouver l'adresse IP internet de votre machine. Mais plutôt que de le lancer manuellement, vous pouvez laisser Linux s'en occuper pour vous. En effet, à chaque fois que la connexion ppp est initialisée, Linux exécute le script " /etc/sysconfig/network-scripts/ifup-ppp" (*). Vous n'avez donc qu'à rajouter une ligne lançant ce script au début de ce fichier. Par exemple, quelque chose comme "/usr/local/sbin/iptables-final-1.sh".

  Pour des raisons de sécurité, je vous conseille de donner la propriété de ce fichier au root, et de le laisser en écriture uniquement pour le super utilisateur.
  Exemple :

```
[root@phoenix /]# chown root:root /usr/local/sbin/iptables-final-1.sh
[root@phoenix /]# chroot 755 /usr/local/sbin/iptables-final-1.sh
```

  (*): en fait, ce n'est pas une obligation absolue. Et cela dépend en partie de l'outil que vous utilisez pour vous connecter sur Internet. Par exemple, pour ma connexion modem j'utilise l'interface graphique "/usr/bin/kppp". Et ce programme propose de lancer un script une fois la connexion Internet établie. C'est là que l'on pourrait mettre le " /usr/local/sbin/iptables-final-1.sh"
- Dans ce script, vous pouvez voir que les règles de port forwarding sont situées avant celles d'IP masquerading. Simple effet de style de ma part ? Non, pas du tout. Un paquet venant d'Internet et à destination du port 80 de la machine satisfait à 2 règles de la chaîne "FORWARD" :

```
iptables -t filter -A FORWARD -i $WAN_INTERFACE -o $LAN_INTERFACE -s $WAN_NETWORK -d $LAN_NETWORK \
                   -p $PF_PROTO --dport $PF_PORT -m state --state ! INVALID           -j ACCEPT
iptables -t filter -A FORWARD -i $WAN_INTERFACE -o $LAN_INTERFACE -s $WAN_NETWORK -d $LAN_NETWORK \
                   -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
```

  De même que pour les paquets sortants de la machine et ayant pour source le port 80 :

```
iptables -t filter -A FORWARD -i $LAN_INTERFACE -o $WAN_INTERFACE -s $LAN_NETWORK -d $WAN_NETWORK \
                   -p $PF_PROTO --sport $PF_PORT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A FORWARD -i $LAN_INTERFACE -o $WAN_INTERFACE -s $LAN_NETWORK -d $WAN_NETWORK \
                   -p all -m state --state ! INVALID           -j ACCEPT
```

  Dans les 2 cas, la 1ère règle est une règle de port forwarding et la 2nd d'IP masquerading. Mais comme les 2nd règles sont plus générales que les premières, les statistiques de Netfilter (commande "iptables -L -n -v -t filter") sont faussées si les règles d'IP masquerading sont placées avant les règles de port forwarding.

  Cet exemple est intéressant, car c'est un des fameux cas où l'ordre des commandes "iptables" a une certaine importance. Comment cela je chipote ? Et bien oui... 😀
- Notez enfin que les différents modules de Netfilter ("iptable_nat", "ip_nat_ftp", etc ...) ne sont chargés que si c'est nécessaire. On aurait par contre pu tous les charger "en bloc" au début du script.

III-10-4 Autres scripts

A titre d'informations, vous pouvez télécharger quelques scripts supplémentaires :

Nom du script	Description
iptables-extrem-accept.sh	Ce script vide toutes les chaînes prédéfinies et supprime toute les chaînes utilisateurs de toutes les tables. C'est utile si vous voulez revenir à une configuration de Netfilter complètement vierge, et aussi complètement insécurisée. Bref, c'est la configuration que vous avez probablement actuellement !

iptables-extrem-drop.sh	Ce script est pour les ultra paranoïaques, car contrairement à son prédécesseur, il interdit toute connexion (après avoir au préalable supprimé toutes les règles et toutes les chaînes utilisateurs). Avec une telle configuration, il ne vous reste plus qu'à débrancher les câbles réseaux de votre machine, et en faire des colliers. Au moins, ils vous seront utiles à quelque chose !

