Les Firewall

1 – Les différents types de filtrages

2.1 – Le filtrage simple de paquet (Stateless)

2.1.1 – Le principe

C'est la méthode de filtrage la plus simple, elle opère au niveau de la couche réseau et transport du modèle Osi. La plupart des routeurs d'aujourd'hui permettent d'effectuer du filtrage simple de paquet. Cela consiste à accorder ou refuser le passage de paquet d'un réseau à un autre en se basant sur :

- L'adresse IP Source/Destination.
- Le numéro de port Source/Destination.
- Et bien sûr le protocole de niveau 3 ou 4.

Cela nécessite de configurer le Firewall ou le routeur par des règles de filtrages, généralement appelées des ACL (Access Control Lists).

2.1.2 – Les limites

Le premier problème vient du fait que l'administrateur réseau est rapidement contraint à autoriser un trop grand nombre d'accès, pour que le Firewall offre une réelle protection. Par exemple, pour autoriser les connexions à Internet à partir du réseau privé, l'administrateur devra accepter toutes les connexions Tcp provenant de l'Internet avec un port supérieur à 1024. Ce qui laisse beaucoup de choix à un éventuel pirate.

Il est à noter que de définir des ACL sur des routeurs haut de gamme – c'est à dire, supportant un débit important – n'est pas sans répercussion sur le débit lui-même. Enfin, ce type de filtrage ne résiste pas à certaines attaques de type IP Spoofing / IP Flooding, la mutilation de paquet, ou encore certaines attaques de type DoS. Ceci est vrai sauf dans le cadre des routeurs fonctionnant en mode distribué. Ceci permettant de gérer les Acl directement sur les interfaces sans remonter à la carte de traitement central. Les performances impactées par les Acl sont alors quasi nulles.

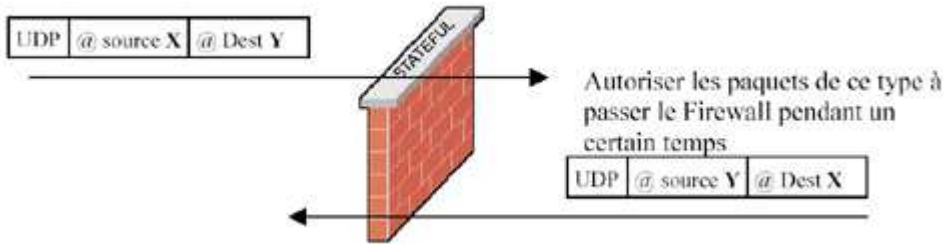
2.2 – Le filtrage de paquet avec état (Stateful)

2.2.1 – Le Principe

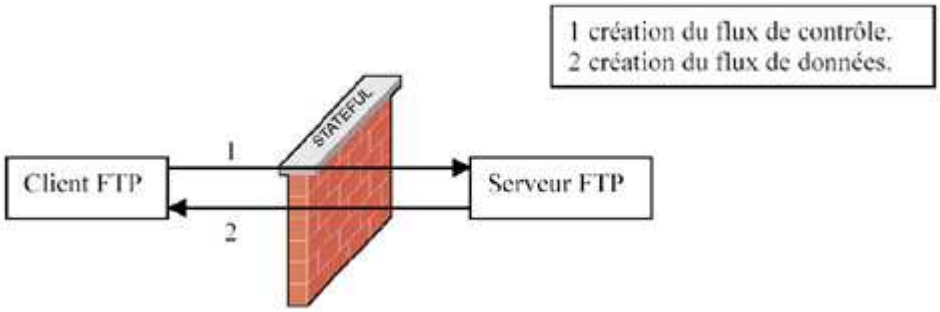
L'amélioration par rapport au filtrage simple, est la conservation de la trace des sessions et des connexions dans des tables d'états internes au Firewall. Le Firewall prend alors ses décisions en fonction des états de connexions, et peut réagir dans le cas de situations

protocolaires anormales. Ce filtrage permet aussi de se protéger face à certains types d'attaques DoS.

Dans l'exemple précédent sur les connexions Internet, on va autoriser l'établissement des connexions à la demande, ce qui signifie que l'on aura plus besoin de garder tous les ports supérieurs à 1024 ouverts. Pour les protocoles UDP et ICMP, il n'y a pas de mode connecté. La solution consiste à autoriser pendant un certain délai les réponses légitimes aux paquets envoyés. Les paquets Icmp sont normalement bloqués par le Firewall, qui doit en garder les traces. Cependant, il n'est pas nécessaire de bloquer les paquets Icmp de type 3 (destination inaccessible) et 4 (ralentissement de la source) qui ne sont pas utilisables par un attaquant. On peut donc choisir de les laisser passer, suite à l'échec d'une connexion Tcp ou après l'envoi d'un paquet Udp.



Pour le protocole Ftp (et les protocoles fonctionnant de la même façon), c'est plus délicat puisqu'il va falloir gérer l'état de deux connexions. En effet, le protocole Ftp, gère un canal de contrôle établi par le client, et un canal de données établi par le serveur. Le Firewall devra donc laisser passer le flux de données établi par le serveur. Ce qui implique que le Firewall connaisse le protocole Ftp, et tous les protocoles fonctionnant sur le même principe. Cette technique est connue sous le nom de filtrage dynamique (Stateful Inspection) et a été inventée par Checkpoint. Mais cette technique est maintenant gérée par d'autres fabricants.



2.2.2 – Les limites

Tout d'abord, il convient de s'assurer que les deux techniques sont bien implémentées par les Firewalls, car certains constructeurs ne l'implémentent pas toujours correctement. Ensuite une fois que l'accès à un service a été autorisé, il n'y a aucun contrôle effectué sur les requêtes et réponses des clients et serveurs. Un serveur Http pourra donc être attaqué impunément (Comme quoi il leur en arrive des choses aux serveurs WEB !). Enfin les protocoles maisons

utilisant plusieurs flux de données ne passeront pas, puisque le système de filtrage dynamique n'aura pas connaissance du protocole.

2.3 – Le filtrage applicatif (ou pare-feu de type proxy ou proxying applicatif)

2.3.1 – Le principe

Le filtrage applicatif est comme son nom l'indique réalisé au niveau de la couche Application. Pour cela, il faut bien sûr pouvoir extraire les données du protocole de niveau 7 pour les étudier. Les requêtes sont traitées par des processus dédiés, par exemple une requête de type Http sera filtrée par un processus proxy Http. Le pare-feu rejettera toutes les requêtes qui ne sont pas conformes aux spécifications du protocole. Cela implique que le pare-feu proxy connaisse toutes les règles protocolaires des protocoles qu'il doit filtrer.

