Table des matières

Introduction **1**

1 Aspects de base de la sécurité réseau **4**

1.1 La problématique de la sécurité réseau 4

1.2 Les attaques 5

1.2.1 Définition 5

1.3 Les services 6

1.3.1 Définition 6

1.4 Les mécanismes 7

1.4.1 Définition 7

1.4.2 Les mécanismes basés sur la cryptographie 7

1.4.3 Les mécanismes non basés sur la cryptographie 8

1.5 Conclusion 8

2 Les firewalls traditionnels **9**

2.1 Le concept de firewall traditionnel 9

2.1.1 Définition 9

2.2 Les objectifs pour un firewall 10

2.3 Les principales fonctionnalités du firewall 10

2.3.1 Le filtrage 11

2.3.2 L'authentification et la gestion des droits 11

2.3.3 Le NAT 12

2.3.4 Le proxy 12

2.4 Les différents types de firewalls 12

2.4.1 Les firewalls bridges 12

2.4.2 Les firewalls matériels 13

2.4.3 Les firewalls logiciels 13

2.5 Les architectures de firewalls 14

2.6 Mécanisme d'identification des hôtes 15
2.7 Les avantages des firewalls traditionnels 15
2.8 Les inconvénients des firewalls traditionnels 16
2.9 Conclusion 17
3 Les firewalls distribués 18
3.1 Le concept de firewall distribué 18
3.2 Mécanisme d'identification des hôtes 19
3.3 Les principales fonctionnalités du firewall distribué 19
3.3.1 Le filtrage distribué 20
3.3.2 Le contrôle d'accès distribué 20
3.4 Les avantages des firewalls distribués 20
3.5 Les inconvénients des firewalls distribués 22
3.6 Les firewalls hybrides 25
3.6.1 Les nœuds distants et IPsec 25
3.6.2 Serveur d'administration et Parc de firewall personnel 25
3.7 Firewalls distribués Vs D'autres types de firewalls 26
3.8 Les attaques sur les firewalls 30
3.8.1 L'usurpation d'adresse 30
3.8.2 Les services inadéquats 31
3.8.3 Le balayage de port 32
3.8.4 Les attaques par déni de service 33
3.8.5 Les attaques internes 34
3.9 Etude de quelques firewalls distribués existants 34
3.9.1 Le prototype utilisant Keynote 35
3.9.1.1 Avantages du prototype utilisant Keynote 36
3.9.1.2 Inconvénients du prototype utilisant Keynote 37
3.9.2 Le prototype utilisant les filtres IP 38
3.9.2.1 Avantages du prototype utilisant les filtres IP 38
3.9.2.2 Inconvénients du prototype utilisant les filtres IP 39
3.9.3 Le firewall intégré 39
3.9.3.1 Les avantages du firewall intégré 41
3.9.3.2 Les inconvénients du firewall intégré 41
3.9.4 Le firewall distribué autonome 42
3.9.4.1 Les avantages du firewall distribué autonome 43

3.9.4.2 Les inconvénients du firewall distribué autonome 43
3.9.5 Présentation de F-Secure Distributed Firewall 44
3.9.5.1 Avantages de F-Secure Distributed Firewall 45
3.9.5.2 Inconvénients de F-Secure Distributed Firewall 45
3.9.6 Présentation de THEGREENBOW Distributed Firewall 45
3.9.6.1 Avantages du firewall TGB 46
3.9.6.2 Inconvénients du firewall TGB 46
3.10 Comparaison des firewalls distribués existants 47
3.11 Conclusion 50
4 Ebauche de Proposition de Solution 51
4.1 Analyse du problème 51
4.2 Les systèmes de détection d'intrusions 53
4.3 Présentation de Snort 54
4.3.1 Avantages de Snort 56
4.3.2 Inconvénients de Snort 57
4.4 Les bases de la solution proposée 58
4.4.1 Le mécanisme de sécurité utilisé 58
4.4.2 L'interaction entre les différentes entités 59
4.4.3 Communication entre les points finaux 61
4.5 Algorithme de l'Agent Analyzer 62
4.5.1 La récupération des données 62
4.5.2 La modification du fichier de stratégies 65
4.6 Simulation et Validation 68
4.6.1 Les paramètres d'entrées 68
4.6.1.1 Agent Analyzer 68
4.6.1.2 Le programme P2P 70
4.6.2 Mise en place d'une plateforme de test 70
4.6.2.1 Architecture de la plateforme 71
4.6.2.2 Quelques attaques mises en œuvre 72
4.6.3 Tests sur les scans de port 72
4.6.3.1 Résultats des scans avant l'exécution du programme Analyzer 73
4.6.3.1.1 Scan par détection des services en écoute sur une machine 73
4.6.3.1.2 Scan de protocoles IP 74
4.6.3.1.3 Scan TCP connect () 74

4.6.3.2 Résultats des scans après l'exécution du programme Analyzer 75
4.6.3.2.1 Scan par détection des services en écoute sur une machine 76
4.6.3.2.2 Scan de protocoles IP 76
4.6.3.2.3 Scan TCP connect () 77
4.7 Conclusion 78
Conclusion Générale 80
Références 83
Annexe A : Code en C de l'agent Analyzer & le script Shell de son exécution 88
Annexe B : Prototype de firewall implémenté sous *iptables* dans chaque hôte 97
Annexe C : Exemple de règle générée par le programme *analyzer* au niveau de l'hôte 98
Annexe D : Code JAVA du programme P2P 99

Table des figures

2.1 : Firewall à double réseau (Dual Homed Firewall) 10
2.2 : Architecture d'hôte à écran 14
2.3 : Architecture de sous réseaux à écran 14
3.1 : Modèle de firewall distribué 19
3.2 : Structure d'un paquet ou datagramme IP 31
3.3 : Demande de connexion par un service non autorisé 32
3.4 : Architecture EFW/ADF 40
4.1 : Architecture générale de Snort 56
4.2 : Mécanisme de sécurité basé sur Snort et le logiciel Netfilter 58
4.3 : Principe de communication de l'IDS Snort et du firewall Netfilter 59
4.4 : Exemple de communication entre les points finaux 61
4.5 : Ordinogramme de l'agent Analyzer 61
4.6 : Exemple de Bloc alerte au sein du fichier « *alert* » 69
4.7 : Architecture de la plateforme de test 71
4.8 : Test Scan par détection des services en écoute sur une machine 1 73
4.9 : Test Scan de protocoles IP 1 74
4.10 : Test Scan TCP connect () 1 75
4.11 : Test Scan par détection des services en écoute sur une machine 2 76
4.12 : Test Scan de protocoles IP 2 76
4.13 : Test Scan TCP connect () 2 77

Liste des tableaux

Tableau 3.1 : Firewalls distribués Vs Autres catégories de firewalls 27
Tableau 3.2 : Comparaison des Firewalls distribués existants..... 48
Tableau 3.3 : Aperçu des problèmes solubles par le firewall distribué..... 50
Tableau 4.1 : Caractéristiques des machines utilisées..... 70