III-10-5 Tests d'intrusion

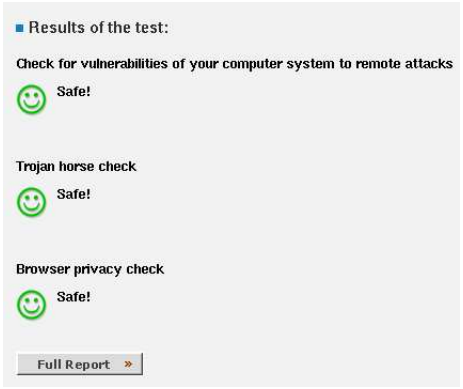
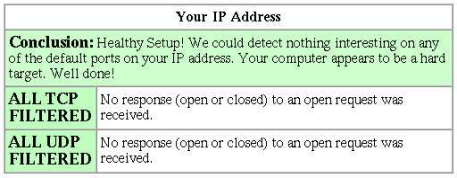
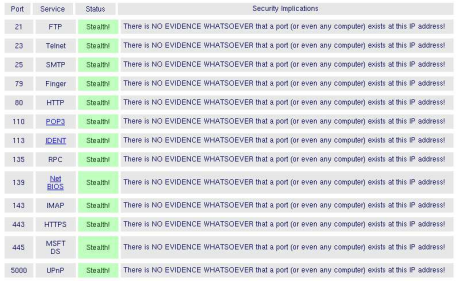
Une fois que toutes vos règles iptables sont écrites, il vous faut vous-même tester la sécurité de votre Linux en pratiquant des tests d'intrusions. Une première approche est d'utiliser "nmap", et de contrôler chacun de vos ports. Cette pratique est très efficace (lisez "man nmap" afin d'utiliser au mieux les options très variées de Nmap), et doit être fait depuis les machines de votre réseau. Pendant vos "attaques" lancées par "nmap", n'oubliez pas de garder un oeil sur votre "/var/log/messages" (par exemple avec un "tail -f /var/log/messages") ou le fichier qui stocke les logs de Netfilter, afin de suivre la progression de l'attaque.

Dans le cadre du réseau proposé ici, il m'a été facile de tester les 2 interfaces réseaux de Phoenix, en lançant les tests depuis Paradise et Pirate. Cependant, dans le cadre d'une utilisation personnelle, il est beaucoup plus difficile de tester l'interface réseau externe ("ppp0" par exemple). En effet, si par exemple depuis Phoenix vous lancez un "nmap phoenix1.internet.net", vous ne ferez que tester les règles de "loopback/localhost", comme nous l'avons déjà vu précédemment . Pour que vous obteniez un résultat correct, il vous faut donc changer vos règles de "loopback" par celles que vous avez définies pour "ppp0". Mais cette solution n'est pas forcément très représentative de la réalité, et le mieux serait de tester réellement les paquets arrivant sur "ppp0".

Il vous faut donc faire confiance à une personne externe à votre réseau, situé sur Internet, pour tester vos règles de Firewall. Pour cela, 3 possibilités :

- Vous avez un accès en "telnet" ou en "SSH" (ce qui est mieux, car plus sécurisé) sur une machine situé sur Internet. Vous vous connectez alors dessus, et vous lancez un "nmap" sur l'adresse IP de votre machine. Attention de ne pas vous tromper d'adresse IP, sinon vous iriez tester les ports d'une autre machine, et son propriétaire pourrait ne pas être très content...
- Vous demandez à un de vos amis de faire ce test pour vous. Encore faut il qu'il sache ce qu'est "nmap" !
- Enfin, vous faire faire ce test par un site web. Certains sites proposent gratuitement ce genre de prestation, entre autre pour vous proposer des solutions de firewall ... Windows®

J'ai testé certains de ces sites, et voici les résultats :

Site de test	Résultat
PcFlank	
DSLreports	
Shields UP!	
AuditMyPc.com	Le site a refusé de tester la machine , ce qui est apparemment dût à la présence du proxy transparent de mon fournisseur d'accès à Internet

Sans surprise, non ?

En fait, tout le mérite en revient au suivi de connexion, qui fait passer notre machine pour un "trou noir". Comme le veut la définition de ce terme, c'est "un corps qui absorbe tout, et qui ne rejette rien"...

Quelques soit ces sites, leurs tests ne sont souvent pas très poussés, et se limitent parfois qu'aux seuls ports ouverts par défaut sous Windows®. De plus, un simple test de "port scanning" n'est pas forcément suffisant, et d'autres outils de tests d'intrusion peuvent être utilisés en complément. Mais à ce niveau là, la limite entre tests de sécurité et piratage est très tenue, donc je vous laisserai chercher par vous-même des moyens de tester un peu plus en profondeur votre système...