2.3.2 – Les limites

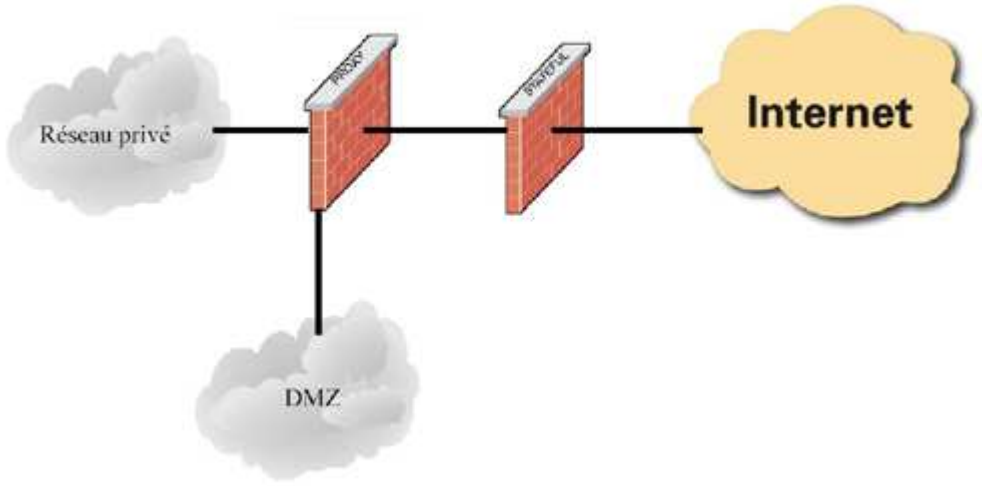
Le premier problème qui se pose est la finesse du filtrage réalisé par le proxy. Il est extrêmement difficile de pouvoir réaliser un filtrage qui ne laisse rien passer, vu le nombre de protocoles de niveau 7. En outre le fait de devoir connaître les règles protocolaires de chaque protocole filtré pose des problèmes d'adaptabilité à de nouveaux protocoles ou des protocoles maisons.

Mais il est indéniable que le filtrage applicatif apporte plus de sécurité que le filtrage de paquet avec état, mais cela se paie en performance. Ce qui exclut l'utilisation d'une technologie 100 % proxy pour les réseaux à gros trafic au jour d'aujourd'hui. Néanmoins d'ici quelques années, le problème technologique sera sans doute résolu.

2.4 – Que choisir

Tout d'abord, il faut nuancer la supériorité du filtrage applicatif par rapport à la technologie Stateful. En effet les proxys doivent être paramétrés suffisamment finement pour limiter le champ d'action des attaquants, ce qui nécessite une très bonne connaissance des protocoles autorisés à traverser le firewall. Ensuite un proxy est plus susceptible de présenter une faille de sécurité permettant à un pirate d'en prendre le contrôle, et de lui donner un accès sans restriction à tout le système d'information.

Idéalement, il faut protéger le proxy par un Firewall de type Stateful Inspection. Il vaut mieux éviter d'installer les deux types de filtrage sur le même Firewall, car la compromission de l'un entraîne la compromission de l'autre. Enfin cette technique permet également de se protéger contre l'ARP spoofing.



3 – Les différents types de firewall

3.1 – Les firewall bridge

Ces derniers sont relativement répandus. Ils agissent comme de vrais câbles réseau avec la fonction de filtrage en plus, d'où leur appellation de firewall. Leurs interfaces ne possèdent pas d'adresse Ip, et ne font que transférer les paquets d'une interface a une autre en leur appliquant les règles prédéfinies. Cette absence est particulièrement utile, car cela signifie que le firewall est indétectable pour un hacker lambda. En effet, quand une requête ARP est émise sur le câble réseau, le firewall ne répondra jamais. Ses adresses Mac ne circuleront jamais sur le réseau, et comme il ne fait que « transmettre » les paquets, il sera totalement invisible sur le réseau. Cela rend impossible toute attaque dirigée directement contre le firewall, étant donné qu'aucun paquet ne sera traité par ce dernier comme étant sa propre destination. Donc, la seule façon de le contourner est de passer outre ses règles de drop. Toute attaque devra donc « faire » avec ses règles, et essayer de les contourner.

Dans la plupart des cas, ces derniers ont une interface de configuration séparée. Un câble vient se brancher sur une troisième interface, série ou même Ethernet, et qui ne doit être utilisée que ponctuellement et dans un environnement sécurisé de préférence.

Ces firewalls se trouvent typiquement sur les switchs.

3.1.1 – Avantages

- Impossible de l'éviter (les paquets passeront par ses interfaces)
- Peu coûteux

3.1.2 – Inconvénients

- Possibilité de le contourner (il suffit de passer outre ses règles)
- Configuration souvent contraignante
- Les fonctionnalités présentes sont très basiques (filtrage sur adresse IP, port, le plus souvent en Stateless).

3.2 – Les firewalls matériels

Ils se trouvent souvent sur des routeurs achetés dans le commerce par de grands constructeurs comme Cisco ou Nortel. Intégrés directement dans la machine, ils font office de « boite noire », et ont une intégration parfaite avec le matériel. Leur configuration est souvent relativement ardue, mais leur avantage est que leur interaction avec les autres fonctionnalités du routeur est simplifiée de par leur présence sur le même équipement réseau. Souvent relativement peu flexibles en terme de configuration, ils sont aussi peu vulnérables aux attaques, car présent dans la « boite noire » qu'est le routeur. De plus, étant souvent très liés au matériel, l'accès à leur code est assez difficile, et le constructeur a eu toute latitude pour produire des systèmes de codes « signés » afin d'authentifier le logiciel (système RSA ou assimilés). Ce système n'est implanté que dans les firewalls haut de gamme, car cela évite un remplacement du logiciel par un autre non produit par le fabricant, ou toute modification de ce dernier, rendant ainsi le firewall très sûr. Son administration est souvent plus aisée que les firewalls bridges, les grandes marques de routeurs utilisant cet argument comme argument de vente. Leur niveau de sécurité est de plus très bon, sauf découverte de faille éventuelle comme tout firewall. Néanmoins, il faut savoir que l'on est totalement dépendant du constructeur du matériel pour cette mise à jour, ce qui peut être, dans certains cas, assez contraignant. Enfin, seules les spécificités prévues par le constructeur du matériel sont implémentées. Cette dépendance induit que si une possibilité nous intéresse sur un firewall d'une autre marque, son utilisation est impossible. Il faut donc bien déterminer à l'avance ses besoins et choisir le constructeur du routeur avec soin.

3.2.1 – Avantages

- Intégré au matériel réseau
- Administration relativement simple
- Bon niveau de sécurité

3.2.2 – Inconvénients

- Dépendant du constructeur pour les mises à jour
- Souvent peu flexibles.

