République Algérienne Démocratique et Populaire
Université Mouloud Mammeri de Tizi-Ouzou
Faculté Génie Electrique et Informatique
Département Informatique

Mémoire de Fin d'études

Pour l'obtention de Diplôme Académique en Master II

Spécialité : Réseaux, Mobilités et Systèmes Embarqués

Thème : Réalisation d'une Interface Graphique pour le Pare-feu Netfilter de Linux

Réalisé par :

- DIALLO Sory Binta
- BELKALEM Sarah

Présenter devant le jury composé de :

- M^{me} AOUDJIT Rachida Lagab	Présidente
- M^{me} BELKADI Malika	Promotrice
- M^{me} OUKFIF Karima	Examinatrice
- Mr DAOUI Mehammed	Co-promoteur

Promotion 2018-2019

Table des matières

INTRODUCTION GENERALE 4

Chapitre I : Les risques et menaces dans les réseaux informatiques, et les notions de protocoles réseaux OSI et TCP/IP 5

1. Introduction 5
2. Les différents types d'attaques réseaux 5

2.1. L'attaque directe 5

2.2. L'attaque indirecte par rebond 6

2.3. l'attaque indirecte par réponse 6

3. Quelques attaques courantes 7

3.1. Les attaques virales 7

3.1.1. Les virus 7

3.1.2. Les chevaux de Troie ou trojans 7

3.2. Les attaques d'accès 7

3.2.1. Le trust exploitation (l'exploitation de confiance) 7

3.2.2. L'attaque Man in the middle (l'homme au milieu) 8

3.3. Les attaques de déni de service 8

3.3.1. Le Ping de la mort (Ping of Dead) 8

3.3.2. Le Smurf 8

3.3.3. L'IP Spoofing 8

4. Les notions de protocoles OSI et TCP/IP 9

4.1. Le modèle OSI 9

4.2. Le modèle TCP/IP 11

5. Conclusion 12

Chapitre II : L'état de l'art sur les pare-feux 13

1. Introduction 13
2. Où placer un pare-feu ? 13
3. Fonctionnement général d'un pare-feu 13
4. Politique de sécurité, comment définir ? 14
5. Architecture de pare-feu 15

5.1. L'architecture simple (couche réseau de transport) 16
5.2. L'architecture proxy (couche applicative) 16
5.3. La zone démilitarisée (Demilitarised zone ou DMZ) 17
5.4. L'architecture Nat (Network Address Translation) 18
6. Types ou catégories de pare-feux 18
6.1. Les pare-feu à filtrage de paquet sans état (Stateless firewall) 19
6.1.1. Les limites 19
6.2. Les pare-feux à filtrage de paquets avec état (Statefull firewall) 19
6.2.1. Les limites 19
6.3. Les pare-feux à filtrage applicatif ou pare-feux de type proxy (Proxing applicatif) 19
6.3.1. Les limites 20
7. Les limites d'un système de pare-feu 20
8. Conclusion 20
Chapitre III : Etude du firewall Netfilter/Iptables 21
1. Introduction 21
2. Architecture de Netfilter 21
2.1. Architecture générale de Netfilter 21
2.2. Architecture interne de Netfilter (code noyau) 22
3. Fonctionnalités 24
3.1. Filtrage de paquets 24
3.2. Nat 25
3.2.1. Types de Nat 25
3.3. Modification des paquets 26
3.4. Suivie de connexions 27
3.4.1. Cas d'une connexion TCP/IP 28
3.4.2. Cas du protocole UDP 29
3.5. Autres auditions 29
4. Iptables 29
4.1. Les tables et les chaines 30
4.1.1. Les tables 30
4.1.2. Les chaines 31
4.1.3. Les chaines que contiennent les tables 32
4.2. Les règles Iptables et les tables 33

5. Conclusion 35
Chapitre IV : Réalisation de notre interface graphique pour le pare-feu Netfilter/Iptables 36
1. Les exigences fonctionnelles et la modélisation structurelle du système 36
1.1. Le diagramme de cas d'utilisation 37
1.2. Le diagramme de classe 37
2. La modélisation comportementale 38
2.1. Le diagramme de séquence 38
3. Implémentation 39
3.1. Outils utilisés 39
3.2. Préparation du Script du pare-feu 39
4. Gestion graphique du pare-feu 40
4.1. L'interface d'accueil 40
4.2. L'interface principale 40
CONCLUSION GENERALE 45
Liste des figures 46
Bibliographie 47

INTRODUCTION GENERALE

Depuis l'invention des technologies de l'informatique et de l'information, ces dernières ont connue un développement remarquablement rapide, mais plus encore lors de ces deux dernières décennies. Ce qui a hautement impacté le monde, au point où on ne peut s'en passé de nos jours.

Le réseau est considéré comme l'une des composantes fondamentales du système d'information, de par sa fonctionnalité d'interconnexion et de partage de ressources sur de petites ou très grandes distances et le moyen d'acheminement des informations nécessaires au bon fonctionnement d'une entreprise.

Toutefois, les utilisateurs de ces technologies ne sont pas tous bienveillants et certains s'en servent pour commettre des actes criminels, d'où la nécessité de la mise en place de la sécurité informatique. La mise en place d'une sécurité informatique permet de protéger les systèmes d'information contre des attaques dites cybercriminelles, ces dernières s'étant elles aussi développée relativement à la technologie. De ce fait, comme pour toute composante critique, la mise en place d'une **sécurité réseau** à la hauteur des enjeux spécifique d'une entreprise est vitale pour la protection et une évolution prospère de celle-ci.

Le thème de notre projet est donc porté sur la sécurité réseau. Nous avons comme objectif de réaliser une interface graphique pour faciliter la configuration d'un pare-feu basé sur l'outil Netfilter de Linux. Cette interface permettra à l'ingénier de sécurité d'introduire les règles du pare-feu sans être obligé d'utiliser les commandes Iptables très complexes et sujettes à des erreurs de configuration. Pour cela nous avons structuré notre mémoire de la manière suivante :

Nous parlerons en premier lieu des risques et menaces auxquels peut être confronté un système d'information dans un réseau et des protocoles réseau dans le premier chapitre.

Puis dans le second chapitre nous feront un état de l'art sur les pare-feux, qui est le principal but de notre projet.

Ensuite nous ferons une étude sur le pare-feu Netfilter/Iptables et ses fonctionnalités sous Linux dans le troisième chapitre.

Enfin, dans le quatrième et dernier chapitre, nous présenterons la solution proposée, c'est-à-dire la réalisation de notre application.

Ainsi, nous allons commencer avec le premier chapitre, dans lequel nous parlerons des risques et menaces auxquels un système d'information peut être exposé et des protocoles réseau.

Chapitre I : Les risques et menaces dans un réseau informatique, et les notions de protocoles réseau OSI et TCP/IP

1. Introduction :

La sécurité informatique est de nos jours un problème majeur dans la gestion des échanges de données dans les réseaux. Internet étant un ensemble de réseaux interconnectés, et facilement accessible à tous car c'est un système ouvert, est donc hors contrôle. Les utilisateurs hostiles profitent de cette liberté pour y réaliser des actes de cyber crimes.

L'évolution rapide de la technologie a malheureusement pour conséquence entre autres, la multiplication de risques et de nouvelles attaques chaque jour de plus en plus sophistiquées dans l'utilisation des réseaux. La transmission d'informations sensibles et le désir d'assurer la confidentialité de celles-ci est devenue un point primordial dans la mise en place de réseaux informatiques. Nous citerons dans les lignes qui suivent, les différents types d'attaques réseau.

2. Les différents types d'attaques réseau :

Nous pouvons repartir les attaques réseau en trois types. Elles se diffèrent de par les techniques avec lesquelles on procède pour les réaliser.

2.1. L'attaque directe :

Une attaque directe, comme son nom l'indique, est réalisée par le hacker en attaquant directement le poste ciblé à partir de son ordinateur. Pour ce faire, il utilisera l'adresse IP de la machine cible pour réaliser diverses actions spécifiques. Chaque utilisateur d'internet dispose d'une adresse IP unique, qui est automatiquement fournie par le fournisseur d'accès. Sauf qu'en exécutant certains programmes sur le net, on laisse des traces qui peuvent permettre à certaines personnes de découvrir notre adresse IP et d'accéder à un certain nombre de ports, qui correspondent chacun à une porte d'entrée.

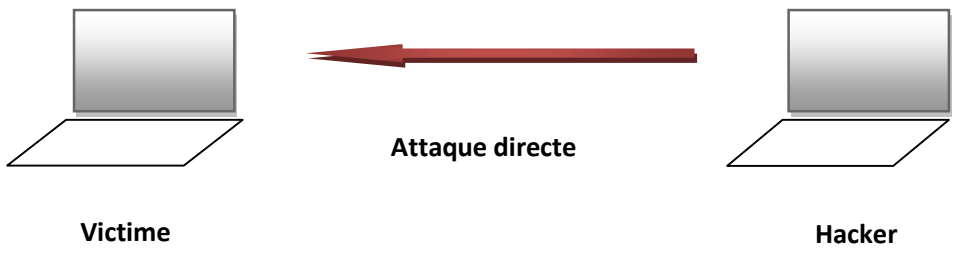


Figure 1: Attaque directe

Les victimes de ce type d'attaques peuvent facilement remonter à la source de l'attaque, connaissant l'adresse IP de celui qui l'a commis.

2.2. l'attaque indirecte par rebond :

Est une attaque qui se réalise en utilisant une machine intermédiaire. Ce qui permettra au hacker de cacher son identité qui correspond à son adresse IP sur le net, et aussi d'utiliser les ressources de la machine intermédiaire pour arriver à ses fins. C'est là que le terme rebond prend son sens.

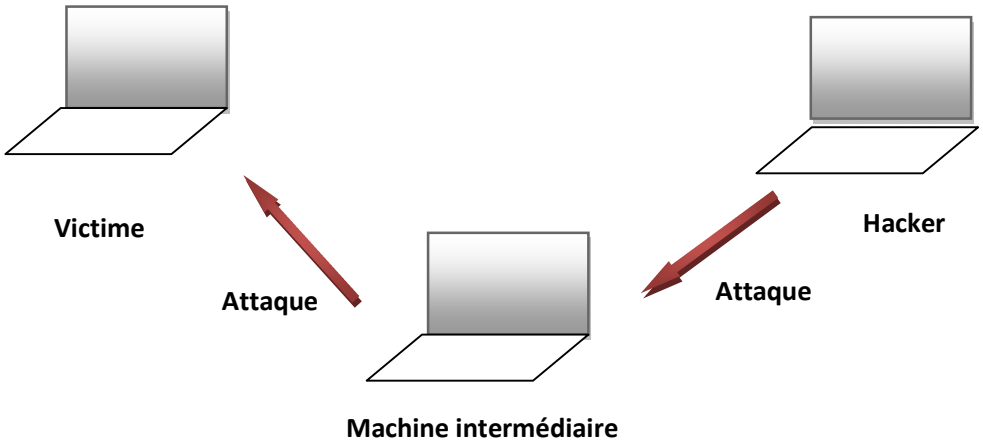


Figure 2: Attaque indirecte sur rebond

Il n'est pas facile d'identifier l'auteur de ce type d'attaques car ne connaissant pas son identité. On pourra par contre remonter jusqu'à la machine intermédiaire.

2.3. l'attaque indirecte par réponse :

Ce dernier type d'attaque est similaire au précédent. Il offre les mêmes avantages du point de vue du hacker. Sauf que au lieu de se servir lui-même de la machine intermédiaire et attaquer la machine cible, il se contente de tromper la machine intermédiaire en se faisant passer pour la machine cible, afin que la réponse de sa requête soit envoyée à la machine cible.

Les hackers utilisent ce type d'attaques pour réaliser des attaques de « **déni de service** » (Denial of service en anglais souvent abrégé Dos) pour surcharger un serveur afin de le rendre inactif de façon temporaire.

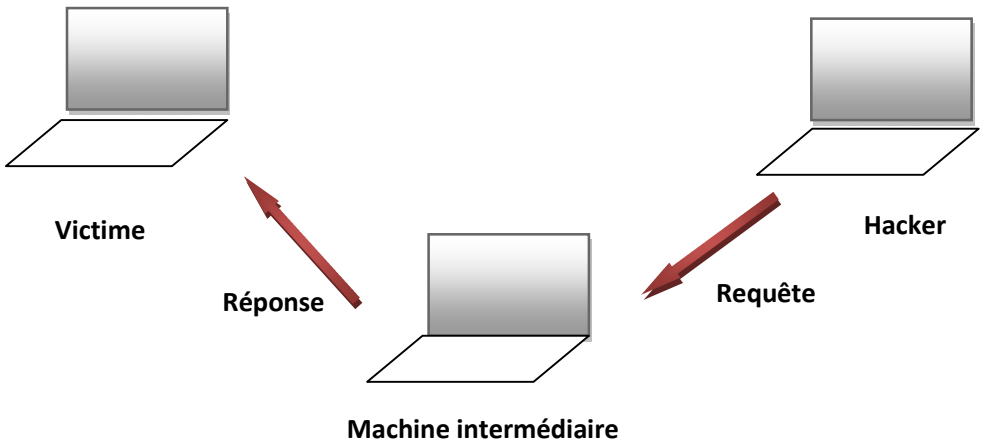


Figure 3: Attaque indirecte par réponse

3. Quelques attaques courantes :

Les attaques réseaux sont très nombreuses, les unes plus dangereuses que d'autres en fonction des actions qu'elles causent.

3.1. les attaques virales :

Sont des attaques qui sont effectuées en partageant des programmes malveillants, souvent camouflés dans des programmes ou application apparemment inoffensifs. Elles existent en grande nombre. Quelques unes sont :

3.1.1. Les virus :

Sont des programmes malveillants, cachés dans des lignes de programmes apparemment fiables pour tromper leurs victimes et pouvoir s'installer sur leurs cibles. Puis une fois activés, ils peuvent alors réaliser leurs objectifs.

3.1.2. Les chevaux de Troie ou trojans :

Comme les virus, sont des programmes cachés dans d'autres qui sont en apparence fiables, mais qui en vrai, exécutent des commandes sournoises en douce. A la différence avec les virus, les trojans sont plus dangereux car ils donnent généralement un accès à distance à la machine victime. Ce qui peut permettre aux pirates de par exemple voler des mots de passe, ou copier des données sensibles ou pire encore , ouvrir volontairement une brèche dans le système de sécurité du réseau en ouvrant des ports de protocoles sensibles de la machine afin de pouvoir s'y inviter ou permettre à d'autres d'entrer quand ils veulent.

3.2. Les attaques d'accès :

3.2.1. Le trust exploitation (exploitation de confiance):

Comme sont nom l'indique, le pirate réalise cette attaque en se servant de la confiance que porte sa machine cible à une autre machine. Voyant qu'il peut facile accéder à cette machine qui lui servira d'intermédiaire, car celle-ci n'est pas bien protéger, il la piratera donc et aura un contrôle à distance sur elle. Ce après quoi il pourra facilement entrer en contact avec sa cible en se faisant passer pour la machine intermédiaire.