III-10-6 Sauvegarde des règles Netfilter

Supposons que vous n'écriviez pas votre adresse IP Internet dans vos règles Netfilter de "ppp0" (comme décrit ci-dessus), vous avez alors des règles de ce type :

```
[root@phoenix /]# iptables -t filter -A OUTPUT -o ppp0 -d 0.0.0.0/0 \
                        -p all -m state --state ! INVALID           -j ACCEPT
[root@phoenix /]# iptables -t filter -A INPUT  -i ppp0 -s 0.0.0.0/0 \
                        -p all -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Dans ce cas, vos règles Netfilter sont génériques , et elles n'ont pas besoin d'être modifiées à chaque connexion Internet. Donc vous pouvez écrire vos règles une fois pour toute, et laisser Linux les charger automatiquement au démarrage de votre machine. Pour cela, écrivez vos règles Netfilter dans un script, puis (pour les possesseurs de distributions Mandrake) tapez la commande :

```
[root@phoenix /]# /etc/rc.d/init.d/iptables save
```

Enfin, pour que le Netfilter soit opérationnel à chaque démarrage, il faut créer les liens symboliques adéquates dans le /etc/rc.d/ :

```
[root@phoenix /]# cd /etc/rc.d/rc3.d/
[root@phoenix /]# ln -s ../init.d/iptables s03iptables
[root@phoenix /]# cd /etc/rc.d/rc5.d/
[root@phoenix /]# ln -s ../init.d/iptables s03iptables
```

Au démarrage, votre Linux lancera le script /etc/rc.d/init.d/iptables qui chargera les règles Netfilter précédemment sauvées.
Si à un moment vous avez besoin d'arrêter la protection de Netfilter, vous n'avez qu'à lancer la commande suivante :