Liste des abréviations

A	
ADF	Automatic Distributed Firewall
ASD	Automatic Software Distribution
AT & T	American Telephony and Telegraphy
D	
DIDS	Distributed Intrusion Detection System
E	
EFW	Embedded Firewall
F	
FAI	Fournisseur d'Accès à Internet
Fws	Firewalls
I	
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IDWG	Intrusion Detection Work Group
IETF	Internet Engineering Task Force
N	
NIC	Network Interface Card
S	
SE	Système d'exploitation
SMS	Microsoft Systems Management Server
V	
VPG	Virtual Private Group
VPN	Virtual Private Network

Introduction Générale

Avec l'extension de l'Internet, la sécurité réseau a suscité une attention particulière dans les communautés industrielles et de recherches. Suite à la menace croissante des attaques réseau, les firewalls sont devenus des éléments importants non seulement pour les réseaux d'organisations (entreprises, universités, etc.), mais également pour les réseaux de petite taille et chez les particuliers.

Considérés comme des boucliers s'interposant entre un réseau sûr et un réseau non sûr (par exemple Internet), les firewalls traditionnels s'appuient principalement sur l'hypothèse selon laquelle la menace vient de l'extérieur et non de l'intérieur. Autrement dit, il faut se protéger des gens qui se trouvent à l'extérieur du réseau local (c'est-à-dire du réseau interne) et faire confiance à ceux qui se trouvent à l'intérieur du réseau local. L'interconnexion des réseaux, a remis cette hypothèse en question.

Le firewall traditionnel constitue un point de vulnérabilité du réseau interne ; dès qu'il tombe c'est tout le réseau interne qui est privé du monde extérieur. Il est également un point de congestion car tout trafic qui vient de l'intérieur ou de l'extérieur passe par lui ; ce qui veut dire que la performance du réseau entier dépend de lui. Aussi, la gestion des extranets constitue un handicap, car ils autorisent des externes à accéder au réseau interne en passant par le firewall ; cela n'est pas sûr dans la mesure où il n'est pas certain que ce soit les gens autorisés qui accèdent au réseau interne depuis l'extranet.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Ainsi, bien que longtemps considérés comme la panacée de la sécurité réseau, les firewalls traditionnels ne constituent plus la solution complète permettant de fournir une sécurité totale aux réseaux. Mais cependant ils restent des dispositifs utiles et importants pour la sécurité des réseaux. Les firewalls sont des mécanismes efficaces pour contrôler et appliquer la stratégie de sécurité du réseau.

Pour résoudre les problèmes soulevés par les firewalls traditionnels tout en conservant leurs avantages, le concept de firewall distribué fut proposé. Dans un tel système, la stratégie de sécurité reste centralement définie comme au niveau des firewalls traditionnels, mais elle est appliquée au niveau de chaque hôte final.

Les firewalls distribués ont résolu de nombreux problèmes soulevés par les firewalls classiques dont le principal semble être la dépendance de la topologie. Cependant comme toute technologie, eux aussi présentent des inconvénients. Une des principales limites des firewalls distribués est la staticité de la stratégie de sécurité des hôtes. En effet, dans la plupart des environnements de firewalls distribués, la stratégie de sécurité distribuée aux hôtes est statique en ce sens qu'ils ne peuvent pas la modifier dynamiquement afin de bloquer les intrusions ou tentatives d'intrusions qui s'opèrent dans leurs environnements. Quelques travaux ont tenté de comblé cette insuffisance pour d'autres types de firewalls distribués. Les firewalls distribués utilisant le filtre *iptables* pour définir la stratégie de sécurité de leurs hôtes souffrent de cette insuffisance.

C'est dans ce cadre que se situent nos travaux, qui ont pour objectif de proposer une solution pour ce type de firewall distribué en utilisant un système de détection d'intrusion open source moyennant un agent, afin de permettre aux hôtes de modifier leurs stratégies de sécurité de façon dynamique pour bloquer les intrusions ou tentatives d'intrusion qui se produisent dans leurs environnements.

Le mémoire est organisé comme suit : le chapitre 1 présente brièvement les aspects de base de la sécurité réseau. Le chapitre 2 donne un aperçu général des firewalls traditionnels ainsi que leurs avantages et inconvénients. Le chapitre 3 présente le concept de firewall distribué dans sa globalité ainsi que ses avantages et inconvénients. Il expose également le problème que nous voulons résoudre dans le cadre de ce travail. Le chapitre 4, qui constitue notre

contribution, expose la solution que nous avons proposée pour le résoudre. Enfin nous concluons et proposons des suites possibles au travail réalisé.

Chapitre 1

Aspects de base de la sécurité réseau

Dans ce chapitre, nous introduisons les aspects de base de la sécurité réseau à savoir les attaques, les services et les mécanismes. Nous montrons également à quel niveau de ces éléments se situe le firewall.

1.1 La problématique de la sécurité réseau

La ***sécurité informatique*** ou sécurité des systèmes est un ensemble de mesures ou d'outils permettant d'assurer la sécurité (ie la prévention, la détection et la réaction ou restauration) des données dans un ordinateur même si celui-ci est connecté à un réseau. Cet aspect n'est pas à confondre avec la ***sécurité réseau*** qui se définit comme étant un ensemble de mesures ou d'outils permettant d'assurer la sécurité lors de la transmission des données entre les hôtes dans un réseau [2].

Cette sécurité réseau n'a pas beaucoup attiré l'attention des gens durant les premières décennies de l'existence des réseaux informatiques, du fait qu'ils aient été principalement utilisés dans les universités, par des chercheurs pour envoyer des e-mails et dans les entreprises, par des employés pour partager des imprimantes. Cependant, de nos jours, des millions de citoyens utilisent les réseaux pour des opérations bancaires, d'achats, etc. [3]. Cela a nécessité, une prise en compte plus sérieuse de la sécurité réseau.

La sécurité réseau utilise les notions d'attaques, de services et de mécanismes.

1.2 Les attaques

1.2.1 Définition

C'est toute action compromettant la sécurité du réseau. Elle peut être dirigée vers une destruction des données, un vol ou la perturbation du bon fonctionnement du réseau (public ou privée). Par exemple, tentative d'accès, tentative de détournement d'informations, etc. Les attaques peuvent être effectuées directement sur les objets, on parle alors de ***protection des ressources*** ou ***de l'information,*** ou bien indirectement à travers les lignes de communication, c'est la ***protection de la communication*** [4]. On distingue deux types d'attaque :

- **Les attaques actives**

  Elles visent à modifier l'état des informations (ou ressources). Les attaques actives entraînent le plus souvent l'insertion, la destruction ou la modification des données. Ce genre d'attaque est plus facile à détecter que les attaques passives. On distingue plusieurs types d'attaques actives parmi lesquelles, nous avons *l'usurpation ou mascarade*, *la fabrication*, *le déni de service* et *le rejeu*.