3.3 – Les firewalls logiciels

Présents à la fois dans les serveurs et les routeurs « faits maison », on peut les classer en plusieurs catégories :

3.3.1 – Les firewalls personnels

Ils sont assez souvent commerciaux et ont pour but de sécuriser un ordinateur particulier, et non pas un groupe d'ordinateurs. Souvent payants, ils peuvent être contraignants et quelque fois très peu sécurisés. En effet, ils s'orientent plus vers la simplicité d'utilisation plutôt que vers l'exhaustivité, afin de rester accessible à l'utilisateur final.

3.3.1.1 Avantages

- Sécurité en bout de chaîne (le poste client)
- Personnalisable assez facilement

3.3.1.2 – Inconvénients

- Facilement contournable
- Difficiles à départager de par leur nombre énorme.

3.3.2 – Les firewalls plus « sérieux »

Tournant généralement sous linux, car cet OS offre une sécurité réseau plus élevée et un contrôle plus adéquat, ils ont généralement pour but d'avoir le même comportement que les firewalls matériels des routeurs, à ceci près qu'ils sont configurables à la main. Le plus courant est iptables (anciennement ipchains), qui utilise directement le noyau linux. Toute fonctionnalité des firewalls de routeurs est potentiellement réalisable sur une telle plateforme.

3.3.2.1 Avantages

- Personnalisables
- Niveau de sécurité très bon

3.3.2.2 – Inconvénients

- Nécessite une administration système supplémentaire

Ces firewalls logiciels ont néanmoins une grande faille : ils n'utilisent pas la couche bas réseau. Il suffit donc de passer outre le noyau en ce qui concerne la récupération de ces paquets, en utilisant une librairie spéciale, pour récupérer les paquets qui auraient été normalement « droppés » par le firewall. Néanmoins, cette faille induit de s'introduire sur l'ordinateur en question pour y faire des modifications… chose qui induit déjà une intrusion dans le réseau, ou une prise de contrôle physique de l'ordinateur, ce qui est déjà Synonyme d'inefficacité de la part du firewall.

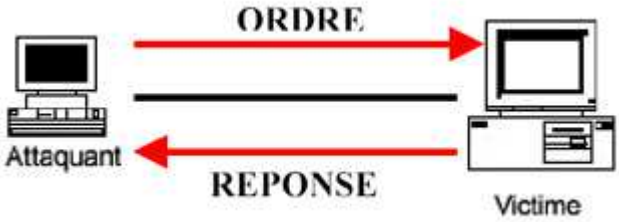
4 – Attaques, outils, défenses

4.1 – Scénarios d'attaques (Pénétrations de réseaux)

Qu'est-ce qu'une backdoor ? Une backdoor est un accès (« caché ») sur votre système qui permet à un pirate d'en prendre le contrôle à distance. Il existe une multitude de sortes de backdoor, et en général dans ce domaine, l'imagination des pirates rivalise avec l'incrédulité des utilisateurs. Voici quelques scénarios d'attaques plus ou moins classique.

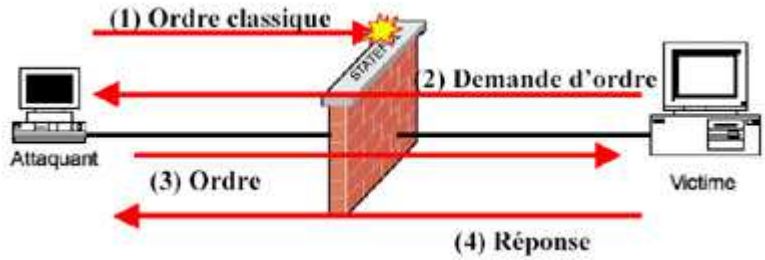
4.1.1 – Premier cas : Pas de protection

Considérons un ordinateur victime sur lequel on a installé une backdoor en exploitant une des failles du système. L'attaquant a alors la possibilité d'utiliser tous les services présents sur cet ordinateur. Il lui suffit d'envoyer ses ordres à la backdoor et de récupérer les réponses.



4.1.2 – Deuxième cas : Filtrer les flux entrants illégaux

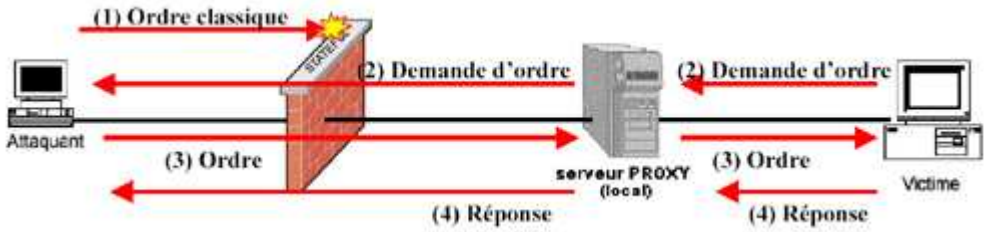
La sécurité de notre système ne nous semblant pas infaillible, nous décidons alors d'installer un Firewall avec états (Un Firewall sans état nous semblant quelque peu légers). Le trafic entrant est maintenant stoppé comme il se doit. Malheureusement, le pirate étant rusé et malicieux, il a pris soin de s'arranger pour que sa backdoor initie elle-même les sessions. Du coup le Firewall laisse passer les requêtes de l'attaquant qui sont considérées comme des réponses par celui-ci.



4.1.3 – Troisième cas : Bloquer les flux entrants et sortants

Dans le cas précédent, le problème était dû aux flux sortants qui permettaient au cheval de Troie d'initier les sessions avec la machine de l'attaquant. Il s'agit donc de bloquer les flux sortants. Pour cela la défense insère donc un proxy afin de contrôler ce qui sort du réseau. Malheureusement le trojan peut encore sortir, certes avec plus de difficultés puisqu'il devra se renseigner sur les flux autorisés à sortir par le proxy, et les utiliser pour passer le proxy. Par exemple on peut encapsuler des ordres dans du HTTP (Ip over Http),dans du SSL (Ip over Ssl), DNS (Ip over Dns), Smtp (Ip over Smtp).

Dans le cas d'un proxy avec authentification, le pirate fera preuve de grande imagination en réalisant un trojan capable de profiter des applications (comme IE par exemple) qui une fois qu'elles se sont authentifiées sont utilisées pour passer le proxy.