```
[root@phoenix /]# /etc/rc.d/init.d/iptables stop
```

III-11 Firewall applicatif

Jusqu'à présent, les utilisateurs de Windows® ont pu comparer le fonctionnent des techniques de Firewall de Linux avec ce qu'ils trouvent sur leur OS. Et bien que les fonctionnalités de Netfilter sont au moins équivalentes à ce que l'on trouve dans les outils de firewall sous Windows®, il reste une fonctionnalité qui manque à Linux : c'est le firewall applicatif.

On appelle "firewall applicatif" la capacité d'un logiciel à contrôler à la fois les flux de données entrantes et sortantes en fonction de certains critères (adresse IP, port, type de connexion, etc...), mais aussi des applications qui exploitent ces connexions. En gros, un firewall applicatif va permettre de contrôler quelles sont lesconnexions réseau que fait tel ou tel logiciel .

C'est typiquement ce que font des applications fonctionnant sous Windows®, tel que par exemple ZoneAlarm®, TinyFirewall®, Lock 'n' Stop®, etc... A chaque connexion que tente une application, ces firewall applicatifs vérifient que la connexion est autorisée, ou alerte l'utilisateur.

Mais à quoi sert exactement ce type d'application ? En fait, elles permettent à l'utilisateur de surveiller si telle ou telle application tente une connexion non autorisée , quasiment dans le dos de l'utilisateur. Par exemple lorsque Word® ou Excel® font une connexion Internet sans raison apparente, que le MediaPlayer® envoie une requête alors qu'il est en pleine lecture de vidéo, ou qu'une dll/ActiveX/autre utilise Internet Explorer® pour accéder à Internet. L'utilisateur de Windows® n'aurait donc pas confiance aux applications qu'il utilise ? Oui, c'est du moins ce qui ressort des discussions des utilisateurs de Windows® et de firewall applicatifs. Et pourquoi ne passent ils pas sous Linux alors ? Parce que ce type de fonctionnalité n'existe pas peut-être ? Que nenni, c'est faisable aussi sous Linux !

La mise en place d'un système de firewall applicatif se fait en fait en 2 temps :

- Premièrement, il faut charger le module du kernel appelé "ip_queue"

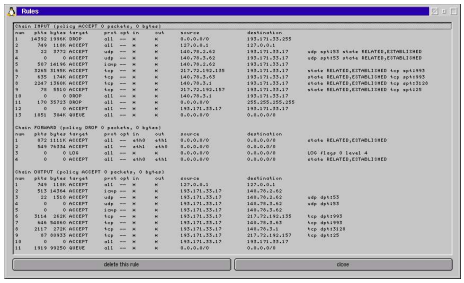
  ```
  [root@phoenix /]# modprobe ip_queue
  ```

- Puis dans les règles de Netfilter, par exemple au niveau des chaînes INPUT et OUTPUT de la table "Filter", il faut créer une règle qui envoie tout les paquets suspects vers la cible "QUEUE" (nous ne l'avons pas encore vue jusqu'ici). L'idéal est de la mettre en tant que dernière règle de la chaîne, afin que les autres règles aient déjà filtrés les paquets :

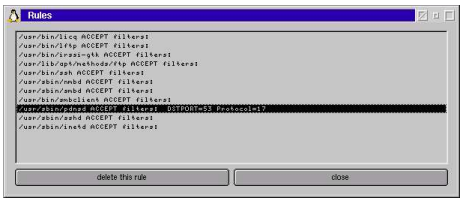
  ```
  [root@phoenix /]# iptable -t filter -A OUTPUT -j QUEUE
  [root@phoenix /]# iptable -t filter -A INPUT -j QUEUE
  ```

  Dans le jargon du kernel, on dit que l'on envoie le paquet dans le "user space" ("espace utilisateur" en français)
- Enfin, il faut avoir un applicatif qui va recevoir ces paquets, les analyser en fonction par exemple du programme qui a fait la connexion ou à qui elle est destinée, et refusé ou non le paquet. Une interface graphique peut par exemple montrer à l'utilisateur ce qui se passe, libre à ce dernier d'accepter ou non le paquet.

Pour l'instant, il n'existe que peu d'applications sous Linux qui fournissent de telles fonctionnalités "toutes faites". L'un d'entre eux est FireFlier. Comme le montre les captures d'écran ci-dessous, ce logiciel gère à la fois les règles Netfilter classiques, ainsi que les règles spécifiques aux applications :



Règles Netfilter



Règles de l'espace utilisateur

Vous pouvez évidement développer votre propre firewall applicatif. Pour cela, vous aurez besoin :

- de la LIBIPQ, dont vous trouverez des informations ici.
- éventuellement de ipqmpd qui permet de récolter les paquets IP venant de Netfilter. Vous pourrez le trouver ici
- de quelques neurones, afin de créer votre programme de gestion des paquets, et des règles d'entrée / sorties.

III-12 Bilan

Cette partie conclue ce (long !) chapitre sur le firewall de Linux qu'est Netfilter. Comme vous avez pu le voir, le mise en place d'un tel firewall nécessite l'apprentissage de quelques notions. Mais par la suite, l'implémentation est somme toute assez facile.

La vraie avancée de Netfilter sur d'autres solutions logicielles non libres est la technique du suivi de connexion, qui permet un réglage très fin des flux