Cette attaque est un exemple du type d'attaques sur rebond.

3.2.2. l'attaque Man in the middle (l'homme au milieu) :

Cette attaque est réalisée en surveillant le réseau, afin de pouvoir intercepter des paquets qui y circulent pour les modifier ou juste espionner. Généralement l'intrus crée un point d'accès wifi ouvert, pour faciliter l'accès au réseau à des utilisateurs. Ces derniers ne voyant pas le piège se connectent donc au réseau sans se douter qu'il est surveillé.

3.3. Les attaques de déni de services :

Ces attaques sont réalisées pour paralyser les serveurs d'un réseau de façon temporaire, ou de ralentir les échanges sur ce réseau.

3.3.1. le ping de la mort (ping of dead) :

Cette attaque exploite une faille d'anciennes implémentations de TCP/IP de certains systèmes d'exploitation. Le pirate envoie un paquet d'une taille énorme, supérieur à 65535 octets. La pile TCP/IP de n'étant pas prédisposée à recevoir de paquets de cette taille ne donc pas que faire face à cela et se met à chercher que faire de façon indéfinie. Ce crée le disfonctionnement du réseau.

3.3.2. le Smurf :

A une technique qui est basée sur l'utilisation de serveurs broadcast. Ces derniers servent à dupliquer et diffuser des messages à toutes les machines d'in réseau. La technique est la suivante :

Le hacker va envoyer des Ping à un ou plusieurs serveurs broadcast, mais en changeant son adresse IP et la remplaçant par celle de la machine cible. Ce qui fait que quand le serveur broadcast recevra le ping, il le diffusera à son tour à l'ensemble des machines du réseau, que toutes répondront par le pong à l'adresse de la machine cible pour laquelle le hacker s'était fait passer dans le ping.

Se retrouvant avec autant de messages à traiter en même temps, la machine cible devient lente jusqu'à devenir inactive, étant dépassée par la quantité de tâches à remplir. D'où le déni de services.

3.3.3. L'IP Spoofing :

Cette attaque utilise l'usurpation d'identité d'une machine de confiance pour atteindre la machine cible. Le but est d'envoyer des paquets à la machine cible en lui faisant croire qu'ils viennent de la machine de confiance. Il fait cela en remplaçant son adresse IP par celle de la machine de confiance dans les paquets émis à la machine cible, pour que cette dernière les accepte.

Ici le hacker ne prend pas contrôle du la machine de confiance comme dans l'attaque du trust exploitation, mais cherche juste a avoir son adresse IP afin de se faire passer pour elle au près de la machine cible.

4. Les notions de protocoles OSI et TCP/IP

4.1. Le modèle OSI :

OSI, Open Systems Interconnexion, est un modèle de référence conceptuel qui décrit les fonctions du système de réseau ou de télécommunication indépendamment de l'infrastructure

technologique sous-jacente. Il divise le processus de communication de données en sept couches d'abstraction et normalise les protocoles en groupes appropriés de fonctionnalités réseau. Le modèle OSI a été développé à l'origine pour faciliter l'interopérabilité entre les fournisseurs et définir des normes claires pour la communication réseau. Cependant, l'ancien modèle TCP / IP reste le cadre de référence omniprésent pour les communications Internet.

Le modèle est crée dans le but de donner une description visuelle de ce qui se passe avec un système de réseau particulier. Cela peut aider les gestionnaires de réseau à réduire les problèmes : s'agit-il d'un problème physique ou d'un problème lié à l'application? , Ainsi que les programmeurs informatiques lors du développement d'une application, avec quelles autres couches doit-elle travailler ?

Il s'agit d'une architecture à 7 couches, chaque couche ayant une fonctionnalité spécifique à exécuter. Toutes ces couches travaillent en collaboration pour transmettre les données d'un équipement à un autre comme l'illustre le schéma suivant :

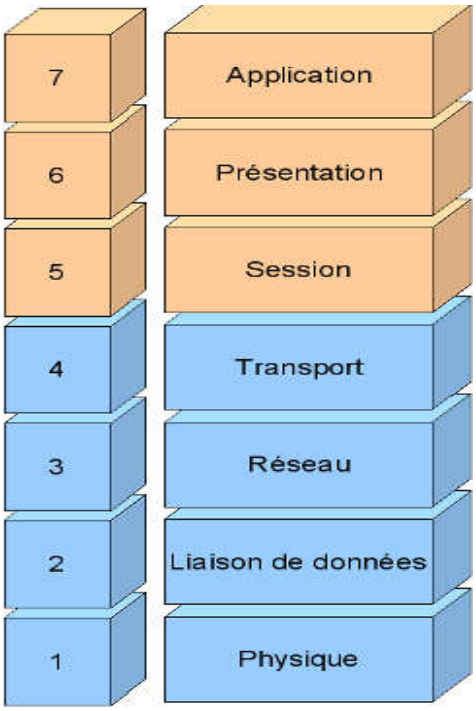


Figure 4 Les couches du modèle OSI

7. Couche application

C'est la couche en haut niveau du modèle OSI. Il s'agit de la couche avec laquelle l'utilisateur final est en train d'interagir. Cette couche permet d'accéder aux ressources du réseau.

6. Couche présentation

C'est la couche dans laquelle le système d'exploitation utilise les données. Les fonctions principales de cette couche incluent la translation, le cryptage et la compression des données.

Fondamentalement, l'utilisateur interagit avec la couche application, qui envoie les données à la couche présentation.

5. Couche session

Cette couche a pour tâche de maintenir une communication correcte en établissant, en gérant et en mettant fin aux sessions entre deux ordinateurs. Par exemple, chaque fois que nous visitons un site Web, notre ordinateur doit créer une session avec le serveur Web de ce site.

4. Couche transport

Cette couche a un travail très important. Elle décide combien d'informations doivent être envoyées à la fois. Ainsi, lors d'une connexion avec un site Web, cette couche décide du volume de données à transférer et à recevoir à un moment donné. En outre, cette couche fournit un processus fiable pour traiter la remise des messages et la reprise sur incident.

3. Couche réseau

Le travail principal de cette couche consiste à acheminer les paquets de la source à la destination et à fournir un inter-réseau.

2. Couche de liaison de données

Cette couche est chargée d'organiser les bits dans les trames et d'assurer la livraison saut par saut. C'est la couche sur laquelle les commutateurs fonctionnent. Comme les routeurs fonctionnent au niveau du réseau, nous pouvons donc dire que l'adresse MAC réside au niveau de la couche liaison de données.

1. couche physique

C'est la couche sur laquelle la transmission réelle des bits de données a lieu à travers un support. Comme son nom l'indique, cette couche regroupe tout le matériel physique qui relie les ordinateurs entre eux.

4.2. Le Modèle TCP/IP :

TCP / IP spécifie la manière dont les données sont échangées sur Internet en fournissant des communications de bout en bout qui identifient la manière dont elles doivent être divisées en paquets, adressées, transmises, acheminées et reçues au niveau de la destination. TCP / IP nécessite peu de gestion centralisée et est conçu pour rendre les réseaux fiables, avec la possibilité de récupérer automatiquement à partir de la défaillance de tout périphérique du réseau.

Le modèle TCP/IP est divisée en quatre couches, chacune comprenant des protocoles spécifiques :

1. Couche accès réseau

Cette couche correspond à la combinaison de la couche liaison de données et de la couche physique du modèle OSI. Il s'occupe de l'adressage physique et les protocoles présents dans cette couche permettent la transmission physique des données.

2. Couche réseau

Cette couche est parallèle aux fonctions de la couche réseau d'OSI. Elle définit les protocoles responsables de la transmission cohérente des données sur l'ensemble du réseau. Les principaux protocoles résidant sur cette couche sont:

IP - Internet Protocol (Internet Protocol) signifie qu'il est responsable du routage et de la livraison des paquets de l'hôte source à l'hôte de destination en examinant les adresses IP dans les en-têtes de paquet. Le protocole IP a 2 versions: IPv4 et IPv6.

ICMP - signifie Internet Control Message Protocol. Il est encapsulé dans des datagrammes IP et est chargé de fournir aux hôtes des informations sur les problèmes de réseau.

ARP - représente le protocole de résolution d'adresse. Son travail consiste à rechercher l'adresse physique d'une carte réseau d'un hôte à partir d'une adresse IP connue.

3. Couche transport

Cette couche est analogue à la couche de transport du modèle OSI. Il est responsable de la communication de bout en bout et de la livraison des données sans erreur. Il protège les applications de la couche supérieure de la complexité des données. Les deux principaux protocoles présents dans cette couche sont:

TCP – Il est connu pour fournir une communication fiable et sans erreur entre les systèmes en extrémité. Il effectue le séquençage et la segmentation des données. Il possède également une fonction d'accusé de réception et contrôle le flux des données via un mécanisme de contrôle de flux.

UDP - C'est le protocole idéal si une application ne nécessite pas de transport fiable. Contrairement à TCP, qui est un protocole orienté connexion, UDP est sans connexion.

4. couche application

Cette couche remplit les fonctions des trois couches supérieures du modèle OSI: couche application, présentation et session. Il est responsable de la communication nœud à nœud et contrôle les spécifications de l'interface utilisateur. Certains des protocoles présents dans cette couche sont les suivants: HTTP, FTP, Telnet, SSH, SMTP, SNMP, NTP, DNS, DHCP, etc.

5. Conclusion :

Nous avons vu lors de ce chapitre, les risques et menaces qu'encourent un réseau informatique, et les modèles OSI et TCP/IP.

La connaissance des ces modèles est primordiale pour comprendre le fonctionnement de réseaux et les conditions d'acheminement des flux de données traversant les firewalls.

La sécurité réseau est de nos jours un véritable challenge, car la technologie réseau se développe et les risques et menaces avec. Il faut donc penser à prendre des mesures nécessaires de protection à la hauteur de la sensibilité des données en jeu, pour tout système informatique offrant des services de connectivités et d'échanges avec l'extérieur.

Nous verrons dans le chapitre suivant l'état de l'art sur les pare-feux, qui sont le principal objectif du thème de notre projet.

Chapitre II : Etat de l'art sur les pare-feux

1. introduction :

Il existe plusieurs solutions de sécurité des systèmes informatiques qui ont été mises en place et les pare-feux sont l'une d'elles. Aussi appelé un coupe-feu, un garde-barrière ou un firewall en anglais, un pare-feu est un système matériel ou logiciel servant d'interface entre un ou plusieurs réseaux, afin de protéger un ordinateur ou un réseau interne des intrusions provenant de réseaux externes. Il comporte donc au minimum deux interfaces réseau :

- Une interface pour le réseau interne
- Une autre pour le réseau externe

Il permet aussi de contrôler les sorties à travers le réseau à protéger.

Le pare-feu a pour rôle de faire respecter la politique de sécurité du réseau préalablement définie, celle-ci énumérant quels sont les types de paquets pouvant circuler dans et à travers le réseau à protéger ; ce en surveillant et contrôlant les applications et les flux de données (paquets).

Le but est de fournir une connectivité contrôlée et maitrisée entre les zones de différents niveaux de confiance.

2. Où placer un pare-feu ?

A l'image d'une porte servant à sécuriser une maison des voleurs, un pare-feu doit être :

- Placer entre deux réseaux (donc à coté d'un routeur) ;
- Un passage obligatoire pour accéder ou sortir du réseau interne ;
- Pour un pare-feu logiciel, installé sur un système d'exploitation. Ici il est aussi soumis aux failles de sécurité de l'os. Mais n'étant qu'un logiciel comme tout autre dans ce cas, une faille dans l'os pourrait par exemple désactiver le pare-feu.

3. Fonctionnement général d'un pare-feu :

Pour protéger un réseau, un système de pare-feu contient un ensemble de règles prédéfinies, permettant :

- De filtrer et n'autoriser la connexion que sous certaines conditions (adresse IP et port), avec la commande ***Allow***,
- De bloquer la connexion et empêcher toute intrusion dans le réseau à l'aide de ***Deny***,
- Rejeter la demande de connexion sans avertir l'émetteur à laide de ***Drop***,
- Aussi de contrôler les sorties des utilisateurs internes.

Pour réaliser cela, un firewall se base sur les couches 3 (IP) et 4 (TCP/UDP donc les ports) du modèle OSI mais aussi sur la couche 7, la couche applicative, donc sur des protocoles comme HTTP.

Un firewall qui se base sur les couches 3 et 4 permet de contrôler les adresses IP et les ports, pour par exemple bloquer ssh ou Telnet, alors qu'un firewall qui se base sur la couche 7 permet lui de bloquer l'utilisation d'un logiciel Peer-to-Peer par exemple.

L'ensemble de ces règles permet de faire appliquer la méthode de filtrage agréée par la politique de sécurité qu'adopte le réseau. Chaque paquet traversant le pare-feu est soumis à ces règles. On distingue habituellement deux types de politique de réseau par défaut :

1- Autoriser uniquement les actions ayant été explicitement autorisées : << Tout ce qui n'est pas explicitement permis est interdit >> ;
2- Bloquer toute action ayant été explicitement interdite.

La première stratégie est la plus prudente, mais impose toute fois une définition précise de communication. La seconde stratégie est la plus radicale, il faudra chaque fois ajouter des règles en fonctions des nouveaux besoins.

Pour un paquet correspondant à une règle, la décision associée est prise. Il est soit autorisé, soit rejeté.

Il est à préciser qu'un firewall ne permet pas de bloquer les virus. Pour cela, il faut un antivirus. En général, on couple les deux dans une société. Et le firewall ne peut protéger un réseau d'une personne se trouvant à l'intérieur du réseau.

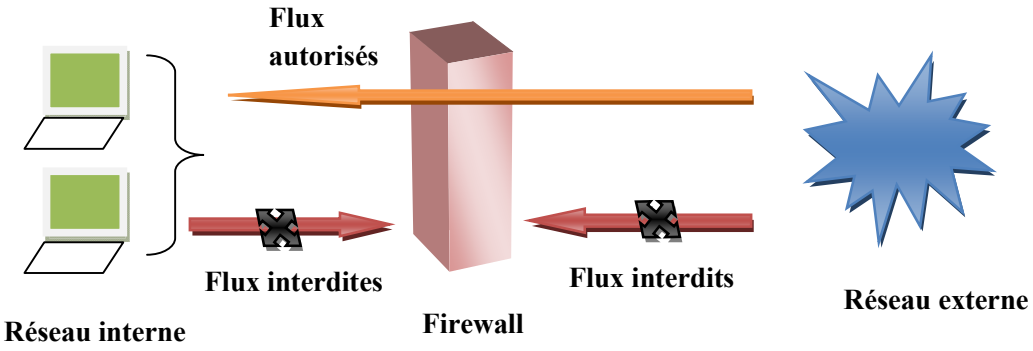


Figure 5: Le fonctionnement d'un pare-feu

4. Politique de sécurité, comment définir ?

Une politique de sécurité c'est le fait d'exposer les besoins de l'entreprise (son utilisation d'Internet) et d'en déduire les règles qui lui permettent son bon fonctionnement, tout en sécurisant au maximum l'entreprise.