  Au niveau de l'*usurpation*, on prétend être quelqu'un qu'on est pas en prenant l'identifiant d'un autre. La *fabrication* consiste à modifier quelque chose de plus dans le message. Le *déni de service* consiste à empêcher le fonctionnement normal des moyens de communication. Enfin l'*attaque par rejeu* consiste à capturer les données et à les retransmettre tel quel (sans aucun déchiffrement) au destinataire.

- **Les attaques passives**

  Elles visent à prendre frauduleusement connaissance de la nature des données transmises ou stockées. Ces attaques ne modifient pas les données, ni le fonctionnement du réseau. On en distingue principalement deux types : *l'interception de contenu* et l'*observation du trafic*.

  L'interception de contenu consiste à intercepter une information qui ne nous est pas destinée. Par exemple, on intercepte un message qui ne nous est pas destiné. Tandis que l'*observation du trafic* consiste à observer les paquets dans le but de savoir quel type de trafic (Telnet, ICMP) se passe dans le réseau.

1.3 Les services

1.3.1 Définition

Ce sont les moyens mis en œuvre pour contrer les attaques. Ils permettent ainsi, de renforcer la sécurité dans un réseau. Les services de bases comprennent [7, 9] :

- ***L'authentification*** : C'est le fait de fournir l'assurance que l'entité communicante est bien celle qu'elle prétend être. En d'autres termes, si B envoie à A, A sait que c'est B qui envoie le message. En outre B sait que c'est A qui le reçoit.

- ***Le contrôle d'accès*** : Il consiste à empêcher l'utilisation non autorisée d'une ressource. Ce service contrôle qui doit avoir accès à la ressource, selon quelles conditions et quant il y accède, qu'est ce qu'il peut faire avec la ressource. A ce niveau, on peut utiliser un firewall pour empêcher l'entrée ou la sortie de paquets non autorisés [5].

- ***La confidentialité des données*** : Elle permet de protéger les données contre une lecture, un dévoilement non autorisé. Par exemple, l'information échangée entre deux correspondants ne peut pas être consultée par un tiers. Ce service doit assurer la protection de toutes les données sur une connexion.

- ***L'intégrité des données*** : Elle assure que les données reçues sont exactement celles envoyées par l'entité autorisée, et qu'elles n'ont pas été modifiées, ni supprimées, ni augmentées. Ce service permet d'assurer l'intégrité des données sur une connexion avec une possibilité de récupération ou non.

- ***La non répudiation*** : Elle assure la protection contre un déni par une des entités d'avoir participé à la communication ou à une partie de la communication. Ainsi, elle permet à un hôte de réception d'identifier un autre hôte comme origine d'un message. L'expéditeur ne peut pas nier d'avoir envoyé le message et, est donc responsable de son contenu.

- ***La disponibilité*** : Elle permet de prévenir un système contre les perturbations et les interruptions de son fonctionnement et aussi contre toute utilisation abusive des services et des ressources du système. Le réseau doit à tout moment être disponible pour envoyer et

recevoir des messages même s'il est soumis aux attaques. Les menaces possibles pour la disponibilité sont sous forme de dénis de service.

1.4 Les mécanismes

1.4.1 Définition

Ce sont les moyens utilisés par les services pour contrer les attaques. Ils comprennent des outils basés sur la cryptographie ou non. *Les mécanismes basés sur la cryptographie* utilisent le chiffrement, la signature numérique, l'authentification de l'échange, etc. *Les mécanismes non basés sur la cryptographie* utilisent la détection d'événements, l'audit des traces laissées par les tentatives d'intrusion, les firewalls, une autorité de certification. L'ensemble de ces mécanismes sont utilisés par les services pour empêcher les attaques [6, 7].

1.4.2 Les mécanismes basés sur la cryptographie

Ces derniers se basent sur le chiffrement. La cryptographie se définit comme étant l'art d'étudier les écritures sécrètes. Les mécanismes basés sur la cryptographie utilisent entre autres :

- ***Le chiffrement*** : Il consiste à utiliser des algorithmes mathématiques pour transformer du texte clair (message non crypté) en texte chiffré (message crypté) ;

- ***La signature numérique*** : C'est une information ou un code cryptographique que l'on ajoute au message originel (texte clair), pour permettre au récepteur du message de s'assurer de la source et de l'intégrité du message reçu. La signature numérique permet également au récepteur du message de vérifier que celui-ci n'a pas été modifié lors de son acheminement ;

- ***L'authentification de l'échange*** : Ce mécanisme vise à assurer l'identité d'une entité au moyen d'échanges d'informations.

1.4.3 Les mécanismes non basés sur la cryptographie

Pour mettre en œuvre les services de sécurité ces mécanismes utilisent entre autres :

- ***La détection d'événements*** : Ce mécanisme vise à détecter les événements relatifs à la sécurité. Comme exemple, le fait de détecter une tentative illégale d'ouverture de session ;

- ***L'audit des traces laissées par les tentatives d'intrusion*** : A ce niveau, un ensemble de données sont collectées et utilisées pour mettre en oeuvre un audit de sécurité. Cette dernière est indépendante de la consultation et du contrôle des activités et des logs du système ;

- ***Les firewalls*** : Ce mécanisme vise à utiliser des firewalls (ou pare feux) pour contrer les attaques ;

- ***Une autorité de certification*** : Elle consiste à une tierce partie qui certifie aux entités en communication que les informations qu'elles échangent proviennent réellement d'elles.

1.5 Conclusion

Dans ce chapitre premier nous avons vu les problèmes soulevés par la sécurité réseau à savoir les attaques et nous avons vu les services de base, ainsi que les mécanismes utilisés pour assurer la sécurité réseau. Nous avons vu qu'au niveau du contrôle d'accès on pouvait introduire un firewall pour empêcher l'entrée ou la sortie de paquets non autorisés par l'organisation. Ce concept est traité dans le prochain chapitre.

Chapitre 2

Les firewalls traditionnels

Dans ce chapitre, nous présentons ce qu'est un firewall, les objectifs pour un firewall, ses principales fonctionnalités, les différents types de firewalls, les architectures de firewall les plus utilisées, le mécanisme d'identification des hôtes au niveau des firewalls ; enfin nous présentons les avantages et les inconvénients des firewalls. Nous terminons ce chapitre en introduisant le concept de firewall distribué.

2.1 Le concept de firewall traditionnel

Les firewalls sont des mécanismes non basés sur la cryptographie intervenant au niveau du contrôle d'accès. Ils sont en général utilisés pour quatre raisons : se protéger des malveillances externes, éviter la fuite d'information contrôlée vers l'extérieur, surveiller les flux internes/externes et faciliter l'administration réseau [8].