de données entrant et sortant. Mais ne vous y fiez pas : croire que le "conntrack" et son principe de "trou noir", appelé aussi technique "stealth" ("furtif" en français), est la protection absolue en terme de sécurité est risquée, très très risqué même . En effet, même si il est très difficile pour un intrus de tester votre adresse IP, vous laissez vous-même des traces sur Internet. Le seul fait de lire en ligne cette documentation permet au serveur web qui la publie de déterminer quelle est votre adresse IP. Car forcément, il faut bien qu'il sache où vous envoyer la page HTML que vous avez demandé. De même, lorsque que vous êtes sur IRC ou que vous faites du "chat" en ligne ("discussion" en français), vous fournissez votre adresse IP. Enfin, lorsque vous envoyez un email, l'adresse IP de votre machine y apparaît la plupart du temps.

Pour toutes ces situations, un intrus qui a flairé votre présence connaîtra votre adresse IP, du moins le temps de votre connexion, et il pourra toujours tenter de se forer le passage à travers votre Netfilter. N'allez donc pas narguer des intrus en herbe sur des forums IRC de "l'undeground", sous prétexte que vous avez configuré le firewall de votre Linux ! .

De plus, gardez en tête que tout logiciel que vous utilisez, ce navigateur avec lequel vous lisez cette page, votre client IRC, votre application de mails, etc..., est une faille de sécurité potentielle. Donc une trame "officiellement" destinée à votre application, et qui à ce titre passe à travers votre Netfilter, peut s'avérer en fait être une malicieuse attaque de "buffer overflow", destinée à commettre le pire sur votre machine. Et ceci est encore plus vrai lorsqu'il s'agit d'un logiciel serveur que vous avez lancé sur votre machine, et dont vous voulez laisser l'accès à l'extérieur.

Pour conclure, les chapitres II et III que nous venons de voir vous permettrons de mieux sécuriser votre machine, et de cesser sa ressemblance à un gruyère. A vous maintenant de vous maintenir au courant des progrès en matière de sécurité, et bien sûr d'attaques, afin d'éviter qu'une souris ne s'y fasse un petit trou !

IV OUTILS ET LIENS

PLAN :

IV-1 Outils de configuration de firewalls Linux

IV-1-1 Interfaces à Iptables

Comme nous l'avons vu tout au long du précédent chapitre, la plus grosse partie de la configuration de Netfilter est d' utiliser la commande "iptables". Sa syntaxe peut rebuter quelque peu l'utilisateur néophyte, aussi existe t'il des interfaces graphiques qui permettent de créer assez simplement ces règles.

N'étant pas spécialement un adepte de ce style d'interface, et de plus ayant des besoins assez spécifiques, je ne me suis pas spécialement intéressé à celles-ci. Cependant, vous en trouverez vous-même de nombreuses sur le site Freshmeat.net, qui a pour vocation de référencer les applications libres.

J'en citerai quand même 2 très connues, et qui ont plutôt une bonne réputation :

- ShoreWall : Un programme de configuration de Netfilter / Iptables, possédant beaucoup d'options de configuration.
- FWBuilder : Un outil beaucoup plus générique, permettant de configurer facilement divers outils de firewall, comme "ipchaine" (l'ancêtre d'"iptables"), "iptables", certains firewalls matériels, etc...

IV-1-2 Distributions spécialisées

Linux est un système d'exploitation qui peut être utilisé pour un très grand nombre de fonctionnalités différentes. Les distributions qui nous intéresses spécialement dans le cadre de ce document sont celles constituées autour des fonctionnalités de firewall/routeur. Il s'agit en fait de distributions Linux très très simplifiées, et n'ayant qu'un seul but, celui de faire office de firewall et/ou de passerelle. On peut en trouver un grand nombre sur Internet, certaines plus ou moins libres, d'autres plus ou moins petites. Certaines peuvent même tenir sur une disquette 1,44Mo, et fonctionner sur une machine à base de 386 ou de 486 ! Transformer un ancienne machine en un puissant firewall, que voilà de la récupération intéressante et peu coûteuse !

En voici une petite liste, non exhaustive bien entendu !

- SME : Distribution avancée permettant de faire office de firewall, serveur de mails, serveur POP, serveur HTTP, etc...