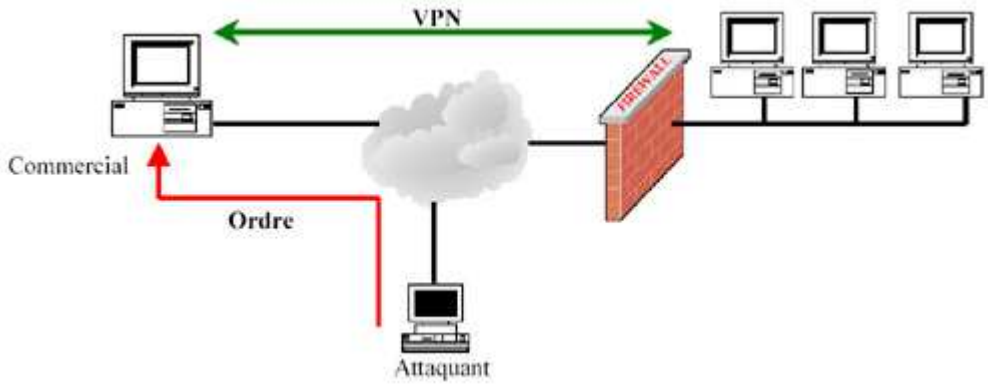
4.1.4 – Quatrième cas : Protection locale via un Firewall personnel

L'idée du Firewall personnel est de surveiller le trafic entrant et sortant de la machine infectée. Malheureusement ceux-ci sont fortement attaquables, on peut :

- Passer au-dessus du Firewall via une application autorisée. ® Appel de CreateRemoteThread permettant de l'injection de code à la volée sous Windows.
- Passer en dessous du Firewall via une bibliothèque adaptée (Winpcap sous Windows ou pcap sous Linux).
- Attaquer le Firewall lui-même en tant qu'applicatif (Arrêt du processus).

4.1.5 – Cinquième cas : piratage de VPN

Imaginons qu'un pirate mal intentionné installe un trojan sur l'ordinateur d'un commercial. L'ordinateur possédant entre autre un système Windows et un client VPN. Normalement, le client VPN fonctionne de telle sorte que toutes les liaisons réseaux n'étant pas dans le VPN ne fonctionnent pas. Malheureusement, il s'agit du même problème que pour les Firewall personnels et le trojan permet à l'attaquant d'accéder au réseau de l'entreprise via le VPN.



On peut se rendre compte de l'importance de la robustesse du système d'exploitation, sur lequel est installé le VPN. Après une attaque réussie et la prise de contrôle en root de la machine par le pirate, celui-ci va essayer d'installer ce que l'on appelle un rootkit.

Il existe deux types de rootkit, les rootkit simples et les rootkit noyaux. Les rootkit simples se contentent de remplacer toute une collection de programmes système (ps, netstat, ifconfig, ….) lui permettant d'effacer les traces de son passage et ainsi masquer complètement sa backdoor. Un simple outils tel que Tripwire permet de vérifier l'intégrité des commandes, en enregistrant de façon cryptée les signatures (générées par une fonction de hashage) des fichiers de commandes en question.

Les rootkit noyaux sont beaucoup plus difficiles à détecter, puisqu'ils modifient le noyau et donc modifient son comportement. Par exemple rendre invisible un processus à chaque appel (non pas de la commande ps) mais des fonctions systèmes du noyau, qui elles même sont appelées par la commande ps. Evidemment un rootkit noyau modifie en plus les routines de journalisassions du système, masque les connexions réseaux, … Pour se protéger des rootkit noyaux, le plus simple est de s'en prémunir. Pour cela, l'idéal est de patcher son noyau pour empêcher l'installation d'un rootkit, et de désactiver les LKM (Loadable Kernel Modules qui permettent au root d'introduire un nouveau code dans le système d'exploitation pendant que ce dernier est en cours d'exécution), malheureusement cela ne suffit pas toujours. Il existe un

outil pour Linux capable d'un certain nombre de vérification de modules de backdoor. Cet outil s'appelle rkscan et permet de détecter les versions de rootkits les plus populaires.

4.2 – Les techniques et outils de découvertes de Firewall

Il existe beaucoup d'outils et beaucoup de techniques permettant d'identifier un Firewall. L'objectif de ce paragraphe est d'en exposer quelques-uns et quelques-unes. Il est évident que la plupart des outils utilisés par les pirates pour découvrir les Firewall sont utilisables pour une activité tout aussi louable telle que la vérification du bon fonctionnement du firewall et de la robustesse du réseau.

Dans un premier temps il convient de localiser le ou les Firewalls. La localisation du firewall ne pose pas de gros problèmes, un simple traceroute (ou tracert.exe) suffira, bien que dans certains cas netcat apporte de meilleurs résultats.

Exemple :

```
C:> nc –v –n 10.10.1.8 25
(UNKNOWN) [10.10.1.8] 25 (?) open
421 10.10.1.8 Sorry, the firewall does not provide mail service to you.
```

Ensuite l'attaquant cherchera à identifier le Firewall, soit en espérant exploiter une faille même du Firewall, soit il cherchera à identifier les règles du Firewall afin d'y détecter une faille dans le filtrage de paquet. Pour identifier les règles d'un Firewall, il faut utiliser un scanner de port. Il existe de nombreux scanner de ports, les plus connus sont Firewalk, Nmap et Hping2 :

4.2.1 – Firewalk

Le Firewalking est une technique qui permet de déterminer les règles de filtrages de niveau 4 (Transport) sur les équipements (routeurs, Firewall, passerelles) qui acheminent des paquets de niveau 3 (Réseau).
Le principe de cette technique repose sur le champ Ttl (Time To Live) des entêtes IP des paquets. C'est à dire le nombre d'équipement (routeur) que peut traverser le paquet. Le logiciel traceroute utilise aussi la technique du Ttl. Lorsque l'on envoie un paquet Udp avec un Ttl de 1, le premier routeur recevant le paquet émettra un paquet Icmp Ttl-exceeded. Et l'on répète le procédé en augmentant le Ttl de 1 à chaque fois. En fait ce procédé peut être réalisé avec d'autres protocoles de niveau 4 comme Tcp ou de niveau 3 comme Icmp.

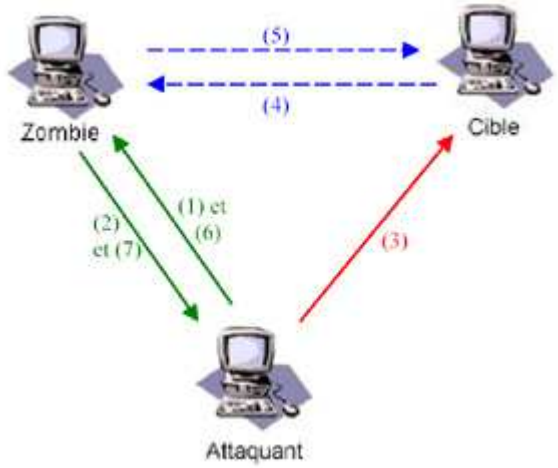
Firewalk fonctionne en construisant des paquets avec un IP Ttl calculé de façon à expirer sur un segment situé après le firewall. En fait si le paquet est autorisé par le firewall, il pourra le passer et expirera comme prévu en envoyant un message « Icmp Ttl expired in transit ». A l'inverse, si le paquet est bloqué par l'ACL du firewall, il sera abandonné et aucune réponse ne sera envoyée ou bien un paquet de filtre admin Icmp de type 13 sera envoyé.

Il est possible de bloquer les paquets Icmp Ttl EXPIRED au niveau de l'interface externe, mais le problème est que ses performances risquent d'en prendre un sérieux coup car des clients se connectant légitimement ne sauront jamais ce qui est arrivé à leur connexion.