2.1.1 Définition

Un firewall est un composant ou un ensemble de composants matériels ou logiciels qui restreint l'accès à un réseau protégé à partir d'un réseau non protégé, ou d'une interconnexion de réseaux non sécurisée [2]. Dans sa structure la plus simple, le firewall se présente de la manière suivante :

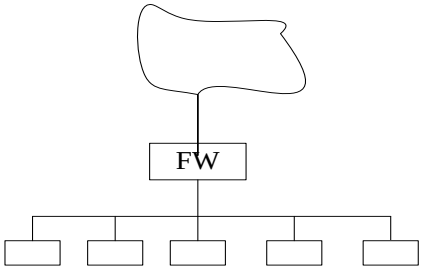

Figure 2.1 : Firewall à double réseau (Dual Homed Firewall)

Commentaire : La fonction principale du firewall de la figure 2.1 est de filtrer les trafics entrants ou sortants. Dans son architecture la plus simple, ce firewall utilise un routeur ou un ordinateur qui a deux cartes réseau. Cependant, notons qu'il s'agit là, d'un firewall très élémentaire c'est-à-dire moins sécurisé.

Ce concept de firewall, également appelé firewall traditionnel s'appuie sur une hypothèse selon laquelle « il faut faire confiance à celui qui se trouve dans le réseau interne et considérer celui qui est à l'extérieur comme un ennemi ou comme un ennemi potentiel » [1].

2.2 Les objectifs pour un firewall

On en distingue trois [9] à savoir :

- Tout le trafic sortant et entrant doit obligatoirement passer par le firewall;
- Seul le trafic défini par l'autorité locale de contrôle sera autorisé à passer ;
- Le firewall lui-même doit être protégé contre les attaques. Ceci implique l'utilisation d'un système sûr avec un système d'exploitation sécurisé.

2.3 Les principales fonctionnalités du firewall

On en distingue principalement quatre à savoir : le filtrage, l'authentification/gestion des droits, le NAT et le proxy.

2.3.1 Le filtrage

On en distingue principalement trois : le filtrage de paquet, le filtrage dynamique et le filtrage applicatif.

- ***Le filtrage de paquet*** : Il s'agit d'un filtrage réalisé au niveau des couches 2 à 4 dans un routeur (une passerelle), un pont ou un hôte. Le principe consiste à appliquer les règles de filtrage des paquets sur la base des champs des entêtes de couche 2 (adresse MAC), de couche IP (adresse IP source et/ou destination) et transport (n° de port UDP ou TCP) [10]. Le filtrage se fait selon des règles, le paquet est accepté ou rejeté [9, 11].

- ***Le filtrage dynamique*** : Ce type de filtrage également appelé « stateful inspection » est basé sur l'inspection des couches 3 et 4 du modèle OSI. Il permet d'effectuer un suivi des transactions entre le client et le serveur, et donc de s'assurer de la bonne circulation des données de la session en cours. Un paquet qui n'appartiendrait pas à une session encore active, sera soumis au filtrage prédéfini (filtrage de paquets). Les services comme ftp qui utilise un port standard (< 1024) pour la connexion, puis des ports dynamiques pendant les transferts, sont soumis à ce type de filtrage. Les ports dynamiques ont une valeur comprise entre 49152 et 65535 [9, 11, 12].

- ***Le filtrage applicatif*** : Il est utilisé au niveau de la couche 7 (application) du modèle OSI. Il doit contrôler les communications au niveau des applications (analyse application par application). L'idée est de reconnaître directement à partir d'un flot de paquets, l'identité de l'application. Ce type de filtrage permet d'identifier une application insérée dans une autre, et de reconnaître les applications sur des ports non conformes. Un firewall utilisant un filtrage applicatif est appelé *passerelle applicative*, *filtre applicatif*, *serveur mandataire* ou *serveur proxy*, car il permet de relayer des informations entre deux réseaux en effectuant un filtrage fin au niveau du contenu des paquets échangés [2, 5, 9, 10, 11, 12].

2.3.2 L'authentification et la gestion des droits

La principale fonction d'un firewall est de centraliser le contrôle d'accès. L'identité de l'utilisateur est reconnue par des mécanismes d'authentification qui sont, dans le pire des cas, un login et un mot de passe. Il est possible de lui affecter les droits et les privilèges qui sont en

général associés à un utilisateur. Les systèmes d'exploitation peuvent permettre un accès à des utilisateurs suivant les permissions accordées [13, 14].

2.3.3 Le NAT

Il s'agit d'un mécanisme de translation d'adresse (Network Address Translation) qui a été mis au point pour répondre à la pénurie d'adresses IP avec le protocole IPv4. Le principe du NAT consiste à utiliser une passerelle (firewall, routeur, etc.) de connexion à Internet, possédant au moins une interface réseau connectée sur le réseau interne et au moins une interface réseau connectée à Internet avec une adresse IP routable, pour connecter l'ensemble des machines du réseau. L'objectif est de modifier, au niveau de la passerelle, les adresses IP dans les entêtes des paquets (datagrammes) provenant du réseau interne vers le réseau externe [15, 16]. Le NAT peut permettre de rendre les machines d'un réseau local inaccessibles directement de l'extérieur si on n'utilise pas ***la redirection de port*** ou « Port Forwarding », qui consiste à configurer la passerelle pour transmettre à une machine spécifique du réseau interne tous les paquets reçus à partir d'un port particulier.

2.3.4 Le proxy

Egalement appelé serveur mandataire, le *proxy* est un dispositif qui a pour fonction de relayer différentes requêtes et d'entretenir un *cache* de réponses. A l'origine, il a été créé pour relayer les navigateurs des réseaux locaux vers Internet. Aujourd'hui, il est doté de nouvelles fonctionnalités : cache, journalisation des requêtes, filtrage, navigation anonyme et sécurité du réseau local. Le proxy peut être de niveau application, il comprend alors et interprète les commandes du protocole d'applications. Le proxy sert d'intermédiaire dans une connexion entre le client et le serveur pour relayer la requête qui est faite. Ainsi, le client s'adresse toujours au proxy et celui-ci s'adresse au serveur [2, 17].

2.4 Les différents types de firewalls

On en distingue 3 types : les firewalls bridges, les firewalls matériels et les firewalls logiciels.

2.4.1 Les firewalls bridges

Ils sont assez répandus et ne disposent pas d'adresses IP sur leurs interfaces. Les firewalls *Bridges* ne font que transférer les paquets d'une interface à une autre en leurs appliquant les

règles prédéfinies. L'absence d'adresse IP permet à ce type de firewall d'être indétectable sur le réseau. Le plus souvent, ils ne disposent que des fonctionnalités de base (filtrage sur adresse IP, port). Donc ils agissent comme des firewalls de bas niveau. Ce type de firewall se trouve typiquement sur les switchs et coûte généralement moins cher [17, 18].

2.4.2 Les firewalls matériels

Ils se trouvent généralement sur les routeurs et sont intégrés au matériel réseau, ce qui réduit les coûts de maintenance. Ces firewalls offrent un bon niveau de sécurité. Ils sont peu flexibles en ce sens qu'ils dépendent du constructeur pour les mises à jours. De plus, ils coûtent généralement cher [17, 18].