C'est le premier pas à faire avant de créer les règles firewall. Elle doit être construite en fonction des besoins de l'entreprise puis être applique ensuite sur le firewall et non l'inverse.

Une entreprise qui vend des vêtements sur Internet par exemple, son besoin est d'avoir un serveur accessible depuis Internet. Les règles firewall, ne doivent donc pas empêcher cette action.

Pour que ces suites de règles puissent correctement répondre aux attentes de l'entreprise, la politique de sécurité doit être créée en concertation avec d'autres membres de l'entreprise afin de pouvoir collecter toutes informations nécessaires au bon fonctionnement du firewall. Ce n'est donc pas une tache que l'administrateur système doit accomplir tout seul si l'on veut comme résultat un firewall qui respecte l'esprit du fonctionnement de l'entreprise.

Parmi les deux types de politique de sécurité cités plus haut, le plus recommandé est le premier, qui consiste à interdire tout ce qui n'est pas explicitement permis. On commence donc par tout interdire au début, puis au fur et à mesure on autorise ce qui a été retenu dans des réunions avec les différents services de l'entreprise. Par exemple, si une réunion a conclu que:

- Les utilisateurs ont besoin d'accéder au Web = ouverture port http ;
- Les utilisateurs ont besoin d'accéder à leurs mails = ouverture port POP ou IMAP ;
- Le serveur doit être accessible depuis Internet = ouverture port HTTP.

La politique de sécurité consistera donc à interdire tout sauf les protocoles :

- HTTP
- POP
- IMAP

Il est évidemment possible d'affiner cela, en définissant sur quels sites les utilisateurs ont le droit d'aller, ou plutôt leur interdire certains sites. Définir des utilisateurs avec des droits différents, si par exemple l'équipe A et B a besoin de télécharger certains fichiers, on les regroupe et leurs autorise le téléchargement FTP, mais il restera bloqué pour les autres utilisateurs.

De plus, si le réseau interne est composé de plusieurs sous-réseaux (un réseau composé des serveurs et un pour les utilisateurs par exemple), il faudra établir quels utilisateurs ont le droit d'accéder au réseau des serveurs et si les serveurs ont le droit d'accéder aux autres réseaux.

Il n'y a pas de règles prédéfinies, chaque entreprise a un besoin particulier qu'on doit comprendre et analyser.

Cette politique de sécurité est la base du travail. Si elle est mal conçue, le reste de du travail, même s'il est bien fait, reposera sur de mauvaises bases et risque de ne pas fonctionner ou de ne pas être suffisamment protecteur.

5. Architectures de pare-feux :

Tout comme pour les politiques de sécurité, il n'existe pas parmi les différentes architectures, une qui peut parfaitement satisfaire les besoins d'une entreprise donnée. Là aussi il faudra adapter une architecture se rapprochant le plus de ce qu'on veut, aux besoins de l'entreprise. Il s'agit en fait de la façon dont sera positionné le firewall.

Pour se faire, on devra prendre en compte :

- Les matériaux à protéger : Serveurs,…
- Les besoins : Les matériaux seront-ils accessibles depuis Internet ?
- Le degré de confidentialité des données : Il y a-t-il des données personnelles à protéger ? Ces données peuvent appartenir aux clients et (ou) au personnel de l'entreprise.

- Les risques : Est-ce une entreprise à risque (qu'un pirate voudrait plus particulièrement attaquer qu'un autre) ?
- Le besoin de contrôle : Quel contrôle le chef de l'entreprise veut-il avoir sur ce qui se passe sur son réseau ?

Ce après quoi on choisira une architecture qui pourra répondre à tous ces besoins.

Nous allons donc citer quelques architectures, leurs modes de fonctionnement et dans quels cas les utiliser.

5.1. Architecture simple (couches réseau et transport) :

C'est l'architecture la plus connue et la plus utilisée. Ici le firewall est positionné de façon classique entre le réseau interne et le réseau externe comme représenté sur la figure 5. L'administrateur Cloud et infrastructure doit avoir une bonne connaissance des règles à appliquer. Il fonctionne de la manière suivante :

- Le filtrage est fait au niveau des adresses IP (couche 3 réseau) et des ports TCP/UDP (couche 4 transport) ;
- On autorise les règles définies dans la politique de sécurité ;
- Le filtrage sur les services tels que HTTP ou FTP n'est pas possible dans cette architecture.

Cette solution est peu coûteuse, un hardware peu puissant combiné à un firewall open source est suffisant (un routeur filtrant par exemple).

Il est conseillé d'utiliser cette architecture lorsque le client ne possède pas de serveur interne ouvert sur l'extérieur.

5.2. L'architecture proxy (couche applicative) :

Un proxy est un matériel ou un logiciel servant d'intermédiaire entre deux réseaux. Une des utilisations les plus courantes de proxy concerne l'utilisation d'Internet (HTTP). Pour qu'un utilisateur du réseau interne puisse accéder à internet il devra d'abord passer par le proxy, qui à son tour enverra la requête HTTP vers internet.

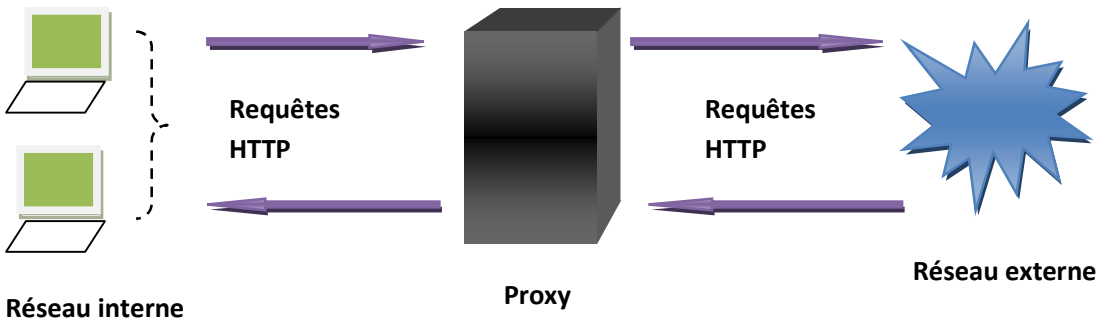


Figure 6 Représentation de l'architecture Proxy

Il s'agit en fait d'une amélioration de la précédente architecture. Elle fonctionne de la manière suivante :

- Le filtrage se fait au niveau de la couche applicative. Ceci permet donc d'aussi filtrer des protocoles tels que HTTP et FTP (il devient possible d'empêcher du Peer-to-Peer) ;
- Permet de faire circuler le trafic HTTP et FTP via le proxy. En cas d'attaque, le proxy servira donc de bouclier au poste utilisateur et subira l'attaque à sa place. Ce qui permet donc de voir les attaques potentielles destinées au réseau interne.
- Contrôle le réseau par utilisateur et permet de garder un historique et d'avoir un contrôle sur l'utilisation complète du réseau (application, bande passante) par utilisateur.

Il est à noter qu'il est tout à fait possible de combiner le filtrage de niveau réseau et transport et filtrage de niveau applicatif.

Cette solution est très coûteuse. Plus il y a d'utilisateurs, plus la connexion internet est puissante et plus le firewall doit être puissant. Il faudra dans ce cas un investissement beaucoup plus conséquent que pour un firewall de couches 3 et 4.

L'utilisation cette architecture est conseillée quand on a besoin de contrôler l'utilisation du réseau, comme une école par exemple ou un réseau wifi libre-service. Car c'est dans ces cas-là que l'on trouve le plus souvent des utilisateurs dangereux (pirates), imprudents (téléchargements) ou dans l'illégalité (Peer-to-Peer).

Un proxy est souvent derrière un firewall sur un hôte dit **bastion**, celui-ci étant un élément du réseau informatique qui fournit un point d'entrée et / ou de sortie unique vers un réseau externe. Pour un échange entre le réseau interne et externe, ces deux doivent obligatoirement passer par le bastion. Il fournit donc tous les services du proxy et peut être implémenté sur un hôte à double interfaces (par une combinaison routeur filtrant-hôte par exemple).

5.3. La zone démilitarisée (Demilitarised zone ou DMZ) :

La zone démilitarisée, comme sont nom l'indique, consiste à créer une zone frontalière où le niveau de sécurité est moyenne, entre deux réseaux dont un a un niveau de sécurité élevé (réseau de confiance) et l'autre un niveau de sécurité très faible (internet) ; et qui est accessible aux deux réseaux. Ceci se fait en ajoutant un deuxième firewall entre le réseau interne de confiance ou LAN et les serveurs de celui-ci placés dans la DMZ. Et les serveurs seront protégés du réseau externe par le premier firewall dit firewall externe ou WAN.

- Le firewall externe est configuré de telle sorte qu'il offre une sécurité suffisante à la DMZ lors de ses échanges avec le WAN ;
- Une règle permet au WAN d'accéder à la DMZ par le firewall externe, selon le protocole voulu (http par exemple). Tout échange par les autres protocoles sera interdite, et la connexion ssh impossible ;
- Une autre règle permet au réseau interne d'accéder à la DMZ (par ssh par exemple), tout en empêchant l'inverse, pour protéger le LAN au cas où la sécurité du DMZ serait compromise. Cela par le billet du firewall interne.
- Cette architecture est utilisée lorsqu'on dispose des serveurs dont on veut rendre accessible depuis le WAN, tout en sécurisant le LAN.
- Il est à noter possible d'ajouter les concepts des deux premières architectures à celle-ci.

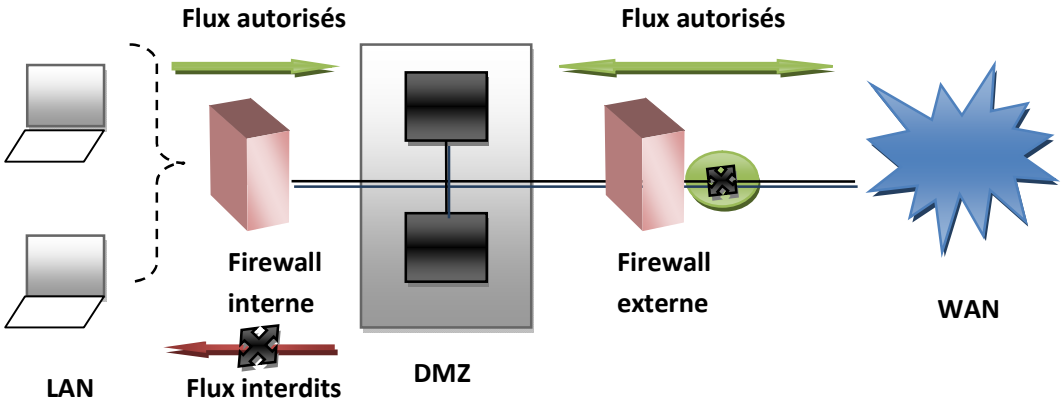


Figure 7 Représentation de l'architecture DMZ

5.4. L'architecture NAT (Network Address Translation):

Le NAT, Traduction d'Adresse Réseau en français, permet de protéger le LAN des attaques directes et contre l'écoute du réseau, en masquant les adresses IP du LAN. Nous expliqueront de façon détaillée le fonctionnement du NAT dans le chapitre suivant.

6. Types ou catégories de pare-feu :

Les firewalls font parties des moyens de sécurit les plus vieux dans la sécurité informatique et ont subit au fur du temps, de nombreuses évolutions. Suivant ces évolutions et le rôle des pare-feux, on peut les classer par catégories.

6.1. Les pare-feux à filtrage de paquets sans état (Stateless firewall) :

C'est le type de firewall le plus vieux et le plus basique, utilisé dans le filtrage réseau. Tout firewall doit au moins fournir cette fonctionnalité. Il regarde chaque paquet indépendamment des autres, les filtre en fonction des règles préconfigurées et conformes à la politique de sécurité adoptée, en se basant sur les informations contenues dans la couche réseau (IP) et transport (TCP ou UDP) du modèle OSI. Ce type de pare-feu a pour avantages la simplicité dans la définition des règles et sa rapidité de traitement.

6.1.1. Les limites :

- Le contrôle effectué sur les paquets est sommaire :

- Les ports supérieurs à 1023 sont fermés en entrée et ouverts en sortie.
- Les ports inférieurs à 1023 sont fermés en entrée, sauf pour les serveurs disponibles.

- Ne protège pas contre les attaques utilisant les paquets incohérents ;
- Les fichiers logs générés ne contiennent pad d'informations très pertinentes (seulement les adresses IP et ports).

6.2. Les pare-feux à filtrage de paquets avec état (Statefull firewall) :

Plus amélioré, ce type de firewall vérifie que les paquets appartiennent à une session régulière, en se référant à sa table d'états où est stocké un suivi de chaque connexion établie, ce qui permet au firewall de prendre des décisions en fonction des états de connexion et de réagir dans le cas de situations protocolaires anormales.

Pour tout paquet appartenant à une nouvelle connexion, il applique les règles de filtrage et une fois le paquet admis, il rajoute alors le paquet à la table de connexion.

6.2.1. Les limites :

- Une fois l'accès à un service est autorisé, il n'y a aucun contrôle effectué sur les requêtes et réponses des clients et serveurs ;
- Les protocoles personnalisés utilisant plusieurs flux de données ne passeront pas, car le système de filtrage dynamique n'aura pas connaissance du protocole.

6.3. Les pare-feu à filtrage applicatif ou pare-feux de type proxy (proxying applicatif) :

Comme son nom l'indique, le filtrage applicatif est réalisé au niveau de la couche application, en plus des couches réseaux et transport. En générale, les informations au niveau 7 sont complexes et nécessitent donc une application pour chacun des protocoles (http, ftp,…). Les requêtes sont traitées par des processus dédiés, par exemple une requête de type http sera filtrée par un processus proxy http, le proxy étant le module qui gère les informations de la couche application.

Le firewall s'intercale dans la session et analyse les informations afin de vérifier les échanges protocolaires. Celles-ci sont autorisées si elles sont conformes aux normes, ou soit rejetés soit classées à des fins d'analyses. Il peut aussi soumettre l'information pour un scan contre les virus, spams, etc… en plus des règles ***Allow***, ***Deny*** et ***Drop***, la règle ***Filter*** est ajoutée, permettant de soumettre les paquets au serveur proxy pour des prises de décisions.

6.3.1. Les limites :

Pour que le proxy puisse bien faire son travail, il faut qu'il connaisse les règles protocolaires de tous les protocoles de niveau 7. Nuance vu le nombre de protocoles de la couche application.