4.2.2 – Nmap

Nmap (Network Mapper) est certainement le scanner de port le plus célèbre disponible sous linux, Windows et même MAC. En règle générale, Nmap vérifie que l'hôte à scanner est connecté au réseau. Il réalise pour cela à la fois un Tcp ping sur le port 80 et un ping Icmp normal. Ce comportement peut être détecté par un IDS (Inspection Detection System) et pour cela on peut changer le comportement de Nmap. Nmap permet d'effectuer différents types de scans, en voici les exemples principaux :

- Tcp connect : option -sT Principe : une connexion Tcp habituelle est tentée sur chaque port. Inconvénient, ce genre de scan est visible dans les logs des firewalls. Les noms de services sont associés aux ports ouverts par le fichier Nmap-services et non /etc/services. Il n'est donc pas exclu que le service désigné soit faux. Avantage, possibilité de déterminer l'utilisateur sous lequel est lancé un démon via Ident.
- Syn scan : option -sS Principe : Seul un paquet Syn est envoyé. Si le port est ouvert, un Syn|ACK est renvoyé, sinon un RST est renvoyé. En cas de port ouvert Nmap renvoie un paquet RST pour fermer la connexion immédiatement. Avantages, rapide et moins détectable. Fait une différence entre les ports filtrés et ouverts. Inconvénient, impossibilité de déterminer l'utilisateur via Ident.
- IDLE scan : option -sI @zombie:port zombie Principe : Une machine « zombie » permet de masquer la source du scan. Avantages, quasiment impossible à tracer, permet de déterminer les règles du firewall à partir du zombie plutôt que de la machine qui initie le scan. Inconvénients, pas de prise d'empreinte d'OS, ni d'utilisation de Ident.



(0) S'assurer que la machine zombie n'est pas trop chargée pour permettre de bien mesurer l'incrémentation (un petit calibrage avant le scan est donc nécessaire en envoyant une dizaine de paquets Syn).
(1) et (6) Envoi d'un paquet Syn|ACK au zombie pour récupérer l'IP ID (le numéro de séquence).
(2) et (7) Réponse à (1) et (6) par un paquet RST contenant l'IP ID du zombie.
(3) Envoi d'un paquet Syn à la cible avec comme adresse source celle du zombie.
(4) Si le port est ouvert, la cible répond en envoyant un paquet Syn | ACK au zombie, sinon par un paquet RST.
(5) Le zombie incrémente son IP ID, et répond à la cible en envoyant un paquet RST à la cible.
(8) Il est aussi conseillé de tester plusieurs fois chaque port pour éviter les faux-positifs.

- FIN, XMAS et NULL scans : options -sF, -sX et -sN Principe : envoi respectivement de paquet FIN, de paquet FIN| URG |PSH et de paquet sans aucun flag Tcp activé.

Avantages, contournement de certains Firewalls. Inconvénient, ne fonctionne pas contre certains OS qui n'appliquent pas les normes à la lettre (comme Windows, IRIX, …)

- SCAN Udp : option -sU Principe : Envoi d'un paquet Udp vide sur chaque port. Les ports fermés retournent un paquet Icmp port unreachable. Avantage, permet d'avoir des informations sur NFS, TFtp et certaines backdoor. Inconvénient, très lent.
- Scan RPC (Remote Procedure Call) : option -sR Principe : envoi de commandes SunRPC. Avantage, détermine s'il s'agit bel et bien de port RPC et si oui, de quel programme et/ou version il s'agit. Inconvénient, assez voyant.
- ACK et Window scan : option -sA et sW principe : Envoie respectivement un paquet ACK avec un numéro de séquence aléatoire et un paquet ayant une taille de fenêtre non valide. Les ports qui ne répondent pas par un RST sont filtrés. Avantage, permet de vérifier si un port est filtré et s'il l'est avec une règle stateful. Inconvénient, Le Window scan ne fonctionne pas sur tous les systèmes d'exploitation.
- Scans personnalisés : option –scanflags L'utilisateur spécifie les flags Tcp qu'il souhaite activer.
- Scan de protocoles : option -sO Permet de déterminer quels sont les protocoles supportés par le système scanné, généralement un routeur.

Il n'est pas difficile de scanner dans l'anonymat :

- Changer la vitesse du scan, grâce à l'option -T suivie des arguments Paranoïd, Sneaky, Polite, Normal, Aggressive or Insane. Nmap attend 5 minutes entre chaque paquet envoyé en mode Paranoïd et 15 secondes en mode Sneaky.
- Utiliser des leurres (decoys, option -D) en envoyant d'autres paquets IP spoofés et semblant provenir d'une autre adresse. Sur un réseau local on peut spoofer (option -S) l'adresse source, et récupérer les réponses par sniffage en mode promiscuous. Sur la technique de l'IDLE scan.

L'auteur de Nmap, Fyodor, décrit sa technique de prise d'empreinte du système d'exploitation (fingerprinting) dans l'article « Détection d'OS distante par prise d'empreinte de pile Tcp/IP » traduit en français par ArHuman, et l'original en anglais « Remote OS detection via Tcp/IP Stack FingerPrinting ». Le principe repose sur le test de différentes particularités des piles Tcp/IP des différents systèmes d'exploitation, lesquelles sont :

- Le type d'incrémentation du numéro de séquence initiale (ISN).
- La présence ou non du flag IP Don't Fragment.
- Des tests sur les tailles de fenêtres.
- Le type de service (ToS).
- Différents tests sur des paquets TCP.
- Différents tests au niveau Icmp, comme la limite de vitesse d'envoi de certains types de paquets Icmp d'erreurs.

Nmap utilise l'option -O pour effectuer une prise d'empreinte du système, et en général on associe l'option -v (Verbose) pour obtenir des informations supplémentaires comme les ports servant à effectuer les tests de la prise d'empreinte.

Il m'a semblé important de remarquer le fait que de scanner un Firewall ou un système protégé par un Firewall prend plus de temps qu'un système non protégé.

4.2.3 – HPING2

HPING2 est différent de Nmap d'abord parce qu'il est beaucoup plus configurable. On peut facilement modifier n'importe quel octet de l'entête TcpIP. Cela permet d'être réellement créatif au niveau des techniques de balayage à des fins de reconnaissance. On peut bien sûr insérer des données malveillantes dans les paquets (buffer overflow, trojan, …) et les utiliser pour pénétrer des réseaux.
HPING2 permet de :

- tester les règles de Firewall,
- de faire du scan sophistiqué,
- de tester les performances d'un réseau utilisant différents protocoles, le ToS et la fragmentation,
- de faire du firewalk,
- de l'empreinte de système d'exploitation.