2.4.3 Les firewalls logiciels

Présents à la fois dans les serveurs et les machines, on peut les classer en deux grandes catégories : les firewalls plus « fiables » et les firewalls personnels.

Les firewalls plus « fiables » tournent généralement sous Linux, car ce système d'exploitation offre une sécurité réseau plus élevée et un contrôle plus précis. Cette catégorie de firewall se comporte souvent, comme les firewalls matériels des routeurs à la seule différence qu'ils sont configurables à la main. Le plus courant est *iptables* (anciennement *ipchains*) [18].

Les firewalls personnels sont apparus en 1999 avec les technologies permettant aux utilisateurs d'obtenir de grands débits (ADSL, modem câble). Le firewall personnel est un logiciel qui fonctionne directement sur l'ordinateur à protéger. Il permet le passage sélectif des flux de données entre un ordinateur personnel et un réseau public, tel que Internet et la neutralisation des tentatives d'intrusion en provenance du réseau public. Il offre également la possibilité de contrôler quels logiciels vont se connecter sur Internet. Ces firewalls sont assez souvent commerciaux et quelque fois très peu sécurisés, car dès qu'un hacker parvient à s'introduire dans l'ordinateur, il pourra facilement désactiver le firewall. Ils s'orientent plus vers la simplicité d'utilisation que vers l'exhaustivité, afin de rester accessible à l'utilisateur final [18, 19].

2.5 Les architectures de firewalls

Les deux architectures les plus utilisées sont : l'architecture d'hôte à écran et celle de sous réseaux à écran.

L'architecture d'hôte à écran utilise un hôte appelé *hôte bastion* à partir duquel tous les hôtes de l'extérieur vont se connecter. Le bastion (hôte, serveur, etc.) est le point fort (ou critique) de la sécurité du réseau et représente le seul élément accessible de l'extérieur. Si une passerelle de filtrage de paquets doit être déployée, le bastion doit être configuré pour que toutes les connexions provenant de l'extérieur passent par lui afin d'éviter une connexion directe entre le réseau local et Internet [2, 13, 14].

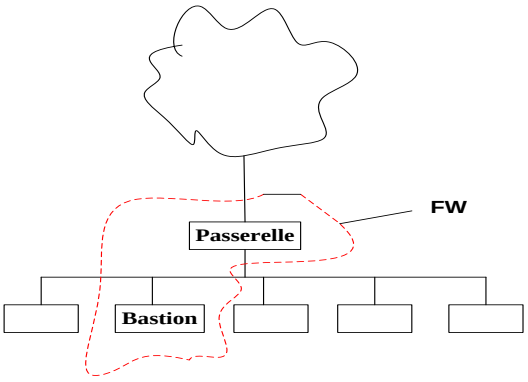

Figure 2.2 : Architecture d'hôte à écran

L'architecture de sous réseaux à écrans est similaire à celle d'hôte à écran, mais ajoute un niveau de sécurité plus solide, en créant un réseau qui contiendra l'hôte bastion. Ce réseau sépare le réseau local du réseau externe. Cela permet de garantir, au cas où l'hôte bastion est corrompu, que l'attaque soit restreinte au réseau contenant le bastion [2, 13].

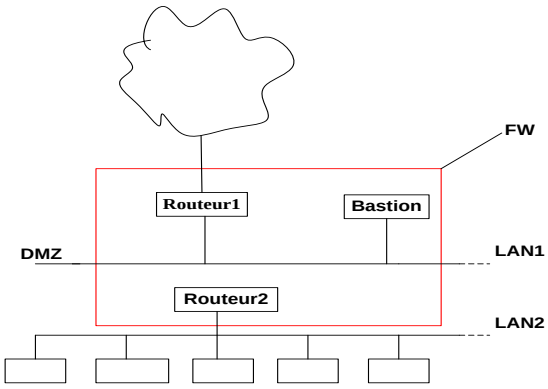

Figure 2.3 : Architecture de sous réseaux à écran

La DMZ ou zone démilitarisée permet l'installation de machines jugées à risque (serveur Web, serveur POP, Bastion) dont la chute ne mettra pas en cause le réseau interne.

2.6 Mécanisme d'identification des hôtes

Les firewalls conventionnels s'appuient sur la topologie du réseau. Les hôtes sont identifiés par leurs adresses IP et les interfaces réseaux des firewalls ; qui sont désignées comme « intérieure », « extérieure », « DMZ », etc. Notons, toutefois, que ce type de mécanisme est faible. En effet, tout celui qui dispose d'un accès au réseau interne et qui obtient une adresse IP interne est considéré comme fiable. Du fait que tous les hôtes du réseau interne jouissent du même niveau de confiance, si une de ces machines est corrompue elle peut être utilisée pour lancer des attaques vers les autres hôtes par usurpation d'identité via le protocole *rlogin* [20].

2.7 Les avantages des firewalls traditionnels

Il y a plusieurs raisons d'utiliser un firewall traditionnel. La principale raison semble être le fait qu'il soit un moyen facile pour imposer une stratégie de sécurité. En le plaçant comme passerelle au niveau du réseau local, les hôtes internes sont protégés des attaques malveillantes provenant de l'extérieur. Ces firewalls ont plusieurs avantages [1, 10, 11, 21, 22] :

- Ils sont un mécanisme de prise de décisions et de contrôle de la stratégie de sécurité. Les firewalls traditionnels permettent à l'administrateur réseau de définir des stratégies pour les accès externes à partir du réseau local et des accès internes à partir de l'extérieur ;

- Ils peuvent être utilisés pour protéger les applications provenant d'anciens systèmes de fichiers. Un exemple de ce type d'applications est le système de fichier réseau (NFS), qui autorise un ordinateur à accéder à des fichiers via le réseau, car les versions antérieures de ce type de système ont été conçues sans penser à la sécurité ;

- Ils offrent des logs d'événements qui permettent d'auditer le trafic du réseau. Ces logs peuvent fournir des traces tangibles en cas d'incident, si le firewall n'est pas lui-même une des cibles ;

- Ils peuvent facilement contrer les attaques qui s'opèrent au niveau des couches inférieures en utilisant les fonctionnalités de filtrage décrites précédemment (voir section 2.3.1).