- FloppyFW : Distribution basée sur une simple disquette, et faisant office de firewall.
- IPCop : Une distribution proposant des fonctions de firewall, DNS, serveur DHCP, etc...
- BBIagent : Similaire à "FloppyFW"
- Linux router : Un site qui propose (proposait...) un ensemble d'outils permettant de créer sa propre distribution routeur/firewall personnalisée.

Si vous voulez retrouver d'autres distributions Linux spécialisées, je vous conseille la lecture de ces sites, qui recensent les différentes distributions Linux, et leur fonctionnalités :

- LWN : Liste de distributions Linux, classées par genre. Les sections " Secured Distributions" et "Floppy-based" sont tout spécialement intéressantes.
- Linux Online : Autre site proposant les distributions, classées par type.
- DistroWatch : Un autre site de référence listant les distributions Linux.

IV-2 Un exemple : Netfilter_cfg

Parmi tous les outils de configuration d'Iptables que vous pouvez trouver, j'en propose un moi-même. Je sais ce que vous pouvez penser : "Ce type a fait un long document sur la sécurité Linux, et à la fin, une fois que l'on est bien endormis, hop il propose un vulgaire script qu'il a bricolé sur un coin de table ?" Et bien non, pas du tout ! Le propos n'est pas là. Il s'agit tout simplement de mettre un peu plus en applications tout ce que nous venons de voir jusqu'à présent.

Lorsque j'ai commencé à m'intéresser aux problèmes du firewall sous Linux, j'ai dût répondre à des besoins particuliers :

- Ma machine passe régulièrement d'une connexion Internet RTC à une connexion ADSL ou par passerelle, voir à pas de connexion Internet du tout. Il me fallait donc un script qui puisse s'adapter facilement à ces changements de configuration, sans que ce soit trop pénible pour moi.
- Étant de nature assez soupçonneuse vis à vis d'Internet, il me fallait utiliser des commandes "iptables" utilisant mon adresse IP Internet (comme vu précédemment). Donc il fallait que ce script trouve tout seul cette adresse IP.
- Enfin, il m'arrive de me connecter sur Internet, et de rendre temporairement accessible certains serveurs tournant sur ma machine. Je voulais donc un script qui puisse ouvrir le plus simplement possibles certains ports de ma machine.

De tout ces besoins différents est né " netfilter_cfg". C'est un honnête petit script de près de 1800 lignes écrit en bash. La syntaxe utilisée se veut

claire, le script est bien commenté, et il est régulièrement maintenu.

Bien entendu, il est distribué sous la licence GPL, ce qui vous permet de l'utiliser et de le distribuer librement, d'étudier et de modifier son code source. Vous pouvez aussi proposer vos propres patchs pour ce script, et même d'en faire une variante à votre nom (à la condition d'indiquer clairement quelle est la source du script originel).

Les points forts de Netfilter_cfg sont :

- Support de multiples interfaces locales : On peut déclarer autant d'interfaces réseaux internes ("eth0", "eth1", "eth2", etc...)
- Connexion externe : RTC / ADSL / passerelle : On peut utiliser tout ces types d'interfaces pour se connecter à Internet. La détection du type de connexion, ou son absence, est automatique.
- Supporte le Conntrack / IP masquerading : Seul le port forwarding n'est pas implémenté. Si nécessaire, je pourrais le développer par la suite.
- LOG / ULOG / Filtrage de logs : Ces différentes techniques de logs sont possible.
- Suppression de certains LOG : Il est possible de ne pas logger certains types de trames, afin d'éviter de surcharger les logs. C'est spécialement utile pour éliminer par exemple les demandes de connexions P2P. Elles sont notamment spécialement importantes en début de connexion.
- Gestion de serveurs : TO / FTP / ... : Ce script gère pour l'instant le passages des trames des serveurs de type Tactical Ops et FTP. Par la suite, d'autre serveurs pourrons être gérés.
- Affichage des règles : Afin de permettre à l'utilisateur de comprendre un peu mieux les règles Netfilter générées, toutes les commandes Iptables utilisées sont affichées ou stockés dans un log.
- Évolutif : De la manière dont ce script est implémenté, il se veut le plus évolutif possible. Vous pourrez rajouter facilement de nouvelles fonctionnalités, voir me proposer d'en rajouter.

Exemple d'utilisation :