7. Les limites d'un système de pare-feu :

Un système pare-feu n'offre bien évidemment pas une sécurité absolue, il n'offre une protection que dans la mesure où l'ensemble des communications vers l'extérieur passe systématiquement par leur intermédiaire et qu'ils sont correctement configurés. Un firewall ne peut protéger l'environnement à sécuriser contre les attaques ou les accès illicites qui ne passent pas par lui et est inefficace en cas d'attaques venant de l'intérieur. N'étant pas un antivirus, un firewall ne peut protéger des virus ; il faudra lui-même le protéger de manière complémentaire contre les infections virales. Dans l'absolu, tout système offrant des services de connectivité doit être protégé d'un antivirus, surtout quand il contient des données sensibles.

Par ailleurs, un firewall logique est lui-même soumis à la vulnérabilité du la machine qui l'héberge. Certains virus ou d'autres programmes hostiles peuvent ainsi désactiver ou contourner un firewall personnel. Un minimum de vigilance reste donc nécessaire, et il est notamment indispensable de ne pas exécuter sans réfléchir les fichiers joints aux courriers électroniques reçus. Un antivirus sera le complément naturel et indispensable d'un firewall personnel, tout en gardant à l'esprit que lui aussi ne protège essentiellement que contre les virus et chevaux de Troie connus.

8. Conclusion:

La mise en place d'un ou de plusieurs systèmes de pare-feu est un moyen de sécurité nécessaire pour un réseau local. Ceci permet de cloisonner le réseau à protéger du reste d'internet de façon physique ou logique et d'avoir un certains contrôle sur les échanges du réseau l'extérieur.

Pour une bonne efficacité du firewall, on doit d'abord énumérer tous les besoin de l'entité à protéger (les échanges et accès qu'elle voudrait permettre sur son réseau) pour définir une politique de sécurité conforme à tous ces besoins, ce en fonction de quoi on choisira une architecture réseau qui jouera le rôle de police afin de faire appliquer les règle de l'entité. Mais le système de firewall à lui tout seul ne peut offrir une sécurité suffisante au réseau et les systèmes d'une organisation. Il faudra l'accompagner d'autres équipements, de mesures et de procédures, qui ensemble répondront au maximum aux objectifs de sécurité préalablement définis.

Nous verrons dans le chapitre suivant une étude détaillée du firewall Netfilter/Iptables de Linux très performant, qui utilise bien dans sa configuration les entêtes de plusieurs protocoles réseau et qui nous servira de base dans la réalisation de notre pare-feu.

Chapitre III : Etude du firewall Netfilter/Iptables

1. Introduction :

Un des apports majeurs du noyau Linux est sans conteste sa couche de firewalling. La fonctionnalité de pare-feu est réalisée directement dans le noyau Linux lui-même par un Framework de filtrage de paquets appelé Netfilter.

Netfilter est un framework de manipulation de paquets intégré aux noyaux Linux 2.4 et 2.6. Il s'agit d'une fonctionnalité optionnelle au moment de la compilation, mais elle est généralement présente dans toutes les distributions Linux normales en tant que partie du noyau fourni par les responsables de la distribution.

Netfilter n'est que le nom générique du paquetage. Celui-ci contient une commande d'administration appelé **Iptables.** Lorsque l'on parle de filtrage sous Linux, ont emploi généralement Netfilter et Iptables indifféremment. Hors que Netfilter est le code noyau intégré au Kernel et Iptables est un système de sélection de paquets dans l'espace utilisateur. Iptables est superposé au Framework Netfilter. Ensemble, ils produisent un système modulaire qui permet au noyau Linux de fonctionner en tant que pare-feu.

Ce chapitre a pour but de présenter succinctement tout les aspects de Netfilter. On parlera de l'architecture et la conception de Netfilter, les principales fonctionnalités de Netfilter, ainsi que l'outil de configuration Iptables.

2. Architecture de Netfilter :

Lors de son développement, les développeurs de Netfilter se sont grandement inspirés de la syntaxe de ses prédécesseurs (ipfwadm avec le noyau 2.0 et ipchains avec le noyau 2.2) et les concepts sont à bien des égards les mêmes. Ces solutions restent tout de même très différentes conceptuellement.

Alors que pour les versions du noyau antérieures, le code source correspondant au filtrage des paquets étaient disséminés un peu partout à travers l'implémentation de la pile TCP/IP, avec l'arrivée du noyau 2.4, les développeurs ont introduit la notion de « **hooks** » qui ont placé à des endroits spécifiques du code. Cela permettait d'avoir un code à la fois plus clair et plus souple.

2.1 Architecture générale de Netfilter :

De façon générale, Netfilter consiste de :

- Une partie noyau établissant la structure ''statique'' de l'algorithme, qui sont les points d'entrées au niveau de la traversée de paquets dans la pile de protocole IP. On les appelle : **les hooks;**

- Des modules noyau chargeables dynamiquement, établissant le lien avec les outils disponibles en espace utilisateur ;

- La bibliothèque partagée (en mode user, donc) permettant de configurer les règles de filtrage;

- Iptables, la commande utilisateur, bien que ne pouvant être utilisée que par l'administrateur, est l'outil générique de gestion des règles de filtrage et de la translation d'adresses (NAT) ;

- Les extensions chargeables dynamiquement, spécifiques à chaque protocole. Les extensions livrées en standard avec le paquetage Netfilter concernent les protocoles UDP, TCP, et ICMP, mais aussi bien d'autres aspects du processus de filtrage. Ces extensions sont composées d'une partie module noyau, et d'une bibliothèque chargeable dynamiquement en espace utilisateur.

2.2 Architecture interne de Netfilter : (code noyau)

La structure de Netfilter s'appuie sur des points bien précis appelés points d'entrée, appelés aussi « hooks », crochets ou bien « points d'accrochage ». Ils sont insérés dans le code noyau de Netfilter, en des points stratégiques de la pile de protocole, pour le moment, IPv4, IPv6, DECnet, permettant d'effectuer autant d'actions particulières que nécessaire aux endroits clés de la traversée des paquets.

Au fur et à mesure que les paquets progressent dans la pile, ils déclencheront les modules du noyau qui sont implémentés dans ces points d'accrochages. Le Hook qu'un paquet va déclencher dépend du sens de circulation du paquet, soit entrant ou sortant, de la destination du paquet et aussi de d'autres paramètres qu'on verra par la suite.

Un schéma conventionnel utilisé pour résumer illustrer ceci est le suivant :

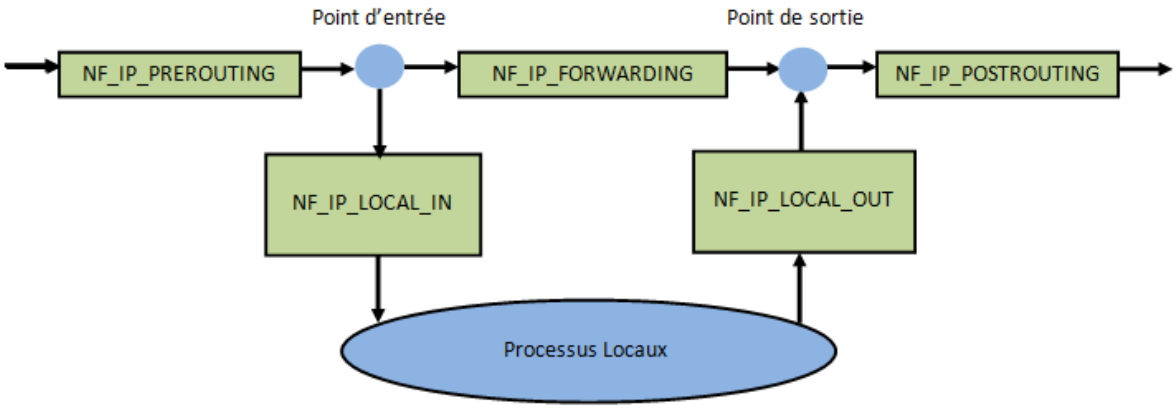


Figure 8 Les hooks ou points d'accrochage

Sur la figure, les paquets entrent par la gauche, provenant de l'interface réseau en entrée. Une fois les différents tests de conformité passés (c'est-à-dire pas de troncature, somme de contrôle IP exacte, etc...) et une éventuelle défragmentation effectuée, les paquets sont passés au point d'entrée ***NF_IP_PRE_ROUTING.***
Puis ils traversent le code assurant le routage, qui décide s'ils sont destinés à une autre interface, ou bien à la destination d'un processus local (SMB, FTP, http etc.). Il se peut tout de même que le code de routage supprime les paquets non routables.
Si le paquet est destiné à un processus local, le point d'entrée ***NF_IP_LOCAL_IN*** est sollicité avant que le paquet ne lui soit adressé.
Si le paquet est destiné à une autre interface, le point d'entrée ***NF_IP_FORWARD*** est appelé. Comme on peut le déduire, ceci consiste le trajet des paquets qui traversent la machine sur laquelle est installé le firewall dans sa fonction de routeur.
Enfin, le paquet atteint le point d'entrée ***NF_IP_POST_ROUTING***, avant d'être réinjecté sur le réseau à travers l'interface de sortie.
Le point d'entrée ***NF_IP_LOCAL_OUT*** lui, est appelé pour des paquets crées par les processus locaux.

On conclue que Netfilter se présente bien comme une série de 5 "hooks", sur lesquels des modules de traitement de paquets vont se greffer. Ces points sont:

- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_FORWARD
- NF_IP_POSTROUTING
- NF_IP_LOCAL_OUT

Ces cinq points d'insertion présentent les points d'accès ou les points de traversée de paquets dans la couche réseau linux. Netfilter dispose de **fonctions de rappel (callback).** Celles-ci sont des suites d'instructions qui précisent ce qui doit être fait lorsque survient un événement.

Concrètement, lorsqu'un paquet réseau atteint un de ces points d'accès, il est passé à Netfilter par l'intermédiaire de sa fonction de rappel. Il est alors examiné pour prendre une décision concernant son traitement futur.

Netfilter se comporte comme un automate qui compare le paquet successivement à plusieurs règles. Et selon le résultat du test, le paquet est traité ou transmis au test suivant.

3. Fonctionnalités

La caractéristique essentielle de Netfilter responsable de sa renommé est qu'il ne s'agit pas d'un simple dispositif de filtrage de paquets. Il permet entre autre de faire à peu près tout ce que l'on souhaite sur les paquets qui traversent la pile TCP/IP de Linux et c'est ce qui le rend très souple.
Pour ce qui est des fonctionnalités disponibles de Netfilter, elles sont fonction des modules chargés par le noyau comme elles peuvent également être compilées avec le noyau. Comme nous l'avons vu précédemment, Netfilter fait partie intégrante du noyau. Les fonctionnalités sont disponibles sous forme de modules que l'on charge avant d'effectuer le filtrage.
Nous aborderons dans ce qui suit les différentes fonctionnalités offertes par Netfilter.

3.1 Le Filtrage de paquets:

Dans son fonctionnement le plus simple, Netfilter permet de rejeter ou de laisser passer les paquets qui entrent et qui sortent du réseau mais aussi le contrôle des machines qui peuvent se connecter, sur quels ports, de l'extérieur vers l'intérieur, ou de l'intérieur vers l'extérieur du réseau. Le pare-feu permet aux administrateurs de filtrer le trafic réseau simple grâce à un mécanisme de sélection de paquets réseaux. Les paquets seront analysés par rapport à leurs caractéristiques en : adresses IP source/destination, numéro de port source /destination, le protocole associé, le trajet qu'il prend dans la structure Netfilter, l'interface par laquelle il est entré et l'interface par laquelle il sort. Grace à cette analyse, le pare-feu saurait alors quelle règle appliquer sur le paquet en faisant la correspondance entre ces paramètres et la règle correspondante.

Un point qui fait la puissance de Netfilter est la possibilité d'un filtrage de paquets **avec** ou **sans état** (stateless /stateful filtering). En effet, il est capable de caractériser les paquets selon le critère d'appartenance à des flux existants permettant donc au filtre de mettre en place dynamiquement des règles pour autoriser les paquets de retour ou des flux en relation avec des flux existants.

Avec Netfilter, il devient possible de déléguer le choix du destin des paquets à une application tournant en mode utilisateur. En effet, la programmation en espace utilisateur est nettement moins compliquée que la programmation noyau, le code est plus maintenable et plus facile à étendre ; il est bien plus facile de déboguer un module de filtrage en espace utilisateur. Seul inconvénient, le filtrage en espace utilisateur, bien qu'extensible à l'infini de façon simple, est moins performant que celui qu'accomplirait un module équivalent en espace noyau. Et la rapidité de traitement dépend bien sûr de la complexité du programme de filtrage

3.2 NAT :

Dans un datagramme, en plus des données, on trouve également quelques informations concernant le protocole utilisé et des identificateurs de l'émetteur et du destinataire. Ce sont ces identificateurs qui nous intéressent :

- L'adresse IP du destinataire.
- Le port du service utilisé sur le destinataire.

Ces informations constituent un "socket", elles sont indispensables pour arriver à joindre le bon service sur le bon serveur.

- L'adresse IP de l'émetteur.
- Le port de réponse.

Ces informations constituent un autre "socket", elles sont indispensables pour que l'émetteur d'un paquet puisse espérer recevoir une réponse.

Avec les fonctions NAT de Netfilter, il serait possible de modifier l'adresse de l'émetteur ou le port de l'émetteur ou les deux. Nous pouvons aussi changer l'adresse du destinataire, ou le port du destinataire, ou les deux. Il termine par ensuite effectuer la modification inverse sur le paquet en réponse arrivant dans l'autre sens.

On retrouve dans Linux deux types de NAT : Fast-NAT ou Netfilter-NAT. Fast-NAT est implémenté dans le code du routage IP du noyau Linux, tandis que Netfilter-NAT est aussi implémenté dans le noyau, mais à l'intérieur du code Netfilter.
Fast-NAT est généralement appelé par ce nom, car il est plus rapide que le code NAT netfilter. Il ne garde pas la trace des connexions. Le traçage de connexion prend de la puissance processeur, et le ralentit, ce qui est une des principales raisons pour laquelle Fast-NAT est plus rapide que Netfilter-NAT.
Cependant, le mauvais côté de Fast-NAT est qu'il ne trace pas les connexions, ce qui indique qu'il ne sera pas capable de faire du NAT sur des protocoles complexes comme FTP, IRC et d'autres fonctions aussi que Netfilter-NAT est capable de faire très bien.

3.2.1 Types de NAT :

On distingue deux types différents définis dans Netfilter : le NAT de Source (SNAT) et le NAT de Destination (DNAT).

- **Le SNAT (Le Nat de source)**

C'est lorsqu'on modifie l'adresse source du premier paquet c'est-à-dire de changer le lieu dont est issue la connexion. Le NAT de source est toujours réalisé après le routage, juste avant que le paquet ne soit envoyé sur la ligne.

Le cas d'utilisation du SNAT le plus fréquent est **le camouflage** d'adresse IP, `**masquerading**' en anglais. C'est le fait de faire correspondre différentes adresses IP à une unique adresse IP. Dans le cas d'un réseau Intranet, cela permet aux machines de ce réseau disposant d'adresses et ne sont ni uniques ni routables à l'échelle d'Internet, de communiquer avec le reste d'Internet en faisant semblant d'utiliser des adresses externes uniques et routables, donc l'adresse IP attribuée au routeur/ point d'accès du réseau.