2.8 Les inconvénients des firewalls traditionnels

Le modèle de firewall traditionnel fonctionne bien pour des réseaux de petites et moyennes tailles. Plusieurs tendances dans la gestion des réseaux avec l'extension de la connectivité réseau telles que les extranets, les liaisons haut débit, le télétravail cependant menacent de le rendre obsolète [1, 20, 22] :

- Les firewalls ne protègent pas les réseaux des attaques internes, car ce modèle stipule de faire confiance aux internes. Une façon de contourner ces menaces est de diviser le réseau en plusieurs petits réseaux dans lesquels on va déployer des firewalls. Ces réseaux seront protégés les uns des autres et de tout réseau extérieur. Du fait que des stratégies différentes sont appliquées à ces firewalls, la charge et la complexité de l'administration augmentent ;

- Les firewalls peuvent sans ambiguïté contrôler le trafic qui passe par eux ; cependant ils ne peuvent rien faire contre le trafic qui ne passe pas par eux. Par exemple, si le site autorise des accès aux systèmes internes par Wireless derrière le firewall, ce dernier ne peut en aucun cas protéger le site en cas d'attaques ;

- Les firewalls sont conçus pour protéger les sites contre les menaces connues. Des firewalls bien conçus peuvent protéger contre certaines nouvelles menaces. Cependant, aucun firewall ne peut protéger contre toutes les attaques qui surviennent. Il n'est pas possible de configurer un firewall une seule fois et de s'attendre à ce qu'il protège le réseau pour toujours ;

- Le chiffrement de bout en bout constitue une menace puisque le firewall n'a aucune visibilité sur les champs qu'il doit filtrer au niveau des paquets. Ceci est dû au fait qu'il n'a pas les clés nécessaires pour déchiffrer les paquets. En autorisant le chiffrement de bout en bout à travers le firewall, les administrateurs réseau font preuve de confiance considérable aux utilisateurs ;

- Le firewall est l'unique point d'entrée dans le réseau à partir duquel on impose une certaine stratégie pour filtrer le trafic. Il est donc un point de vulnérabilité. S'il tombe pour n'importe quelle raison, tous les réseaux internes sont isolés du monde extérieur. Cependant, il existe une alternative qui consiste à configurer des firewalls de secours en cas d'incident ; mais ce genre de solutions peut être coûteux ;

- Les firewalls tendent à devenir des points de congestion, à cause de l'augmentation du débit des réseaux, de la quantité de données passant à travers et de la complexité des protocoles (tel que IPsec) que les firewalls doivent supporter.

2.9 Conclusion

Dans ce chapitre, nous avons abordé la notion de firewall en général. Ce concept présente des avantages tout comme des inconvénients. Ainsi, dans le but de résoudre les problèmes soulevés par les firewalls traditionnels, tout en conservant leurs avantages, il a été proposé le concept de ***firewall distribué***. Ce concept fait l'objet du chapitre suivant.

Chapitre 3

Les firewalls distribués

Dans ce chapitre, nous définissons la notion de firewall distribué. Nous présentons ensuite le mécanisme d'identification des hôtes dans ce modèle, ses principales fonctionnalités, ses avantages et inconvénients. Egalement, nous évoquons le concept de firewall hybride, puis nous comparons le modèle de firewall distribué avec les différents types de firewall. Nous évoquons aussi les attaques qui peuvent survenir lorsque l'on implémente un firewall distribué et enfin nous présentons les firewalls distribués existants ainsi qu'un tableau récapitulatif qui les compare. Nous terminons ce chapitre par un tableau montrant les réponses des firewalls distribués aux problèmes étudiés.

3.1 Le concept de firewall distribué

Ce concept a été proposé pour palier les insuffisances des firewalls traditionnels tout en conservant leurs avantages. Dans ce modèle, chaque *point final* du réseau constitue un firewall, le point final pouvant être un hôte protégeant un segment du réseau. Le concept de firewall distribué implique une définition et une gestion centralisée des stratégies de sécurité couplées avec une application décentralisée de ces stratégies au niveau de chaque point final.

Un firewall distribué est donc une multitude de firewalls implantés dans des hôtes qui sont configurés et gérés de façon centrale [23]. Dans le cas d'un petit réseau, le firewall distribué peut résider sur chaque hôte et être contrôlé par un système d'administration central. Le fait qu'il réside sur chaque point final rend ce modèle indépendant de la topologie [24]. Dans l'architecture du firewall distribué, la stratégie de sécurité reste centralement définie comme dans le cas des firewalls traditionnels, cependant, elle est appliquée au niveau de chaque point

final (hôtes, routeurs, etc.). La stratégie centralisée, définit quelle sorte de connexion est autorisée ou refusée. Ensuite cette stratégie est distribuée à tous les points finaux où elle est imposée [20, 1].

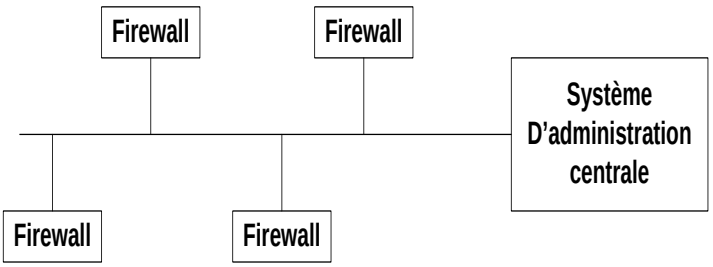

Figure 3.1 : Modèle de firewall distribué

En se référant à l'article de Bellovin [1], les firewalls distribués s'appuient sur trois notions :

- Un *langage de stratégie* qui permet de définir quelles connexions, autoriser ou refuser ;
- Des *outils d'administration système* (tel que Microsoft SMS, ASD, etc.) pour distribuer la stratégie aux machines protégées ;
- IPsec, qui est un *mécanisme de chiffrement* au niveau réseau pour TCP/IP, qui identifie chaque hôte comme élément du réseau sécurisé.

3.2 Mécanisme d'identification des hôtes

Tout comme les firewalls classiques, les firewalls distribués peuvent utiliser des adresses IP pour identifier les hôtes. Mais la fiabilité de ce mécanisme reste à désirer. Il est préférable d'utiliser des *certificats IPsec* pour identifier les machines. Un certificat IPsec est unique pour chaque machine et n'est pas facilement falsifiable à la différence d'une adresse IP. Le certificat est indépendant de la topologie en ce sens qu'il s'applique indépendamment de l'endroit où se trouve la machine. La stratégie est distribuée selon ces certificats. Ils permettent d'identifier aussi bien les hôtes qui participent à la gestion du firewall distribué que les hôtes distants [1].

3.3 Les principales fonctionnalités du firewall distribué

Tout comme le firewall traditionnel, le firewall distribué fait principalement du filtrage et du contrôle d'accès, sauf que, cette fois ci, il s'agit d'un filtrage et d'un contrôle d'accès distribué.

3.3.1 Le filtrage distribué

Il est basé sur la distribution du mécanisme de filtrage sur les machines du réseau à protéger. Ce filtrage distribué propose l'application du mécanisme de filtrage au niveau de chaque machine hôte du réseau et non pas sur une seule machine (comme c'est le cas avec les firewalls traditionnels) pour protéger le réseau. Ceci permet d'appliquer des stratégies de sécurité propres à chaque machine, pour tout trafic entrant ou sortant. Le filtrage distribué peut aussi dépasser le niveau des machines, pour s'appliquer au niveau des utilisateurs réseau, et leur affecter ainsi, des stratégies de sécurité qui s'appliquent indépendamment de la machine à partir de laquelle ils ont accès au réseau [1, 25].