```
[root@phoenix ]# /usr/local/sbin/netfilter_cfg
+ Utilisation des paramètres par défaut :
  --drop-rules' : on
  --spoofing-filter' : on
  --nat' : off
  --wan-ftp-server' : off
  --wan-to-server' : off
  --wan-ping' : off
  (Utiliser '--help pour avoir de l'aide')
+ Règles iptables modifiées. Utilisez 'iptables -L -n' ou 'iptables -L -n -v' pour afficher la liste des tables
+ Fichier de debug : /var/log/netfilter_cfg
```

Lancé sans paramètre, "netfilter_cfg" configure la sécurité au maximum. Pour la plupart des utilisations standards, il suffit de le lancer de cette manière au démarrage de votre machine, ou dès que votre connexion à Internet est établie. On peut difficilement faire plus simple, non ?

Pour avoir la liste de ses paramètres :

```
[root@phoenix /]# /usr/local/sbin/netfilter_cfg --help

                  netfilter_cfg v0.5.4

Usage: netfilter_cfg [--drop-rules <on|off>]
          [--nat <on|off>] [--spoofing-filter <on|off>] [--help]
          [--wan-ftp-server <on|off>] [--wan-to-server <on|off>]
          [--wan-ping <on|off>]

Options :
--drop-rules      Active/désactive le rejet automatique de certains paquets
                  Défaut : on
--nat             Active/désactive le fonctionnement en passerelle Internet
                  Défaut : off
--spoofing-filter Active/désactive les règles anti-spoofing
                  Défaut : on
--wan-ftp-server  Active/désactive le serveur FTP Internet
                  Défaut : off
--wan-to-server   Active/désactive le serveur Tactical Ops Internet
                  Défaut : off
--wan-ping        Autorise ou nom la machine à répondre aux ping
                  Défaut : off
--help            Affiche l'aide