Actuellement, la version HPING3 est en train d'être développé par Salvatore SanFilippo, et d'autres volontaires. Selon son site Web, Hping3 sera nettement supérieur à l'actuelle version. Il y aura des améliorations d'installation, des outputs plus lisibles, et sera exploitable par script. Le statut actuel du projet peut être suivi ici.

Il existe aussi des outils d'évaluation de vulnérabilité :

- Les payants : Retina (eEye) , NetRecon (Symantec), ISS Internet Scanner (ISS), Cybercop Scanner (Network Associates).
- Les freeware : Nessus (Renaud Deraison).

4.2.4 – NESSUS

Nessus a été écrit par Renaud Deraison (depuis début 1998), un grand maître de Linux et de la sécurité. Nessus est devenu le Linux de l'évaluation de la vulnérabilité, il surpasse certains de ses concurrents commerciaux. Nessus emploie un modèle de plug-in extensible qui permet à la sécurité d'ajouter à la demande des modules d'exploration. Nessus utilise un langage de script NASL (Nessus Attack Script Language) pour écrire des tests de sécurité rapidement et facilement. Nessus est basé sur une architecture client-Serveur, le serveur réalise les attaques et tourne sur un système UNIX; les clients initialisent les attaques et existent sur différentes plates-formes X11, Win32 et un client Java. Le fait d'être open source et d'utiliser des plug-in permet à Nessus d'être mis à jour régulièrement au niveau de sa base de données d'attaques et d'être à la pointe de la technologie. Nessus compte à l'heure actuel plus de 500 contrôles de vulnérabilités, dont certains ne sont pas disponibles dans les scanners commerciaux.

4.3 – Configuration théorique des défenses

La configuration d'un firewall est l'élément clef de son efficacité. Un firewall mal configuré peut être tout aussi efficace… qu'aucun firewall du tout. C'est la clef de son bon fonctionnement et de son efficacité.
Il existe deux politiques de configurations différentes en ce qui concerne la « base » du firewall :

Tout autoriser sauf ce qui est dangereux : cette méthode est beaucoup trop laxiste. En effet, cela laisse toute latitude à l'imagination des intrus de s'exprimer. Et à moins d'avoir tout prévu de façon exhaustive, on laissera forcément des portes ouvertes, des failles béantes dans notre système. A éviter absolument.

Tout interdire sauf ce dont on a besoin et ce en quoi on a confiance : cette politique est beaucoup plus sécuritaire. En effet, les services sont examinés avant d'être autorisés à passer le firewall, et sont donc tous soumis à un examen plus ou moins approfondi. Ainsi, pas de mauvaise surprise sur un service que l'on pensait ne pas avoir installé, plus d'oubli : tout service autorisé est explicitement déclaré dans le firewall. Cette politique s'accompagne de la création de deux zones : une zone interne et l'extérieur. On peut considérer que tout ce qui est dans notre réseau local est autorisé, sans prendre de trop gros risques : le firewall est là pour nous protéger des attaques extérieures, pas des attaques internes pour lesquelles il ne peut rien. Cette facette peut changer suivant la politique de l'entreprise (interdire l'accès à certains, jeux, etc.). La zone externe est par contre considérée comme « non sûre », et donc toute requête envoyée sur un service non explicitement déclaré comme accessible de l'extérieur sera interceptée et ignorée. La configuration de la DMZ est ici très importante, et sa gestion aussi. Cette politique s'accompagne de plusieurs points à noter :

Plus de services sont ouverts, plus vulnérable est le système. C'est logique, car plus le nombre de logiciels accessibles de l'extérieur est grand, plus le risque qu'un intrus exploite ces dits logiciels pour s'introduire dans le système est important. C'est ainsi que, par exemple, si on utilise un serveur Web qui interface déjà le serveur de base de données, il est inutile d'autoriser le trafic entrant vers le serveur de base de données… vu que le serveur Web joue le rôle d'interface.

Suivant la politique de l'entreprise, l'accès ou non à certains services peut être bloqué dans les deux sens. Cela peut servir, par exemple, à empêcher le jeu en ligne, ou autres activités que l'entreprise ne désire pas voir se dérouler sur ses propres infrastructures.

Certains protocoles sont assez difficiles à autoriser, notamment le Ftp. Le comportement du protocole Ftp est assez atypique et mérite que l'on s'y attarde. Le fonctionnement du Ftp prévoit que ce soit le serveur qui initie la connexion sur le client pour lui transmettre le fichier. Un exemple concret :

- Le client demande le fichier index.txt
- Le serveur envoie un message au client « accepte la connexion sur le port 2563 »
- Le client attend une connexion sur ce port et renvoie un ACK au serveur
- Le serveur initie la connexion et lance le transfert de données.

Ce comportement implique que le serveur, dans la zone « externe », initie une connexion sur un port choisi par lui-même sur le client. Or, nous avons explicitement interdit ce genre de manipulation via notre politique. Il y a donc deux solutions :

- Interdire le FTP.
- Forcer le client à utiliser la commande PASV, qui indique que le serveur doit adopter un comportement passif, et accepter la connexion du client sur un port spécifié par ce dernier. C'est donc le client qui initiera la connexion, et donc, la connexion sera autorisée par le firewall. Avec la commande PASV, l'échange se passe donc ainsi :
    - Le client envoie la commande PASV

- Le serveur répond avec l'adresse et le port auquel le client peut se connecter
- Le client demande le fichier index.txt (RETR index.txt)
- Le serveur envoie un reçu et attend la connexion du client
- Le client se connecte et reçoit le fichier

La configuration efficace d'un firewall n'est pas chose évidente, et implique une grande rigueur, la moindre erreur ouvrant une brèche exploitable par les hackers.

4.4 – Les réactions des firewalls aux attaques classiques

4.4.1 – IP spoofing

L'IP spoofing consiste à modifier les paquets IP afin de faire croire au firewall qu'ils proviennent d'une adresse IP considérée comme « de confiance ». Par exemple, une IP présente dans le réseau local de l'entreprise. Cela laissera donc toute latitude au hacker de passer outre les règles du firewall afin d'envoyer ses propres paquets dans le réseau de l'entreprise. Les derniers firewalls peuvent offrir une protection contre ce type d'attaque, notamment en utilisant un protocole VPN, par exemple IPSec. Cela va chiffrer les entêtes des paquets, et ainsi rendre impossible leur modification par un intrus, et surtout, l'intrus ne pourra générer de paquets comme provenant de ce réseau local, ce dernier n'ayant pas la clé nécessaire au cryptage. Les algorithmes utilisés dans de tels protocoles sont de type RSA.