3.3.2 Le contrôle d'accès distribué

Il consiste à utiliser un ensemble de mécanismes placés sur des équipements distincts afin de fournir le mécanisme de contrôle d'accès. Le firewall distribué a pour objectif de rendre un service de contrôle d'accès comparable à celui pouvant être réalisé par un firewall traditionnel, mais en plaçant les mécanismes de contrôle d'accès sur des équipements d'extrémité tels que des stations de travail, ou sur des systèmes intermédiaires tels que des routeurs ou des ponts. Dans chaque cas, il s'agit de profiter des mécanismes de contrôle d'accès implémentés dans les équipements existants et de configurer ceux-ci afin qu'ils fournissent un contrôle d'accès cohérent [1, 26].

3.4 Les avantages des firewalls distribués

Les firewalls distribués ont émergé dans le but de combler les insuffisances des firewalls traditionnels. Contrairement aux firewalls classiques, les firewalls distribués fonctionnent bien pour des réseaux de petite, moyenne et grande taille. Cette technologie offre plusieurs avantages [1, 20, 21, 22, 24] :

- **Indépendance de la topologie :**
  Un des avantages les plus importants des firewalls distribués, est qu'ils peuvent protéger les hôtes qui ne sont pas à la frontière topologique. Ceci permet une flexibilité considérable dans la définition du périmètre de sécurité, qui peut facilement être élargi pour contenir sans problème des hôtes et des réseaux distants (tels que les télétravailleurs, les extranets) ;

- **Protection contre les attaques internes :**

  Les internes ne sont plus considérés comme des gens fiables. Un interne qui, à partir d'un hôte interne, essaye de communiquer avec un hôte de confiance en utilisant le protocole rlogin, peut facilement être détecté par le firewall distribué, qui saura que la demande vient d'une machine interne et donc la rejettera ;

- **Les hôtes prennent les meilleures décisions :**

  Les firewalls traditionnels ne sont pas capables de détecter des paquets ACK usurpés. Ils les laissent passer en pensant qu'il s'agit de réponses destinées aux hôtes internes qui entament un dialogue. Ce qui n'est pas toujours vrai. Avec le firewall distribué, une machine hôte qui entame un dialogue ou qui envoie des requêtes sait exactement quels paquets elle attend, étant donné qu'elle a suffisamment de connaissance pour déterminer la légitimité des paquets attendus. En plus, si un message arrive sous forme de plusieurs paquets fragmentés, le firewall distribué rassemblera tous les paquets avant de les contrôler ;

- **Disponibilité de stratégies pour des applications spécifiques :**

  Les stratégies pour des applications spécifiques peuvent être disponible au niveau des hôtes finaux en utilisant le même canal de distribution (IPsec, un serveur web, un autre protocole, Microsoft SMS, ASD, etc.) ;

- **Elimination de l'unique point de congestion et de vulnérabilité :**

  Avec un firewall traditionnel, nous avons un seul point de vulnérabilité (le firewall lui même) car s'il tombe, c'est tout le réseau qui coulera avec lui. Le firewall distribué nous offre plusieurs points de vulnérabilité, ce qui fait que, si un hôte est corrompu, le réseau entier ne sera pas affecté et continuera à fonctionner. Le firewall classique constitue le seul point de congestion, car tout le trafic est traité à son niveau. Tandis que le firewall distribué, délègue ce traitement à tous les hôtes. Ce qui fait que la performance, la fiabilité et la disponibilité ne dépendent plus d'une seule machine, mais de plusieurs machines.

- **Protection des utilisateurs mobiles :**

  Les utilisateurs mobiles peuvent se déplacer d'un réseau à un autre et chercher à accéder aux ressources du réseau interne. Si un nœud mobile est corrompu, cela peut constituer une sérieuse menace pour le réseau de l'entreprise. Le firewall distribué peut toujours

protéger les utilisateurs mobiles en distribuant de façon sécurisée, une nouvelle stratégie de sécurité au nœud mobile ;

- **Attribution de droits spécifiques aux machines conformes :**
  Le firewall distribué attribue des droits spécifiques aux machines possédant les informations d'authentification spécifiées par la stratégie centrale. Par exemple, un ordinateur portable directement connecté à Internet a le même niveau de protection qu'un PC se trouvant dans le réseau de l'entreprise, car toutes ces machines ont été prévues par l'administrateur pour participer à la gestion du firewall distribué. Par contre, un ordinateur portable connecté au réseau d'entreprise par un visiteur n'a pas les informations d'authentification nécessaires et donc ne disposera pas d'accès au réseau de l'entreprise, bien qu'il soit à l'intérieur de la frontière topologique.

3.5 Les inconvénients des firewalls distribués

Les firewalls distribués ne sont pas venus pour remplacer les firewalls traditionnels dans toutes les architectures. En effet, ils les complètent dans certains cas, en travaillant en conjonction avec eux [27]. Ceci veut dire, que les firewalls distribués sont des moyens de protection très fiables. Cependant, comme toute technologie, ces derniers ont aussi des limites. Ils ont soulevé plusieurs problèmes parmi lesquels nous citons les principaux :

- **Incapacité de protéger les applications provenant des anciens systèmes de fichiers :**
  Un problème avec les firewalls distribués, est qu'ils ne peuvent pas protéger les applications en provenance d'anciens systèmes de fichiers (cas des versions antérieures du système NFS). Ceci est du au fait que, les vieux protocoles utilisés dans ces systèmes n'incluent pas la puissante cryptographie utilisée dans un firewall distribué. Un firewall classique, ou une combinaison de firewall distribué et de firewall classique, protège de tels systèmes des attaques externes [1, 20, 21] ;

- **Coût de déploiement de nouvelles machines :**
  La philosophie du firewall distribué stipule la protection de tous les équipements du réseau y compris les imprimantes et les routeurs. Pour cela, la technologie du firewall distribué nécessite au moins un système pour l'administration et la gestion de la stratégie sans compter les systèmes de secours en cas de panne du système principal. Ce qui peut coûter le déploiement de nouvelles machines de plus dans l'entreprise [24] ;

- **Conformité de la stratégie de sécurité :**

  Le plus grand problème au niveau des firewalls distribués semble être la conformité de la stratégie de sécurité des internes. Ce problème ne se produit pas dans un système où les utilisateurs normaux n'ont pas d'accès root, tel que la plupart des universités parce que normalement, les administrateurs de sécurité dans les organisations (entreprises, universités, etc.) sont aussi les administrateurs locaux pour chaque hôte individuel. Dans ce cas, les stratégies de sécurité avec accès root peuvent être envoyées directement au niveau des hôtes finaux. Les utilisateurs finaux doivent se conformer aux stratégies parce qu'ils ne peuvent modifier les programmes au niveau du noyau et les stratégies de sécurité.