Ainsi, il est possible de faire correspondre une seule adresse externe publique visible sur Internet à toutes les adresses d'un réseau privé, afin de pallier l'épuisement des adresses IPv4.

- **Le DNAT (Le NAT de destination)**

C'est lorsqu'on modifie l'adresse de destination du premier paquet, c'est-à-dire de changer le lieu où la connexion va aboutir. Le NAT de destination est toujours effectué avant le routage, aussitôt que le paquet arrive sur (la ligne) l'interface réseau.

Les cas d'utilisation de DNAT les plus fréquents sont les suivants :

- **Proxy Transparent :**

Il s'agit d'un programme qui se situe entre le réseau à protéger (privé) et le monde extérieur, établissant la communication entre les deux. Appelé aussi mandataire transparent, c'est quand on veut faire croire que chaque paquet entrant dans la machine où est le proxy, est destiné à un processus local de cette dernière. On utilise le DNAT pour modifier éventuellement l'adresse IP de destination à fin de faire le routage du paquet vers la machine à celle qu'il est réellement destiné.

Le réseau extérieur ne sait pas qu'il communique avec un proxy mais plutôt avec un hôte distinct du réseau privé, d'où la référence à la transparence.

- **Port Forwarding :**

La redirection de ports en français, consiste à rediriger les paquets réseaux destinés à un port donné d'une machine vers le même port mais sur une autre machine. Cela permet entre autres de proposer aux machines du réseau extérieur d'accéder à des services répartis sur plusieurs ordinateurs de ce réseau.

3.3 Modification des paquets :

En plus des fonctionnalités vues plus haut, Netfilter offre la possibilité de modifier les données des en-têtes de paquets IP avant ou après le processus de routage.

Nous avons déjà discuté du NAT, où nous avons vu qu'on peut modifier un paquet en modifiant les champs Adresse IP source et Adresse IP de destination de l'en-tête IP. Ce brassage de paquets se fait uniquement avec NAT et fait partie du processus NAT.

Par contre ici, on parle de modifier seulement les deux champs suivants de l'en-tête IP :

- **TOS :** il indique le type de service. Il indique la façon selon laquelle le datagramme (le paquet) doit être traité. Il sert à préciser le traitement effectué sur celui-ci pendant sa transmission à travers le réseau.

- **TTL :** il indique la durée de vie d'un paquet sur le réseau. Chaque processus qui touche le paquet doit supprimer un point du champ TTL, et si le TTL atteint 0 , le paquet doit être détruit ou écarté.

3.4 Suivi de connexions

Le pare-feu Iptables est un pare-feu à état, ce qui signifie que les paquets sont évalués en fonction de leur relation avec les paquets précédents du même flow de donnés. Les fonctionnalités de suivi des connexions construites au-dessus de la structure Netfilter permettent à Iptables de visualiser les paquets dans le cadre d'une connexion ou d'une session en cours plutôt que sous la forme d'un flux de paquets discrets et non liés.

Le suivi de connexion, une notion qui intervient aussi bien dans le filtrage que dans la translation d'adresses, est un mécanisme qui permet d'établir des liens de cause à effet entre les paquets qui passent dans la pile ; c'est-à-dire, durant la manipulation des paquets, il procède à conserver l'état d'une connexion en cours. Suivant son état, il serait en mesure de réagir intelligemment sur une connexion donnée en se référant aux règles de filtrages établies auparavant, mais aussi de détecter toute incohérence qui pourrait s'introduire lors des échanges IP.

En pratique, le Kernel maintient une table qui enregistre toute connexion établie par la machine dans le fichier virtuel /proc/net/ip_conntrack. Dès qu'un paquet arrive sur l'interface réseau, le module ip_conntrack responsable du suivi de connexion, est chargé dynamiquement et

vérifie l'entête IP pour indiquer s'il s'agit d'un paquet appartenant à une session qu'il reconnait ou bien une autre nouvelle session.
Selon le cas, il attribut une sorte d'étiquette au paquet désignant l'état de connexion auquel il appartient. Ainsi, quatre états sont possibles :

- **NEW (Nouvelle)**: Si le paquet n'appartient à aucune des précédentes connexions mais d'une nouvelle session.

- **ESTABLISHED (Etablie)** : Si le paquet fait objet d'une connexion établie auparavant au niveau protocole. Exemple : une session TCP.

- **RELATED (Associée):** Si la connexion est en relation avec une autre connexion déjà établie. L'exemple du PONG qui suit une requête PING.

- **INVALID (Invalide):** Quand le paquet n'appartient à aucune des trois catégories précédentes. La connexion n'est donc pas conforme et contient un jeu de flags anormal. Ceci est particulièrement intéressant s'il l'on dispose d'une installation privée et qu'on voudrait interdire à toute connexion « NEW » d'entrer dans notre installation depuis un réseau extérieur, parce que effectivement, il n'y a aucune raison qu'il en rentre si nous n'avons pas de serveur.
  Les paquets INVALID pourront éventuellement être tracés dans **les logs.**

3.4.1 Cas d'une connexion TCP :

L'établissement d'une connexion TCP suit un protocole strict comme suit :

• Une requête de synchronisation [SYN] de la part de l'initiateur du dialogue (le client),

• une réponse d'accusé de réception de la synchronisation [SYN,ACK] de la part du serveur,

• un accusé de réception du client [ACK] est retourné au serveur
Pour résumer :

- Lorsqu'un paquet TCP contient le flag **SYN**, c'est que c'est une nouvelle connexion qui Commence.
- lorsqu'un paquet TCP contient les flags S**YN et ACK**, c'est que la connexion est acceptée, elle est donc établie.
- lorsqu'un paquet ne contient que le flag **ACK**, c'est que la connexion est en continue.

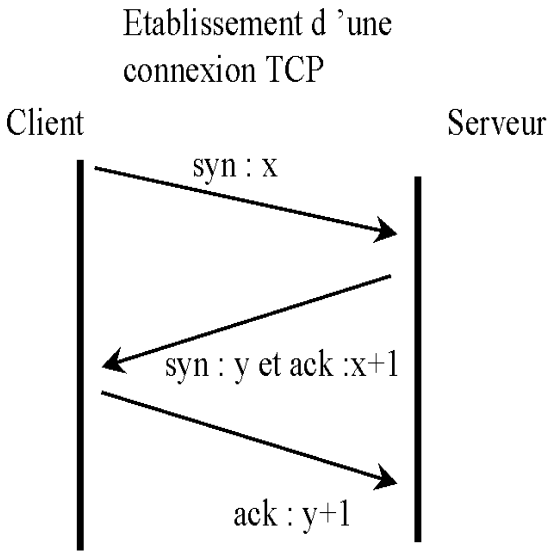


Figure 9

3.4.2 Cas du protocole UDP :

Dans ce cas la, c'est plus délicat puisqu'il n'y a justement pas de connexion. Il sera donc impossible de définir de façon précise l'état d'un échange UDP. La méthode convenable de faire est de mettre en place un "**timer**" pour décider de l'état d'un paquet UDP.

Nous pouvons prendre l'exemple simple d'une requête DNS depuis notre réseau privé.
Le premier paquet UDP sort de notre réseau, sur un port connu et identifié (53 par exemple) vers un serveur DNS. Nous pouvons décider de le laisser passer et nous le qualifions de "NEW". Il déclenche un timer. Si avant expiration du timer, nous recevons un paquet UDP du serveur DNS, nous considèrerons que c'est un paquet "ESTABLISHED".

3.5 Autres additions

Netfilter est extrêmement modulaire. Il existe un module noyau ou une bibliothèque dynamique par protocole, prenant en charge l'ensemble des opérations relatives à un protocole donné : UDP, TCP, ICMP, adresses MAC etc.

Cette flexibilité fournit une opportunité d'ajouter de nombreuses fonctionnalités, et de manière plus importante permet aux gens d'écrire des améliorations ou des remplacements complets.

4. Iptables :

Comme précisé, Netfilter n'est que le nom générique du paquetage du pare-feu et Iptables représente la commande d'administration qui permet de le manipuler. Cette commande s'appuie sur un certain nombre de modules noyau et de bibliothèques dynamiques correspondant à l'ensemble des protocoles gérés.

Le module Netfilter traite tout le trafic IP et filtre ou modifie le trafic en fonction d'un ensemble de **règles** qu'on serait en mesure d'ajouter nous-mêmes à l'aide de l'outil Iptables qui opère en espace utilisateur. L'ensemble des règles est configurables à l'exécution et sont stockées dans **des tables** prédéfinies dans Netfilter. Ces tables elles, regroupent les règles en ce qu'on appelle **des chaines**, chacune remplissant une fonction bien spécifique.

En principe, l'ensemble de ces commandes écrites au niveau d'Iptables vont traduire ce qu'on souhaite réaliser avec notre firewall, tel que l'autorisation aussi bien que le rejet de certains paquets venus sur le réseau et pas mal d'autres fonctionnalités. Mais avant d'aller plus loin, il est impératif de bien s'introduire à ces notions particulières à Iptables qu'on vient de citer : Les tables, les chaines et les règles.

4.1. Les tables et les chaines:

4.1.1 La table :

La notion de table est utilisée pour organiser les règles du pare-feu. Celles-ci sont classifiées suivant la fonction qui sont chargées de faire. Ainsi, si une règle est relative au NAT, elle sera placée dans une table bien précise. De même, si celle-ci constitue une règle de filtrage de paquet, elle sera alors placée dans l'endroit approprié aux règles de filtrage.

Il existe dans Netfilter des tables prédéfinies qui correspondent aux principales fonctionnalités vues plus haut, à savoir:

- La table « **FILTER** » :

  La table filter est plus au moins la table la plus utilisée des tables d'Iptables. Cette table ne devrait jamais modifier les paquets, seulement les filtrer. Elle s'accroche dans Netfilter aux points **NF_IP_LOCAL_IN, NF_IP_FORWARD**, et **NF_IP_LOCAL_OUT.** Cela veut dire que pour un paquet donné, il n'y a qu'une et une seule place pour le filtrer ce qui rend les choses plus simples.

- La table « **NAT** » :

  Elle contient les règles relatives à la translation d'adresses NAT. La structure de Netfilter fournit quatre points d'entrée pour y parvenir : pour les paquets non issus de processus locaux, les points d'entrée NF_IP_PRE_ROUTING et NF_IP_POST_ROUTING sont utilisés pour altérer les renseignements concernant respectivement la destination et la source ; les points d'entrée NF_IP_LOCAL_OUT et

NF_IP_LOCAL_IN remplissent la même fonction pour les paquets issus de processus locaux.

Cette table est légèrement différente de la table `filter', effectivement, seulement le premier paquet d'une nouvelle connexion traverserait la table : le résultat de cette traversé est ensuite appliquée à tous les futurs paquets qui font partie de la même connexion.

- La table « **Mangle** » :

La table Mangle permet d'apporter des changements concrets des informations d'un paquet. Elle est utilisée pour altérer les en-têtes IP des paquets en plusieurs manières. On pourrait par exemple, ajuster la valeur « TTL » ou bien la durée de vie d'un paquet. En outre, elle offre la possibilité de marquage des paquets entrants et ceux générés localement**.** Le marquage de paquets permet un traitement spécifique des paquets marqués dans les tables de routage. Cependant, ce marquage ne touche pas au contenu du paquet mais plutôt à ses caractéristiques représentées en Kernel.

La table mangle s'accroche aux 5 hooks de Netfilter et ce à partir du noyau 2.4.18. En effet, les noyaux précédents n'avaient pas la table Mangle accrochée à tout les hooks.

- La table **RAW** :

La table RAW a une fonction particulière. Son seul but est de fournir un mécanisme de marquage de paquets qui ne doivent pas être vérifiés par le système de suivi de connexion.
La table Raw et ses chaînes sont utilisées avant tout autre table dans Netfilter. Cette table modifie les paquets entrants avant qu'ils atteignent les autres sous-systèmes de Netfilter au hook NF_IP_PREROUTING. De même les paquets générés localement seront altérés avant qu'ils atteignent les autres sous-systèmes de Netfilter au hook NF_IP_LOCAL_OUT. Ces emplacements sont le seul endroit où l'on pourrait opérer sur les paquets avant qu'ils soient vérifiés par le module de suivi de connexion.

4.1.2 Les chaines :

Netfilter intercepte les paquets réseau à différents endroits du système (à la réception, avant de les transmettre aux processus, avant de les envoyer à la carte réseau, etc. Les paquets interceptés passent à travers des *chaînes* qui vont déterminer ce que le système doit en faire. En modifiant ces chaînes on va pouvoir bloquer certains paquets et en laisser passer d'autres. Cette notion, très pratique avec Iptables, permet de définir un filtrage commun à tous les paquets qui les traversent.

Les chaines permettent aux utilisateurs de contrôler l'endroit sur le trajet du paquet, où est ce qu'une règle serait évaluée.

- **Les chaines prédéfinies (les builtins) :**

Le traitement qu'est appliqué au niveau de Netfilter peut différer d'un paquet à un autre suivant le chemin emprunté par celui-ci à travers la structure de Netfilter. Hors, trois trajectoires sont possibles :

- **PREROUTING -> INPUT** : le chemin des paquets qui viennent de l'extérieur vers la couche applicative de la machine en question.
- **OUTPUT -> POSTROUTING:** le chemin des paquets qui viennent de la couche applicative de la machine en question vers l'extérieur.
- **PREROUTING -> FORWARD -> POSTROUTING** : le chemin des paquets qui viennent de l'extérieur, qui passent par la couche réseau de la machine et qui repartent vers l'extérieur.

Les chaînes, à cet égard, correspondent à l'endroit où la pile de Netfilter a été invoquée. On retrouve ainsi les cinq chaines prédéfinies correspondant aux hooks de Netfilter :

- **PREROUTING**
- **INPUT**
- **OUTPUT**
- **FORWARD**
- **POSTROUTING**

Ces chaines sont créées nativement (builtin) dans le code Netfilter, on ne pourrait donc pas les effacer.

- **Les chaines définies par l'utilisateur (user-defined) :**

Il est tout aussi possible de créer ses propres chaines dans Iptables, généralement dans le but de simplifier le traitement. Par exemple, si l'on souhaite créer un filtrage détaillé pour le protocole ICMP, nous pouvons créer une chaine ICMP dans laquelle nous ajouterons toutes les règles de filtrage que l'on souhaite appliquer. Depuis une autre chaine telle que FORWARD ou INPUT, nous ajouterons alors une règle informant que si le protocole est de type ICMP, Iptables devra se référer à la chaine ICMP.

4.1.3 Quelle chaine dans quelle table :

Maintenant que nous avons parlé de la notion de table et celle de chaine séparément, voyons voir quelles chaines contient chacune des tables d'Iptables.