4.4.2 – DOS et DDOS

Le DOS, ou Denial Of Service attack, consiste à envoyer le plus de paquets possibles vers un serveur, générant beaucoup de trafic inutile, et bloquant ainsi l'accès aux utilisateurs normaux. Le DDOS, pour Distributed DOS, implique de venir de différentes machines simultanées, cette action étant le plus souvent déclenchée par un virus : ce dernier va d'abord infecter nombre de machines, puis à une date donnée, va envoyer depuis chaque ordinateur infecté des paquets inutiles vers une cible donnée. On appelle aussi ce type d'attaque « flood ». Les firewalls ici n'ont que peu d'utilité. En effet, une attaque DOS ou DDOS utilise le plus souvent des adresses sources différentes (le but n'est pas de récupérer une réponse ici) et souvent, impossible de distinguer ces paquets des autres… Certains firewalls offrent une protection basique contre ce genre d'attaque, par exemple en droppant les paquets si une source devient trop gourmande, mais généralement, ces protections sont inutiles. Cette attaque brute reste un des gros problèmes actuels, car elle est très difficilement évitable.

4.4.3 – Port scanning

Ceci constitue en fait une « pré-attaque » (Etape de découverte). Elle consiste à déterminer quels ports sont ouverts afin de déterminer quelles sont les vulnérabilités du système. Le firewall va, dans quasiment tous les cas, pouvoir bloquer ces scans en annoncent le port comme « fermé ». Elles sont aussi aisément détectables car elles proviennent de la même source faisant les requêtes sur tous les ports de la machine. Il suffit donc au firewall de bloquer temporairement cette adresse afin de ne renvoyer aucun résultat au scanner.

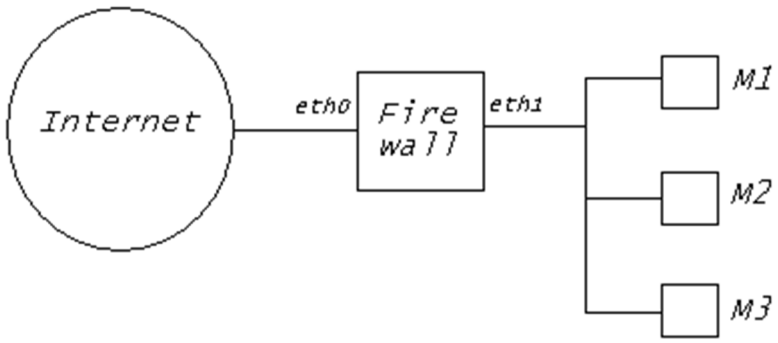
4.4.4 – Exploit

Les exploits se font en exploitant les vulnérabilités des logiciels installés, par exemple un serveur Http, Ftp, etc. Le problème est que ce type d'attaque est très souvent considéré

comme des requêtes tout à fait « valides » et que chaque attaque est différente d'une autre, vu que le bug passe souvent par reproduction de requêtes valides non prévues par le programmeur du logiciel. Autrement dit, il est quasiment impossible au firewall d'intercepter ces attaques, qui sont considérées comme des requêtes normales au système, mais exploitant un bug du serveur le plus souvent. La seule solution est la mise à jour périodique des logiciels utilisés afin de barrer cette voie d'accès au fur et à mesure qu'elles sont découvertes.

4.5 – Un exemple pratique : netfilter

Le réseau est assez basique et est installé comme suit :



Le firewall dispose donc de deux cartes réseau, nommées sous linux eth0 et eth1. Nous allons donc définir deux variables décrivant quelle interface est reliée au réseau local et laquelle est reliée au WAN. Nous préciseront aussi le nom de l'interface de loopback (qui redirige vers le routeur lui même) :

```
IN="eth1"
EXT="eth0"
LO="lo"
```

Viennent ensuite plusieurs variables permettant de décrire dans l'ordre le réseau actuel, les réseaux considérés comme locaux, le chemin vers le binaire d'iptables et une commande permettant de récupérer l'IP associée à l'interface externe, donc notre IP sur le WAN :

```
NETIN="192.168.0.1"
PRIVATE_NETS="10.0.0.0/8 172.16.0.0/12 192.168.0.0/16"
IPTABLES="/sbin/iptables"
EXTIP=`/sbin/ifconfig $EXT | grep inet | cut -d : -f 2 | cut -d \ -f 1`
```

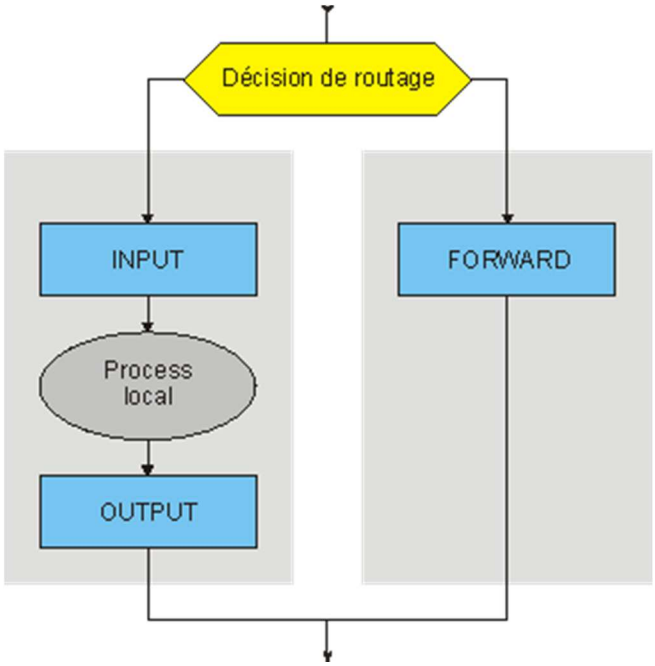
Ensuite, comme nous voulons que nos machines en local puissant accéder au WAN via le Nat, nous allons l'activer :

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

Enfin vient la configuration en elle même des règles du firewall. Iptables repose sur un système de tables qui correspondent aux différents types de trafic. Ici, nous allons utiliser deux tables, filter et nat. La table nat regroupe tout le trafic émis en local, tandis que filter lui va regrouper les trafics entrants non expressément demandés en local. A chaque table est associé différentes chaînes correspondant aux étapes majeures du routage interne effectué pour ce type de trafic :

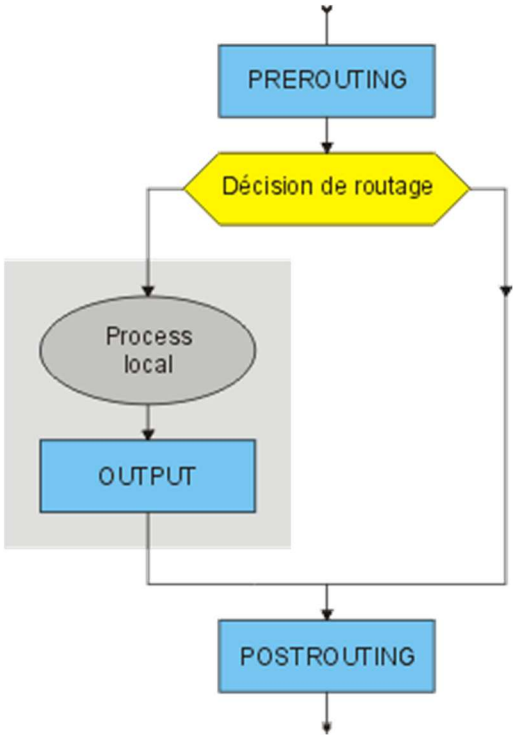
Pour la table filter :

- INPUT et OUTPUT, si les paquets sont destinés au routeur, à un de ses processus, alors il va d'abord passer par la chaîne INPUT, puis être délivré au processus local, qui va émettre ou non une réponse. La réponse, si émise, devra être conforme aux règles de la chaîne OUTPUT.
- FORWARD, si le paquet est destiné à un autre hôte du réseau, alors ce dernier devra passer par cette chaîne.