Exemples :
  netfilter_cfg --nat on --wan-to-server on --wan-ftp-server off
La machine est configurée pour faire du NAT et serveur Tactical Ops.
Le serveur FTP est arrêté
```

Pour plus d'information sur ce programme, visitez la page officielle.

Pour votre information, j'utilise ce script à chacune de mes connexion Internet. Et j'ai commencé à le diffuser autour de moi, avec jusqu'à présent un grand succès ! Des néophytes en matière de Linux l'utilise d'ailleurs, sans pour l'instant rencontrer de problème.

IV-3 Liens

Cette section regroupe un certain nombre de liens sur Internet que j'ai utilisé pour mettre au point ce document, ou qui peuvent être des sources d'informations supplémentaires :

Cours / informations sur Netfilter

Langue	Site	Remarques
FR UK	Guides de Netfilter	C'est la documentation officielle de Rusty Russell, le créateur de Netfilter. La documentation est traduite en plusieurs langues.
FR	Christian Caleca	Un excellent guide sur Netfilter. Les autres sujets proposés sur ce site sont tout aussi bien fait.
FR	Iptables par l'exemple	Un autre tutoriel sur Netfilter
FR	Ce document	Un tutoriel sur le firewall et la sécurité sous Linux. Long à lire, mais il se veut le plus didactique et illustré possible.

Intrusion et tests de sécurité

Langue	Site	Remarque
UK	Nessus	Ensemble d'outils d'audit de sécurité
UK	SATAN	Même chose que Nessus, mais un plus ancien.
UK	PcFlank	Site de scan en ligne

	DSLreports	Site de scan en ligne
	Shields UP!	Site de scan en ligne
	AuditMyPc.com	Site de scan en ligne
	SecurityMetrics	Site de scan en ligne

Sites sur la sécurité

Langue	Site	Remarque
	Linux Security	Site sur la sécurité sous Linux
	Security Focus	Site sur la sécurité en général
	LWN.net security	Recense les alertes de sécurité sous Linux
	CERT	Recense les alertes de sécurité sur les différents OS
	Sécurite sur Debian FR	Cette page recense les alertes de sécurité sur la distribution Debian. Elle sont transposable sur les autres distributions Linux
	Sécurite sur Mandrake	Idem que plus haut, mais pour la distribution Mandrake.

CONCLUSION

Comme vous avez pu le constater, la sécurité d'une machine sous Linux n'est pas une mince affaire. Entre les démons à configurer proprement, et les règles de Netfilter à mettre en place, il y a de la matière à travailler. Ce n'est pas pour autant que l'on peu dire que Linux est moins sécurisé qu'un autre OS. Si j'avais fait par exemple un tel document sur la sécurité sous Windows®, il aurait été largement plus long, et nettement moins complet...

Que devez vous retenir d'un tel document ? Cela sera en fonction de vos désirs et besoins. Si vous estimez que la sécurité de votre machine n'a aucune importance, vous venez de perdre quelques heures à me lire et j'en suis navré. Mais ne venez pas vous plaindre si par la suite vous rencontrez des problèmes avec la sécurité de votre machine, et ne mettez pas cela sur le dos de Linux ! Si par contre vous vous sentez concerné par ces problèmes, il est temps pour vous d'appliquer au plus vite les informations vues ici. En urgence, vous pouvez commencer à utiliser "Netfilter_cfg", qui commencera à vous assurer un début de protection. Puis passez à la configuration de vos serveurs. Et enfin, revenez à la configuration de Netfilter, et voyez si vous pouvez ajuster plus finement vos règles à vos besoins.

Malgré sa longueur, ce document n'a pas traité tous les aspects de la sécurité sous Linux (d'ailleurs, est-ce faisable ?). Car même avec une machine "blindée", il vous faudra mettre à jour régulièrement votre système, lancer de temps en temps des tests d'intrusion, surveiller le contenu de vos disques durs, etc... Disons qu'avec ce que vous avez appris là, vous avez moins à craindre lorsque votre machine est connectée à Internet.

Par contre, l'aspect confidentialité de l'Internaute n'a pas du tout été abordé, car ce n'était pas l'intérêt principal de ce guide. Qu'en est t'il de l'utilisation des cookies, des serveurs de publicités, de l'encryptage de pages en HTTPS, etc... Cela fera peut-être (sans doute ?) l'objet d'un autre guide !

Bien, il ne me reste plus qu'à clore ce chapitre et de ranger mon Xemacs au placard (pour quelques minutes, pas plus !). L'écriture de ce document et la conférence qui y est associée ont demandé plus de 15 jours de travail, dont la moitié à raison de plus de 16h par jours... Ce fut un travail réellement intéressant et motivant, et j'espère qu'il vous sera utile. J'ai cherché à le faire le plus simple possible, voir même un peu trop parfois, afin que les utilisateurs les plus débutants ne soient pas perdus. J'espère cependant que les utilisateurs les plus avancés de Linux y aurons quand même trouvé des sources d'informations, et qu'ils n'estiment pas avoir perdu du temps à le lire.

Si vous avez des remarques à faire sur ce guide, des corrections à proposer ou des points à éclaircir, n'hésitez pas à me contacter (olivieraj@free.fr). Une traduction de ce guide dans une autre langue n'est pas prévue, mais si vous vous en sentez le courage, n'hésitez pas à me le faire savoir !

REMERCIEMENTS

Merci tout d'abord à la Guilde, qui m'a permis de présenter une conférence sur ce sujet le 2 juillet 2003. Cet événement fut le moteur de la création de ce document. Elle a été un franc succès, puisqu'elle a réunit environ 75 personnes dans l'amphithéâtre de l' ENSIMAG. Merci tout spécialement à Jérôme pour l'organisation de la conférence, et à Edgar pour son CSS. Et évidement, merci à son Tux pour m'avoir tenu compagnie durant cette présentation !

Merci aux personnes qui m'ont aidé à la préparation de ce document, et tout particulièrement à Keyvan pour les corrections. Merci aussi bien sur aux nombreux lecteurs qui ont apportés leurs avis, corrections, et impressions.

Merci surtout à Rusty Russell, le créateur de Netfilter et les outils qui y sont associés. Même si il est modeste sur sa création, la plus-value que cela apporte à Linux est énorme.

Merci enfin à toute la communauté Libre qui font que Linux existe. Merci à toutes les personnes qui apportent leur pierre à l'édifice du Libre, et qui le rende tout les jours plus facile et plus fiable à utiliser.

VERSIONS DE CE DOCUMENT

Version	Date	Remarque
0.5.3	23 juillet 2003	Correction majeur sur l'explication du traceroute Diverses autres petites corrections
0.5.2	20 juillet 2003	Première version publique

LICENSE

- la transmission de ce document à un tiers se fait avec transmission de cette licence, sans modification, et en particulier sans addition de clause ou contrainte nouvelle, explicite ou implicite, liée ou non à cette transmission. En particulier, en cas d'inclusion dans une base de données ou une collection, le propriétaire ou l'exploitant de la base ou de la collection s'interdit tout droit de regard lié à ce stockage et concernant l'utilisation qui pourrait être faite du document après extraction de la base ou de la collection, seul ou en relation avec d'autres documents.

Toute incompatibilité des clauses ci-dessus avec des dispositions ou contraintes légales, contractuelles ou judiciaires implique une limitation correspondante du droit de lecture, utilisation ou redistribution verbatim ou modifiée du document.

Free Document Dissemination Licence -- FDDL version 1

This document may be freely read, stored, reproduced, disseminated, translated or quoted by any means and on any medium provided the following conditions are met:

- every reader or user of this document acknowledges that he his aware that no guarantee is given regarding its contents, on any account, and specifically concerning veracity, accuracy and fitness for any purpose;
- no modification is made other than cosmetic, change of representation format, translation, correction of obvious syntactic errors, or as permitted by the clauses below;
- comments and other additions may be inserted, provided they clearly appear as such; translations or fragments must clearly refer to an original complete version, preferably one that is easily accessed whenever possible;
- translations, comments and other additions must be dated and their author(s) must be identifiable (possibly via an alias);
- this licence is preserved and applies to the whole document with modifications and additions (except for brief quotes), independently of the representation format;
- whatever the mode of storage, reproduction or dissemination, anyone able to access a digitized version of this document must be able to make a digitized copy in a format directly usable, and if possible editable, according to accepted, and publicly documented, public standards;
- redistributing this document to a third party requires simultaneous redistribution of this licence, without modification, and in particular without any further condition or restriction, expressed or implied, related or not to this redistribution. In particular, in case of inclusion in a database or collection, the owner or the manager of the database or the collection renounces any right related to this inclusion and concerning the possible uses of the document after extraction from the database or the collection, whether alone or in relation with other documents.

Any incompatibility of the above clauses with legal, contractual or judiciary decisions or constraints implies a corresponding limitation of reading, usage, or redistribution rights for this document, verbatim or modified.

Olivier Allard-Jacquin

Site de référence : http://olivieraj.free.fr/

Dernière modification : le mercredi 23 juillet 2003 à 01:31:50