La table	Les chaines implémentées
FILTER	- INPUT - OUTPUT - FORWARD
NAT	- PREROUTING - POSTROUTING - OUTPUT
MANGLE	- PREROUTING - INPUT - OUTPUT - FORWARD - POSTROUTING
RAW	- PREROUTING - OUTPUT

4.2 Les règles Iptables et les cibles:

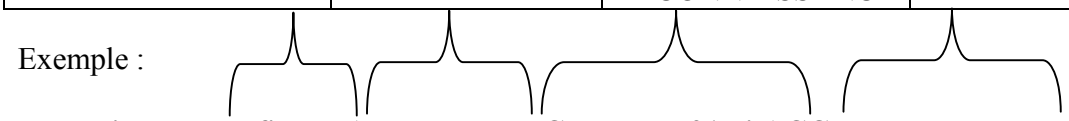
Chaque règle spécifie des critères de sélection (adresse source, protocole, etc.) et ce qui doit advenir des paquets correspondants (rejet, autorisation, archivage, etc.). Les critères disponibles et les actions envisageables ne sont pas forcément les mêmes suivant la chaîne dans laquelle on ajoute la règle. Suivant l'action choisie, le paquet parcourt ou non les règles suivantes de la chaîne. Si aucune règle ne correspond au paquet, c'est la politique par défaut de la chaîne qui décide du sort du paquet.

Pour savoir quelle action serait appliquée sur un paquet donné, Netfilter compare les critères de ce paquet aux critères des règles, ce qu'on appelle « un matching ». Si le paquet satisfait les critères d'une règle, l'action que désigne celle-ci lui serait alors appliquée. Le terme utilisé dans Iptables pour l'action à effectuer sur un paquet est « Cible ».

Les règles implémentées dans Iptables sont responsables de la performance du firewall donc de sa qualité. En effet, le rendu global du pare-feu n'est rien d'autre que l'effet accumulatif des règles définies par l'utilisateur lui-même.

Une règle serait conçue suivant le modèle suivant :

TABLE	CHAINE	MOTIF DE RECONNAISSANCE	CIBLE



Exemple :

iptables –t filter –A INPUT –p TCP –dport 21 –j ACCEPT

- **Les cibles (Targets) :**

Cible ACCEPT	Elle indique que la règle appliquée est acceptée. Sinon la règle ne traversera pas la chaine ou aucune autre chaine dans la même table. Cependant un paquet qui a été accepté dans une chaine peut toujours circuler à travers les chaines des autres tables. Pour utiliser cette cible on utilise **–j ACCEPT**
Cible DROP	Elle efface les paquets et n'effectue aucun processus supplémentaire. Un paquet qui existe dans une règle dont l'action est DROP, sera bloqué. En d'autres termes, le paquet est mort.
Cible REJECT	Celle-ci fonctionne à la base comme la cible DROP, mais elle renvoie un message d'erreur à l'hôte qui a envoyé le paquet. REJECT n'est valide que dans les chaines : INPUT, OUTPUT et FORWARD. L'option utilisé pour définir cette cible est **--reject-with** en indiquant la réponse qui serait envoyé à l'hôte.
Cible LOG	Elle est spécialement destinée à journaliser des informations détaillées sur les paquets. La cible LOG renverra une information spécifique sur les paquets, comme les en-tetes IP et d'autres détails intéressants.
Cible DNAT	Comme son nom l'indique, elle est utilisée pour la traduction d'adresses IP de destination d'un paquet. Si un paquet est apparié et qu'il est la cible de la règle, ce paquet et tous les autres du même flux seront traduit et ensuite routés vers l'hôte approprié.
Cible SNAT	Comme la cible DNAT, cette cible est utilisée pour la traduction d'adresses sources. Elle est disponible que dans la chaine POSTROUTING de la table NAT.

Cible REDIRECT	Elle est utilisée pour rediriger les paquets et les flux vers la machine elle-même. La cible REDIRECT pourrait être particulièrement utile quand on veut faire un proxy transparent. Elle n'est pas valide que dans les chaines PREROUTING et OUTPUT de la table NAT.
Cible MASQUERADE	Elle est utilisée comme la cible SNAT. Seulement, on l'utilise seulement avec des adresses IP dynamiques. Si l'on dispose d'adresses IP statiques, on utilisera le SNAT. Elle est disponible que dans la chaine POSTROUTING de la table NAT.
Cible MARK	Elle sert à placer les valeurs de marquage Netfilter qui sont associés à des paquets spécifiques. Cette cible n'est valide que dans la table MANGLE.
Cible JUMP	Les règles peuvent être placées dans des chaînes définies par l'utilisateur de la même manière qu'elles peuvent être placées dans des chaînes prédéfinies. La différence est que les chaînes définies par l'utilisateur ne peuvent être atteintes qu'avec la cible JUMP. Elles ne sont pas enregistrées avec un hook Netfilter. Les chaînes définies par l'utilisateur agissent comme de simples extensions de la chaîne qui les a appelées. Les JUMP (sauts) sont des actions qui entraînent le passage de l'évaluation à une chaîne différente pour un traitement supplémentaire.

5. Conclusion

Lors de ce chapitre, on a exploré les différentes spécificités du Firewall Netfilter. Malgré son efficacité qui n'est pas à prouver on a remarqué que sa manipulation est plus au moins compliquée vue que l'écriture des règles du système doit se faire manuellement au niveau du terminal. Ce qui nécessite une bonne connaissance du système et du firewall pour parvenir écrire rigoureusement l'ensemble des règles de ce dernier.

<u>Chapitre IV</u> : Réalisation de notre Interface graphique pour le pare-feu Netfilter/Iptables

La configuration du pare-feu Netfilter/Iptables en tapant soit même les lignes de commandes sur un terminale d'une machine Linux demande des connaissances étalées dans le domaine informatique et plus particulière en réseau. Cela empêche donc des personnes n'ayant pas d'affinités avec les outils informatiques ou avec ce système d'exploitation de taper eux mêmes les règles Iptables visant à filtrer leurs réseaux afin de lui apporter une certaine sécurité. Ce qui nous a motivé à créer un espace plus convivial pour le pare-feu Netfilter/Iptables du Os Linux. Ce dernier renferme dans son noyau un système performant de filtrage et de gestion de flux réseau.

Notre application a donc pour but de fournir une interface graphique facile d'utilisation afin de palier au problème énoncé plus haut et de permettre à toute personne de configurer son pare-feu de façon plus aisée mais tout autant efficace, sans besoin d'être expert dans le domaine. Avant de commencer, l'utilisateur devra d'abord créer le fichier script à l'aide du bouton créer un fichier. Si l'utilisateur n'est pas à sa première visite, il continuera d'utiliser le fichier qui a déjà été créé. Puis il pourra accéder à l'interface du pare-feu, lui donnant la possibilité d'ajouter des règles de la table Filter, de définir sa politique par défaut, mais aussi aux tables NAT et Mangles pour les translations d'adresses et les changements d'en-têtes de paquets respectivement. Après avoir remplit les champs nécessaires pour l'action qu'il veut réaliser, l'utilisateur valide cette action en cliquant sur le bouton Sauvegarder, ce qui va sauvegarder dans le fichier script qu'il a créé, les lignes de la règle qu'il vient de formuler. Il cliquera en suite sur le bouton Terminer quand il aura fini pour que cette action soit exécutée par le système Linux.

1. Les exigences fonctionnelles et la modélisation structurelle du système :

Nous avons deux catégories de spécification des besoins. En effet, nous avons les **besoins fonctionnels** qui constituent les objectifs souhaités de l'application et nous avons les **besoins non fonctionnels** qui quant à eux constituent tout ce qu'il nous faut pour la réalisation de l'application.

L'objectif principal de notre application étant de rendre facile la manipulation d'Iptables, les critères suivants doivent donc être satisfaits :

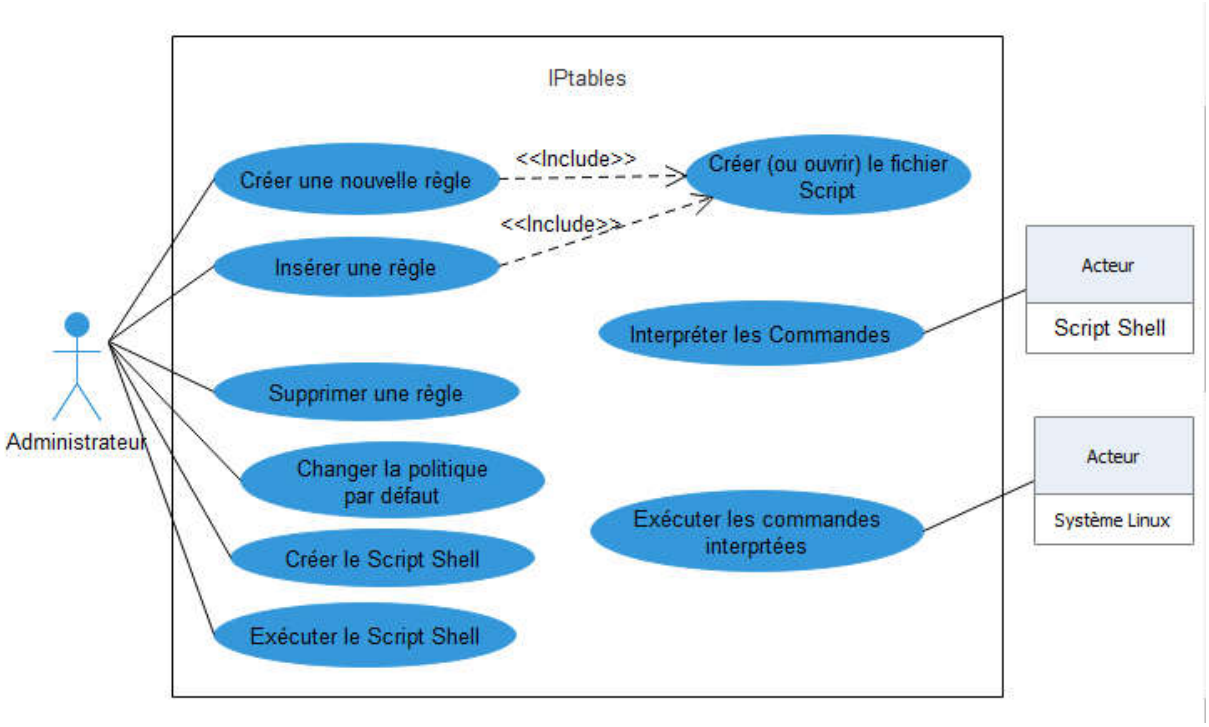
- L'application doit être indépendante, c'est à dire, pouvoir fonctionner sur tout type Linux sans ajout de packages mis à part la machine virtuelle java
- L'application doit être simple et facile d'utilisation afin que n'importe quel utilisateur puisse sen servir
- L'application doit aussi permettre la configuration de Netfilter en permettant l'ajout de nouvelles règles Iptables, la modification et la suppression de façon aisée

Dans les lignes qui suivent, nous illustreront tous ceci avec la norme UML. Celle-ci étant la notation commune pour la représentation de tout type de projets logiciels.

1.1. Le Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs d'un système. C'est pourquoi il est en occurrence le premier diagramme du modèle UML à construire, car il permet de recueillir, d'analyser et d'organiser les besoins. C'est le diagramme qui modélise les relations entres l'utilisateur et les objets que le système met en œuvre. Il est représente dans un cadre pour délimiter le système du monde extérieur.

Nous allons représenter dans la figure ci-dessous le diagramme de cas d'utilisation de notre application, qui contient les tâches qu'effectue chaque acteur (les entités externes agissant sur le système).



1.2. Le diagramme de classes :

Le diagramme de classe est en générale considère comme le plus important lorsqu'il s'agit d'un développement orienté objet. Car il représente et décrit la structure des classes que le système utilise, ainsi que les liens établies entre elles.

La figure ci-dessous représente le diagramme de classe de notre application.

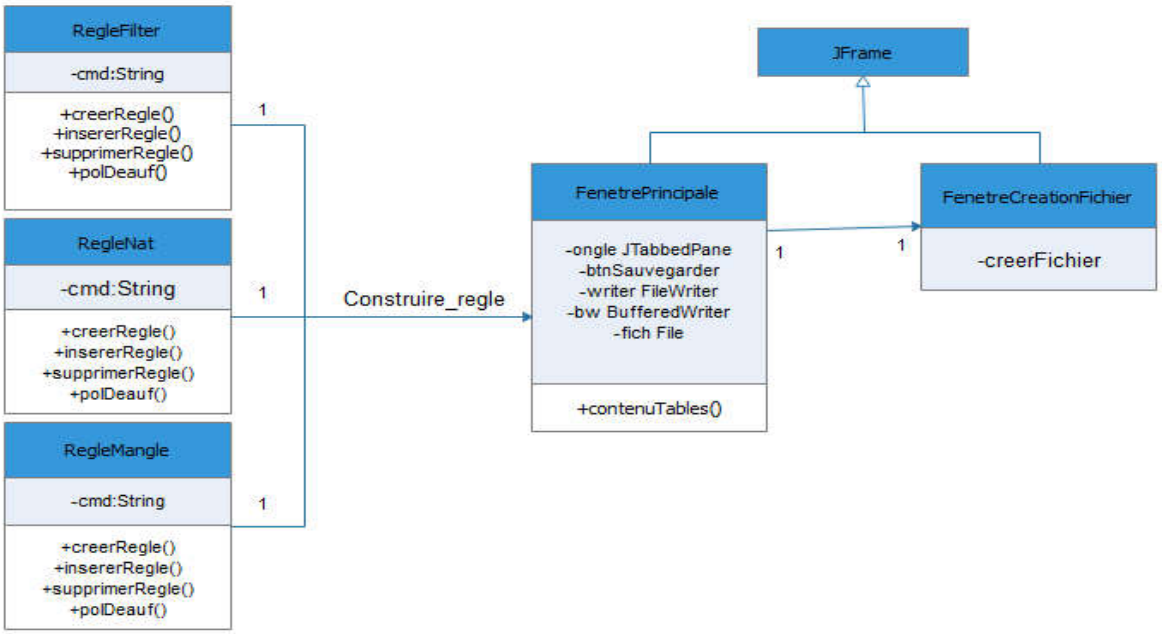


Figure 10 Diagramme de classe de notre application

2. La modélisation comportementale :

2.1. Le diagramme de séquence :

Le diagramme de séquence, quant a lui, permet de représente le schéma de collaboration qui existe entre les objets du système, dans l'ordre chronologique de la production des événements par ces objets.

Voilà ci-dessous le diagramme de séquence de notre application.