Pour la table nat :

- PREROUTING, c'est la chaîne qui va être utilisée pour faire du DNAT, ou Destination NAT.
- POSTROUTING, elle se trouve à la sortie du routeur, et servira à faire du SNAT, ou Source NAT, c'est-à-dire du masquage d'adresse source (cas typique : quand un ordinateur local veut sortir sur le WAN, le routeur va remplacer l'IP du paquet émis en local par sa propre IP).
- OUTPUT, cette chaîne va traiter les réponses émises en local si le paquet avait pour destination le routeur, comme dans le cas de la table filter.



Nous allons donc initialiser ces tables, et les vider pour être sur de commencer sur une base saine et vierge :

```
$IPTABLES -t filter -F INPUT
$IPTABLES -t filter -F FORWARD
$IPTABLES -t filter -F OUTPUT

$IPTABLES -t nat -F PREROUTING
$IPTABLES -t nat -F OUTPUT
$IPTABLES -t nat -F POSTROUTING
```

A ce stade, nous avons un système vierge où toutes les chaînes sont vides, c'est à dire ou aucune règle n'est en vigueur. Nous allons donc créer ces règles. Par défaut, comme décrit auparavant, nous allons interdire tout trafic de la table filter qui ne soit pas expressément autorisé. Ces règles vont être les politiques « par défaut », gérant le cas où aucune règle de la chaîne ne s'applique au paquet concerné.

```
$IPTABLES -t filter -P INPUT DROP
$IPTABLES -t filter -P FORWARD DROP
$IPTABLES -t filter -P OUTPUT DROP
```

Par contre, comme nous avons confiance en nos utilisateurs, nous allons autoriser le trafic vers l'extérieur demandé en interne. Il nous suffirait de calquer notre politique sur celle de la table filter pour obtenir le même type de sécurité et fermer des services à nos utilisateurs locaux :

```
$IPTABLES -t nat -P PREROUTING ACCEPT
$IPTABLES -t nat -P OUTPUT ACCEPT
$IPTABLES -t nat -P POSTROUTING ACCEPT

$IPTABLES -t nat -A POSTROUTING -o $EXT -j MASQUERADE
```

Néanmoins, lorsqu'un de nos utilisateurs locaux va demander par exemple une page Http, la réponse va passer par la table filter et va donc être jetée. Nous devons donc autoriser les réponses à revenir à leur destinataire :

```
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Maintenant, nous allons commencer à sélectionner les paquets qui vont être autorisés à rentrer dans notre réseau. Comme nous allons autoriser des services, nous devons d'abord être sûrs que ces derniers sont autorisés à rentrer. C'est le but de cette commande qui va jeter tous les paquets qui nous sont parvenus dont l'adresse de destination n'est pas la notre :

```
if [ -n "$EXTIP" ]; then
$IPTABLES -t nat -A PREROUTING -i $EXT -d ! $EXTIP -j LOG --log-prefix "FW
abnormal prerouting: "
$IPTABLES -t nat -A PREROUTING -i $EXT -d ! $EXTIP -j DROP
fi
```

La loopback est aussi quelque chose que nous pouvons autoriser, il n'y a aucun danger à ce que le routeur émette des paquets pour lui même :

```
$IPTABLES -A INPUT -i $LO -j ACCEPT
$IPTABLES -A OUTPUT -o $LO -j ACCEPT
```

Ensuite, nous allons autoriser tout le trafic expressément émis par le routeur. Il est assez sûr d'autoriser cela, car on imagine que l'administrateur système et réseau, si il fait son propre firewall en iptables, sait ce qu'il installe sur son routeur :

```
$IPTABLES -A OUTPUT -j ACCEPT
$IPTABLES -t nat -A OUTPUT -j ACCEPT
```

Comme nous avons encore confiance en nos utilisateurs, nous allons autoriser tout le trafic qu'ils génèrent vers le routeur :

```
$IPTABLES -A INPUT -m state --state NEW -i $IN -j ACCEPT
$IPTABLES -A OUTPUT -m state --state NEW -o $IN -j ACCEPT
```

Et nous allons accepter toute redirection de trafic de notre réseau local vers l'extérieur.

```
$IPTABLES -A FORWARD -m state --state NEW -i $IN -o $EXT -j ACCEPT
```

A ce stade, tous les paquets entrants passeront par la table filter, et devront donc être droppés. Nous avons coupé tout accès entrant non expressément autorisé. Typiquement, c'est la configuration bastion ou l'entreprise ne propose aucun service Http ou Ftp, mais veut permettre à ses employés de sortir sur le WAN. Nous avons autorisé tout trafic considéré comme sûr, et si aucune règle ne s'applique au paquet, il subira la politique par défaut : si il est interne il est autorisé, si il est externe il est interdit.

Enfin, nous allons autoriser expressément les services que nous ouvrons à l'extérieur, par exemple le trafic Http, qui est de type Tcp sur le port 80 :

```
$IPTABLES -A INPUT -m state --state NEW -i $EXT -p Tcp --dport 80 -j ACCEPT
```

Il nous suffit ainsi de répéter cette commande, en mettant le numéro de port et le type de trafic ad hoc pour ouvrir nos services à l'extérieur.

Ce firewall est très robuste et permet ainsi un niveau de sécurité assez élevé, largement suffisant dans le cas d'un réseau local.

5 – Conclusion

Nous avons vu, dans cette publication, les différents types de firewalls, les différentes attaques et parades. Il ne faut pas perdre de vue qu'aucun firewall n'est infaillible et que tout firewall n'est efficace que si bien configuré. De plus, un firewall n'apporte pas une sécurité maximale et n'est pas une fin en soi. Il n'est qu'un outil pour sécuriser et ne peut en aucun cas être le seul instrument de sécurisation d'un réseau. Un système comportant énormément de failles ne deviendra jamais ultra-sécurisé juste par l'installation d'un firewall.

Toutes ces technologies sont et seront en pleine évolution, car la base même de tout cela est de jouer au chat et à la souris entre les hackers et les programmeurs de firewall ainsi que les administrateurs. Une grande bataille d'imagination qui n'aura certainement jamais de fi