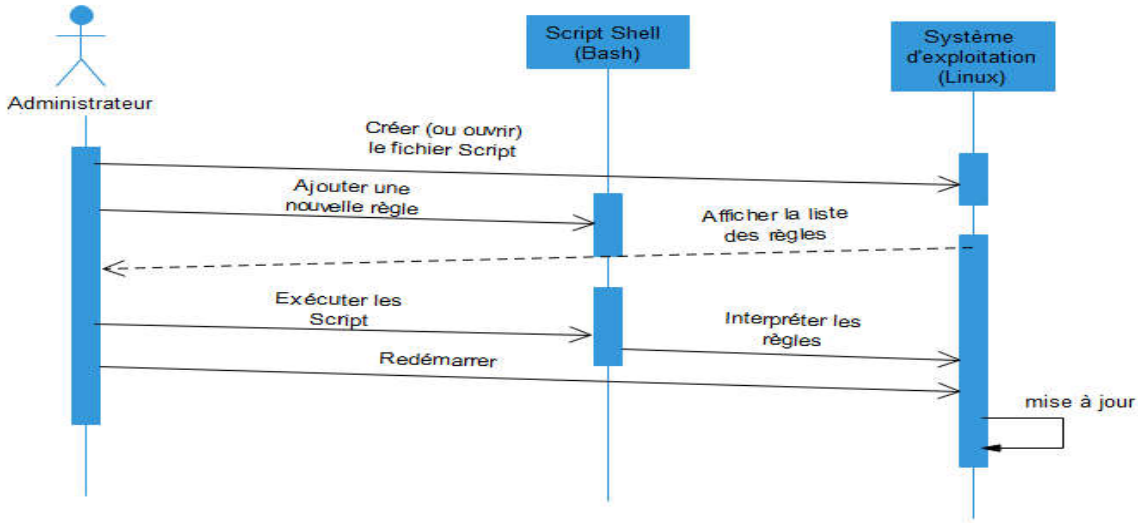


Figure 11 Diagramme de séquence pour l'ajout d'une règle

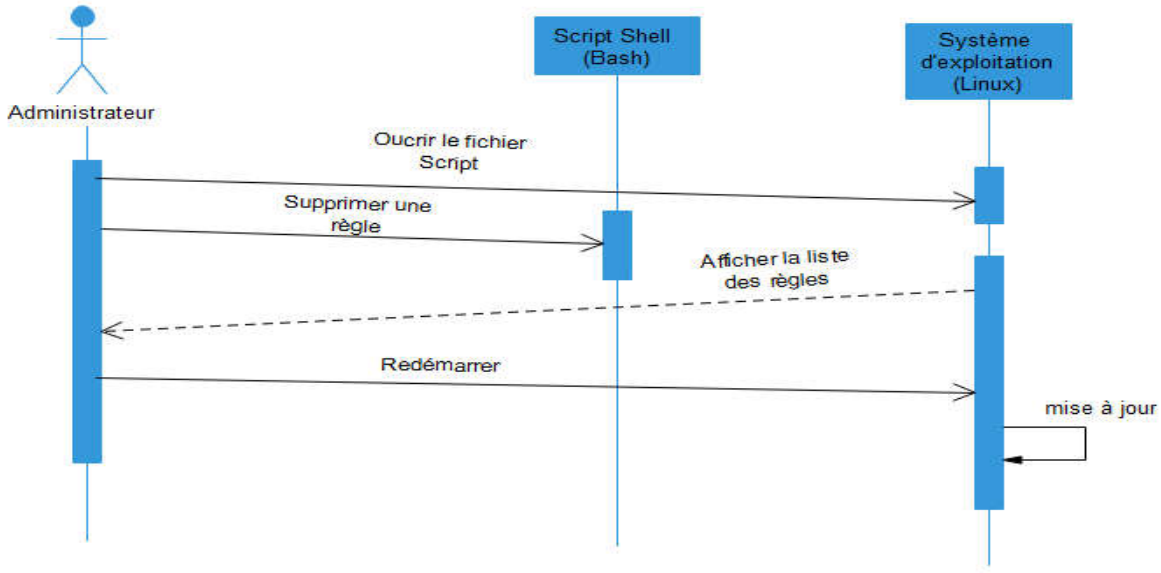


Figure 12 Diagramme de séquence pour la suppression d'une règle

3. Implémentation :

3.1. Outils utilisés :

Pour la réalisation de notre application, nous avons utilisé les matériels et logiciels suivant :

- Nos machines équipées du système d'exploitation Linux OpenS
- Eclipse comme environnement de développement intégré pur Java.
- Bash (Bourne Again Shell) comme script Shell pour interpréter nos commandes sous Linux.
- Edraw Max pour la réalisation de nos diagrammes UML.

3.2. Préparation du Script du pare-feu :

Le Shell est l'intermédiaire entre tout utilisateur et le système Linux, c'est lui qui est chargé de prendre les commandes de l'utilisateur, les interpréter en un langage compréhensible par le système et le lui transmettre.

Un Script est un programme qui peut être interprété par le Shell. Il contient donc les commandes Shell.

Il existe plusieurs types de Shell comme sh, bash, ksh, csh…etc.

Mais pour notre application, nous avons utilisé le bash (Bourne Again Shell) qui est le Shell par défaut de la plupart des distributions Linux.

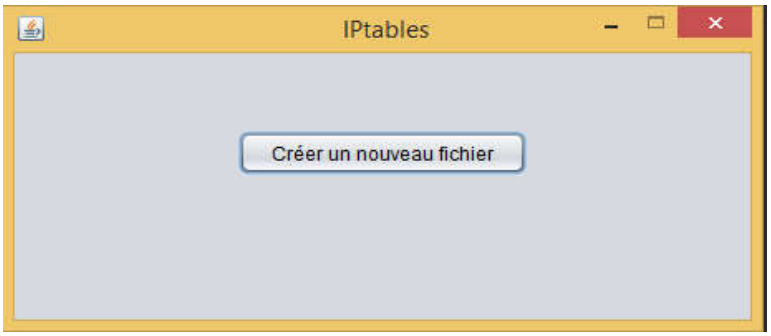
Notre Script pare-feu commence par les commandes initiales qui est **#!/bin/bash** C'est la première ligne de notre Script. Elle précise au système que le Shell utilisée est bash.

4. Gestion graphique du pare-feu :

Ici nous allons présenter les deux interfaces graphiques que nous avons réalisé pour Iptables :

4.1. L'interface d'accueil :

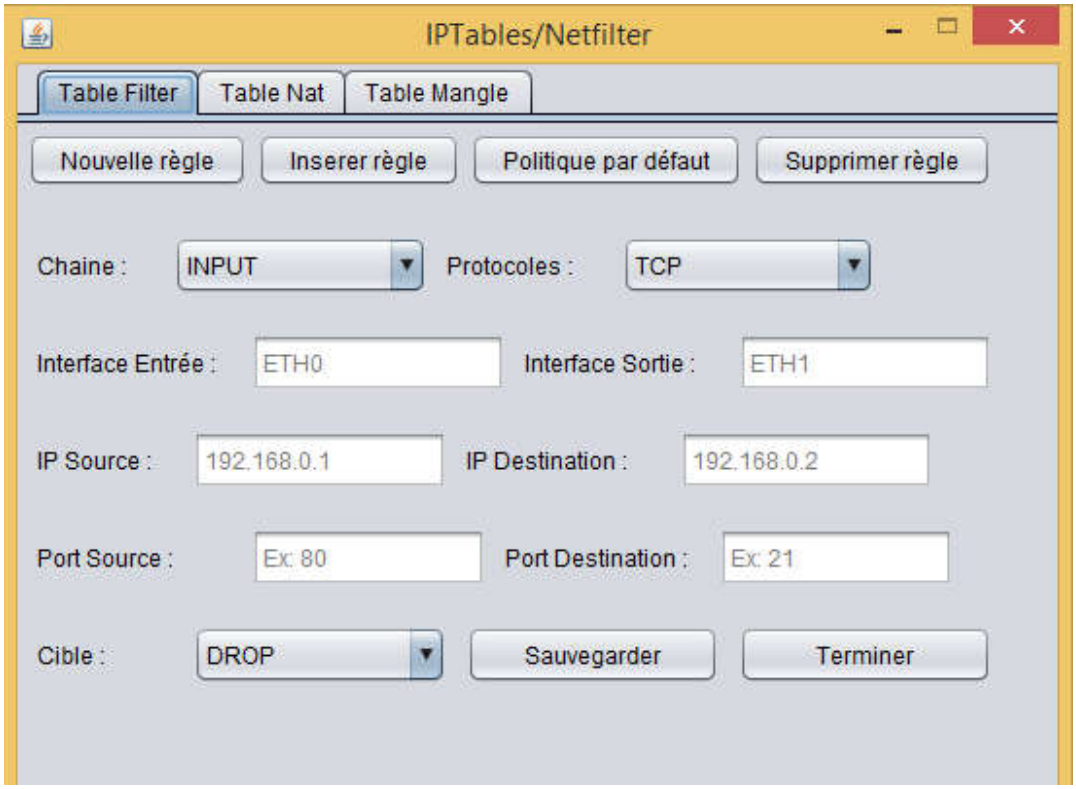
Est la première interface sur laquelle l'utilisateur doit créer un fichier Script en cliquant sur le bouton « Créer un nouveau fichier » si c'est la première fois qu'il utilise le pare-feu, ou rouvrir le fichier Script déjà crée.



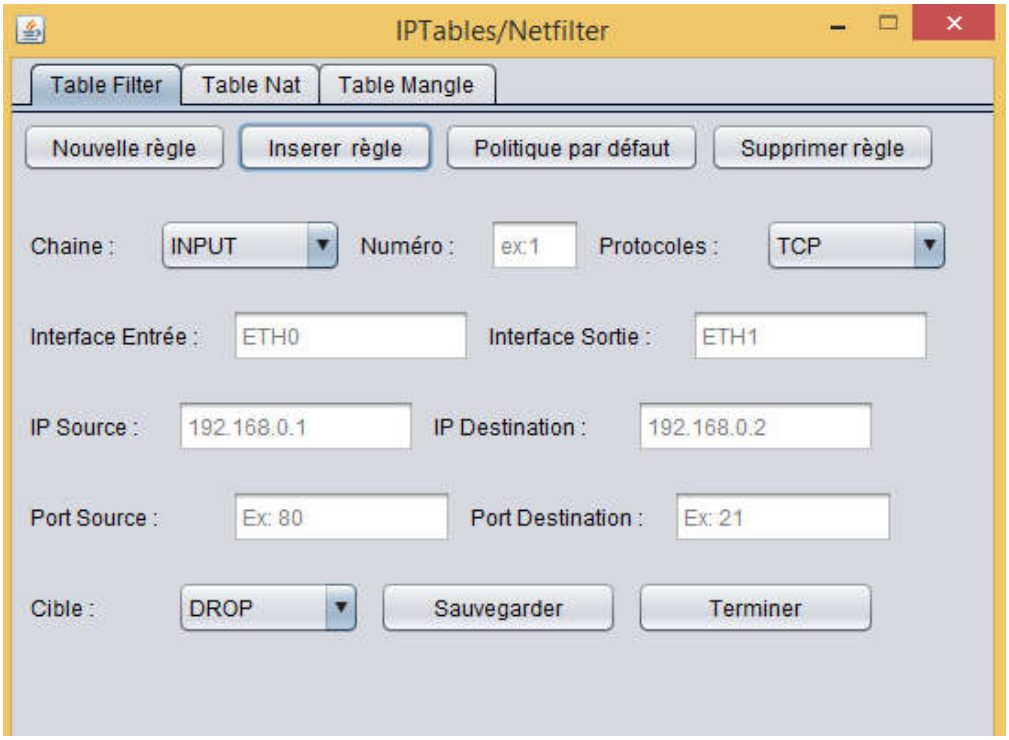
4.2. L'interface Principale :

Il contient trois onglets pour les trois tables Filter, Nat et Mangle. Pour chaque table nous avons les boutons « Nouvelle Règle » et « Politique par Défaut » et les boutons « Sauvegarder » et « Terminer » en Bas de l'interface.

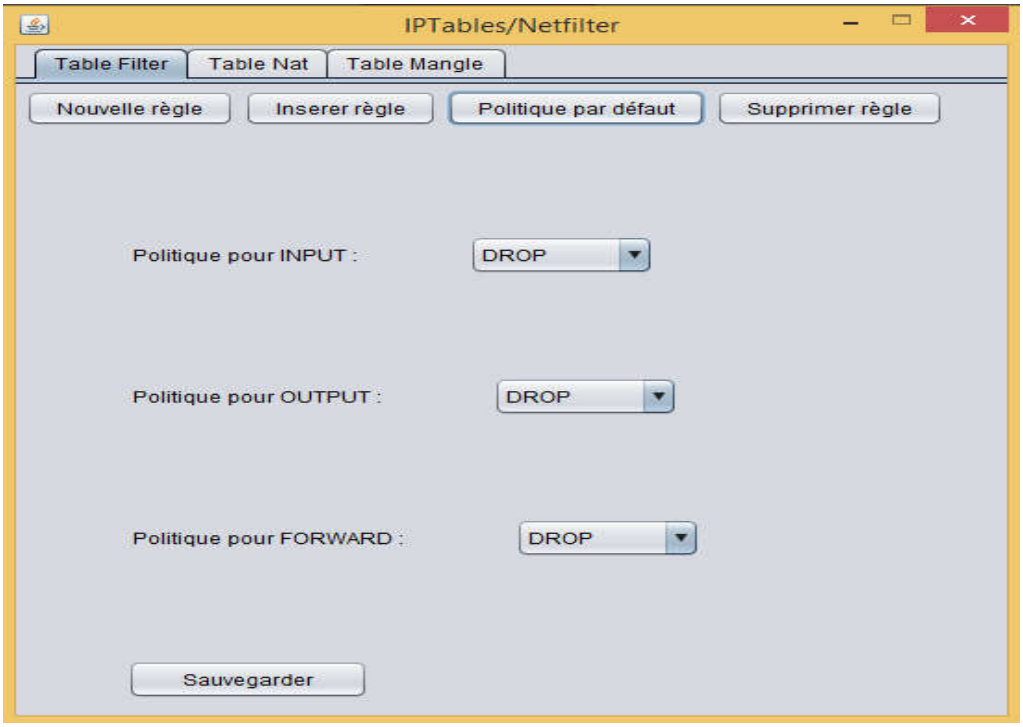
- Créer une nouvelle règle de la table Filter :



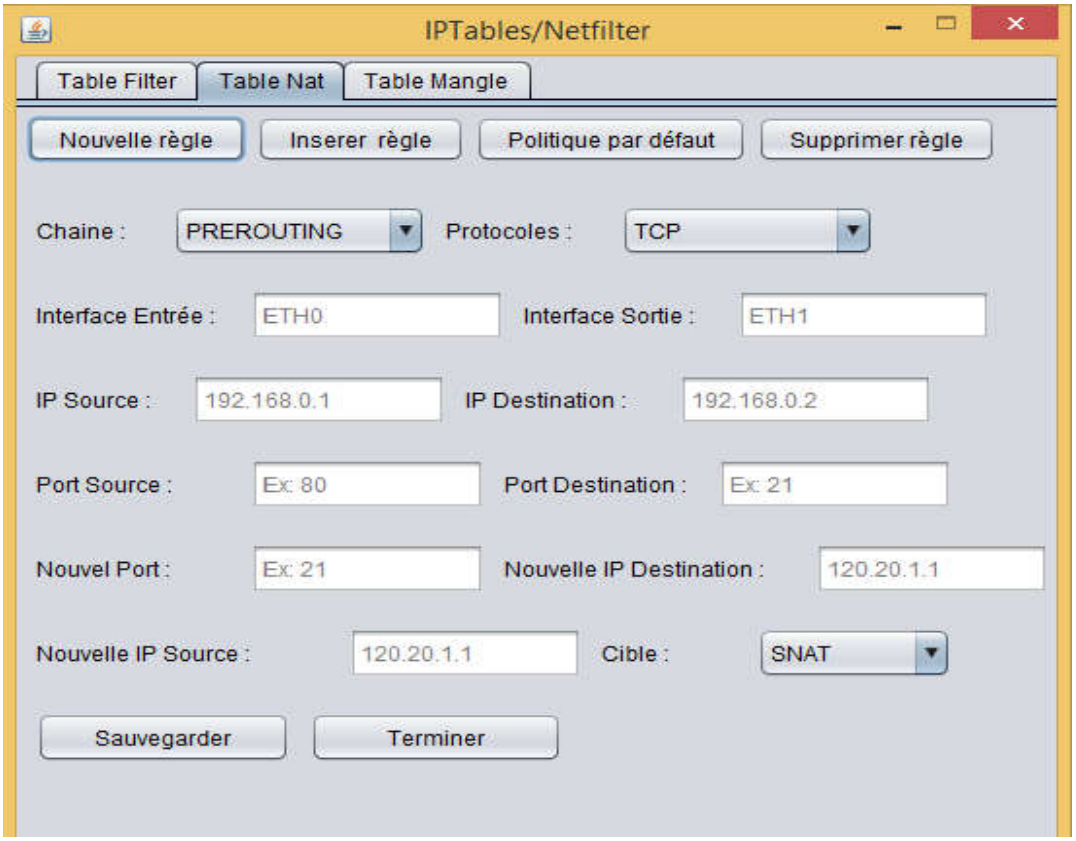
- Insérer une règle dans la table Filter : Cette règle sera insérée à un numéro donné par l'utilisateur dans le champ « Numéro »



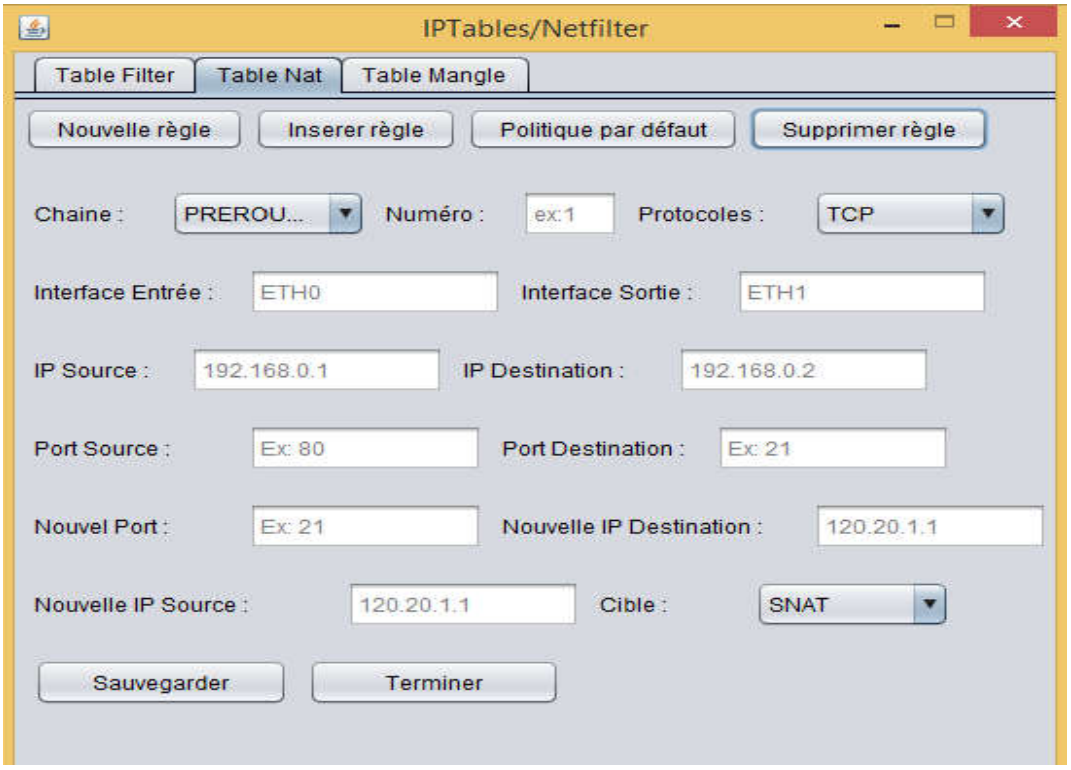
- Définir la politique par défaut pour la table Filter:



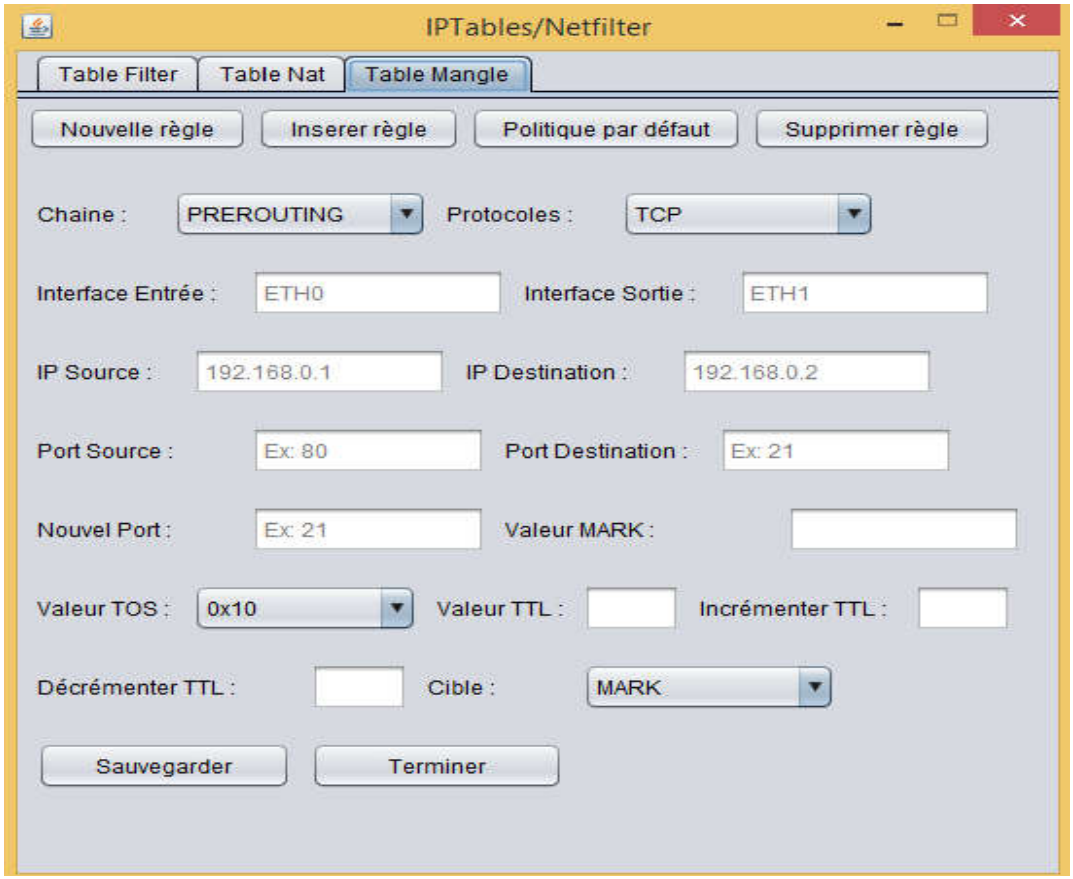
- Créer une nouvelle règle pour la table Nat :



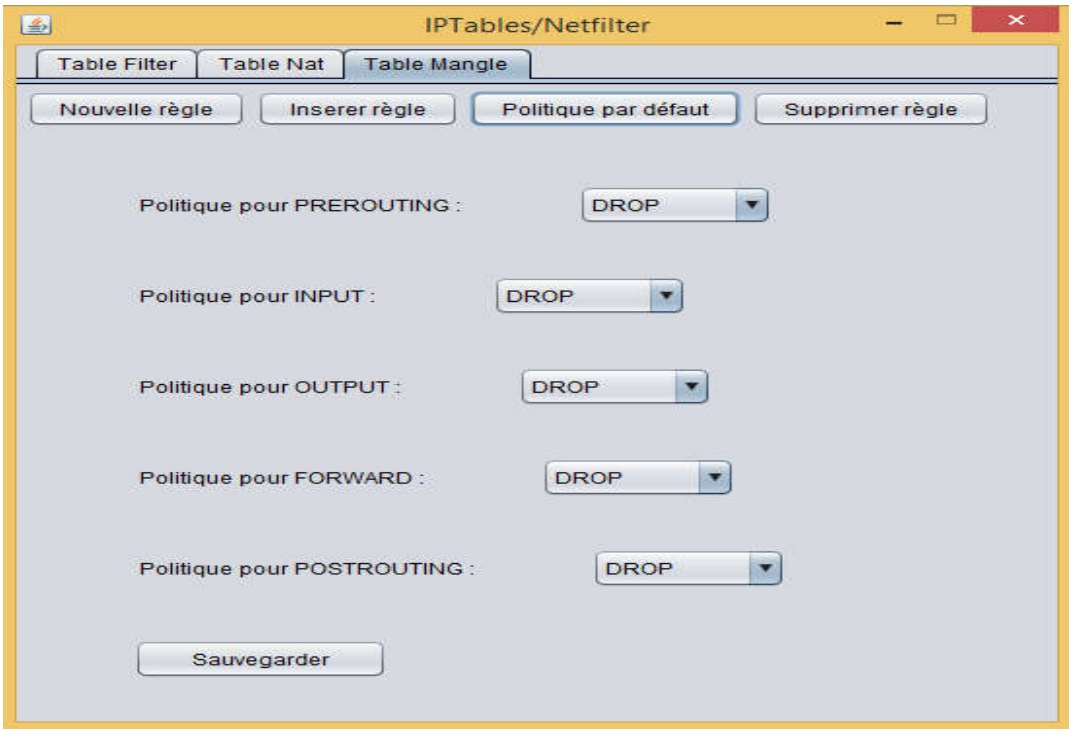
- Supprimer une règle de la table Nat : le numéro de la règle donné dans le champ « Numéro » sera supprimer ou alors la règle recopiée par l'utilisateur sera supprimée.



- Créer une nouvelle règle de la table Mangle :



- Définir la politique par défaut de la table Mangle :



CONCLUSION GENERALE

La complexité accrue de la technologie des réseaux informatiques fait de la sécurité réseau un challenge et une des dimensions essentielles dans la stratégie de développement dune entreprise. Elle mérite une attention particulière car des fuites d'informations pertinentes peuvent se révéler fatale pour une entreprise. Il est important de la prendre en compte dès la conception du système d'information et de penser aux mises à jour assez régulières.

Dans le cas de notre projet, nous avons cherché à rendre plus accessible a un publique plus élargie les mesures de sécurité que nous offre le système Linux. Le but de notre travail a été de développer un environnement plus convivial pour tout utilisateur y compris des non informaticiens, qui va permettre à ses utilisateurs de profiter de la sécurité qu'offre le pare-feu Netfilter Iptables de Linux car ce dernier contient dans la structure de son noyau un système de filtrage très performant et qui d'ailleurs est utilisé dans bon nombre de solutions de sécurité qui ont été proposées.
Pour commencer nous avons dans le premier chapitre, présenté une étude des différents types d'attaques les plus rencontrés dans les réseaux informatiques. Puis dans le deuxième chapitre, nous avons parlé de l'état de l'art sur les pare-feux, de leur fonctionnement et des différents types et architectures de pare-feux. Nous avons ensuite fait une étude sur le pare-feu Netfilter /Iptables de Linux et de son fonctionnement car c'est le noyau de notre projet. Et enfin dans le quatrième et dernier chapitre, nous avons représenté à travers des images illustratives notre application, sa structure interne et ses liens avec l'extérieur.

Nous pouvons conclure ce projet de fin d'étude en disant qu'il nous a permis non seulement de pratiquer ce qui nous avait été transmis durant notre formation, mais aussi d'acquérir de nouvelles connaissances qui nous seront sans doute utile dans la phase d'entrée en fonction en tant que diplômées de la spécialité Réseau, Mobilités et Systèmes Embarqués.

Comme perspective à notre travail, nous pensons que l'adaptation de la même démarche pour d'autres mécanismes de sécurité dont la convivialité est un handicap pourrait être bénéfique et profitable à un bon nombre d'utilisateurs.

Liste des figures :

Figure 1 : Attaque directe 6
Figure 2 : Attaque indirecte sur rebond 7
Figure 3 : Attaque indirecte par réponse 7
Figure 4 : Les couches du modèle OSI 9
Ref : workig.free.fr/OSI.chapter.html
Figure 5 : Le fonctionnement d'un pare-feu 14
Figure 6 : Représentation de l'architecture proxy 17
Figure 7 : Représentation de l'architecturé DMZ 18
Figure 8 : Les hooks ou points d'accrochage 23
Figure 9 : Etablissement d'une connexion TCP 28
Figure 10 : Diagramme de classe de notre application 37
Figure 11 : Diagramme de séquence pour l'ajout d'une règle 38
Figure 12 : Diagramme de séquence pour la suppression d'une règle 39

Bibliographie

[1] https://openclassrooms.com/fr/courses/1946106-securisez-votre-reseau-grace-aux-vpn-et-firewall

[2] https://securite.developpez.com/cours/

[3] Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort - **Michael Rash**

[4] Linux iptables Pocket Reference: Firewalls, NAT & Accounting - **Gregor N. Purdy**

[5] Solange Ghernaouti Ghernaouti-Hélie – Sécurité Informatique et Réseaux – Dunod

[6] Raymond Panko - Sécurité des Systèmes d'information et des Réseaux - Pearson Education

[7] http://tvaira.free.fr/bts-sn/reseaux/cours/cours-reseaux-iptables.pdf

[8] https://www.shellscript.sh/

[9] http://www-igm.univ-mlv.fr/~duris/NTREZO/20052006/MasquelierMottierPronzato-Firewall-rapport.pdf

[10] http://www.ii.uib.no/~serge/SergeGaspers_DUTThesis_Firewall_2002.pdf

[11] https://netfilter.org

[12] https://inetdoc.net/guides/iptables-tutorial/traversingoftables.html

[13] http://netfilter.kernelnotes.org/.

[14] http://koor.fr/Java/Index.wp

[15] cours Mr daoui chapitre 7 Netfilter