  Ce genre de conformité est très difficile à réaliser dans un système où les utilisateurs finaux ont un contrôle total sur leurs hôtes (accès root). On trouve de tels systèmes dans une entreprise de développement de logiciel. Devant de telles situations, les utilisateurs peuvent modifier ou désactiver les stratégies de sécurité pour telle ou telle raison. Il est possible d'utiliser des fichiers de stratégies signées (avec signature numérique) pour rendre la violation plus dure, mais il est impossible de s'en protéger totalement. En effet un interne déterminé peut également changer l'implémentation de certains programmes selon ses besoins au niveau du noyau.

  Il est clair que ce type de problème ne peut être complètement évité dans un environnement à firewall distribué ou même traditionnel [28] ;

- **Détection d'intrusion :**

  La mise en place d'un système de détection d'intrusion dans un firewall distribué semble être plus problématique qu'au niveau du firewall traditionnel [1, 28, 29].

  Dans un firewall distribué, une connexion suspecte peut être logguée sur l'hôte local, mais il est plus raisonnable que les logs soient collectés et analysés par des experts en sécurité au niveau des serveurs centraux.

  La collection des logs signifie un flot de données constant et une disponibilité des connexions entre chaque hôte final et le serveur central. En particulier, si le système de

détection d'intrusion est requis en temps réel, cela peut ralentir l'ensemble du réseau et le rendre plus vulnérable aux attaques de type déni de service ;

- **Contrôle des failles de la topologie :**

  A cause d'une absence de contrôle d'accès centralisé, le firewall distribué ne peut pas restreindre l'accès de l'intérieur vers l'extérieur si un hôte n'a pas de mécanisme de contrôle de paquets en local. Ce qui veut dire qu'un poste de travail nouvellement installé peut se connecter à l'extérieur sans restriction. La même chose s'applique aux périphériques réseau qui peuvent physiquement être installés sur le réseau [28] ;

- **Authentification basée sur les adresses IP :**

  Un inconvénient avec les firewalls distribués est que les hôtes ne peuvent pas poser des hypothèses sur les adresses IP usurpées sur le réseau local et donc ne peuvent savoir à quels hôtes se fier. Bellovin [1] suggère d'utiliser IPsec pour vérifier l'identité cryptographique des hôtes. Dans tous les cas, ceci est une forme d'authentification plus forte que celle qui consiste à s'appuyer sur des adresses IP, et cela fonctionne même si les adresses IP changent. Cependant le support d'IPsec n'étant pas effectif pour tous les systèmes cette solution n'est pas proche d'être utilisé à travers des domaines plus vastes que les réseaux d'entreprises. Aussi les hôtes devront s'appuyer sur l'authentification basée sur des adresses IP pour communiquer avec les hôtes extérieurs à leurs domaines. Ainsi, beaucoup de vieux systèmes pour lesquels le support d'IPsec n'est pas encore disponible vont rester en utilisation ;

- **Adaptabilité de la stratégie de sécurité des hôtes :**

  Dans le cadre des firewalls distribués actuels, les hôtes appliquent la stratégie de sécurité qui leur est déléguée à partir du serveur central. Cette dernière est statique, en ce sens que l'hôte qui la reçoit ne peut pas la modifier, pour réagir et s'adapter aux changements (attaques, intrusions) qui peuvent s'opérer dans son propre environnement. R. Chinchani et Derek Atkins ont proposé une solution pour résoudre ce problème [30]. Mais celle-ci est relative à un firewall distribué bien défini. En effet, Chinchani et son collègue ont ajouté des modules supplémentaires à l'architecture du firewall distribué utilisant Keynote [22] (voir section 3.9.1) pour résoudre le problème de staticité de la stratégie de

sécurité des hôtes. D'autres firewalls distribués soufrent encore de ce problème. Nous en présentons quelques un dans la suite de ce travail (voir section 3.9).

3.6 Les firewalls hybrides

Bien qu'une implémentation de firewall distribué soit plus sûre et plus flexible, des implémentations hybrides peuvent exister. C'est-à-dire, que l'on peut combiner des techniques de firewalls distribués avec des firewalls traditionnels, pour avoir une fonctionnalité adéquate à moindre coût. Il y a plusieurs techniques d'implémentations des firewalls hybrides, dans cette section, nous en présentons deux [1, 24, 31] : Un première technique, qui utilise *IPsec et des nœuds distants* et une seconde, qui utilise des *firewalls personnels*.

3.6.1 Les nœuds distants et IPsec

Dans cette implémentation, certains hôtes sont derrière un firewall traditionnel, pendant que d'autres résident à l'extérieur. Les machines externes (celles des télétravailleurs) se connectent à Internet en utilisant une passerelle IPsec, se trouvant au niveau du site central. L'administrateur du firewall distribue la stratégie de sécurité aux nœuds distants à partir du site central via le tunnel IPsec. Tout comme dans les réseaux privés virtuels (VPNs), les hôtes distants disposent d'un accès complet au réseau interne, par l'intermédiaire du tunnel IPsec. Le trafic provenant des machines internes vers les nœuds distants est protégé de façon similaire. Ce qui change, c'est que le trafic provenant des nœuds distants vers Internet est gouverné par la stratégie de sécurité du site central. Pour assurer une cohérence avec le trafic provenant des hôtes internes vers Internet, on utilise la même stratégie de sécurité pour contrôler le firewall traditionnel. Ce type d'implémentation est commun aux entreprises ayant un site central principal et un certain nombre de télétravailleurs.

3.6.2 Serveur d'administration et Parc de firewall personnel

L'utilisation croissante des modems câble, ADSL et d'autres technologies qui autorisent des abonnés privés et des petites entreprises à être constamment connectés à Internet a nécessité le besoin de déployer une technologie de firewall uniquement sur des stations de travails. Bien que ce concept manque la notion de protéger un réseau d'hôtes, cependant on peut mettre en place un serveur d'administration, qui permet la gestion du parc de firewalls personnels installés, la configuration des règles de sécurité de ces firewalls, indépendamment ou par

groupe, la centralisation et l'exploitation des logs de chaque firewall. Cette forme d'implémentation est souvent perçue comme une forme particulière de firewall distribué. Cependant, les implémentations actuelles de ce genre de dispositif ne prennent pas en charge les communications chiffrées, mais peuvent offrir des possibilités de configuration plus complexes et imposer une certaine rigueur sur les mécanismes d'authentification.

3.7 Firewalls distribués Vs D'autres types de firewalls

Dans cette section, nous présentons un tableau récapitulatif de comparaison des firewalls distribués avec les différents types de firewalls.

Tableau 3.1 : Firewalls distribués Vs Autres catégories de firewalls

	Fws Distribués	***Fws Classiques***	***Fws Hybrides***	***Fws Matériels***	***Fws Personnels***	***Fws Bridges***
Avantages	- Stratégie de sécurité définie de manière centralisée	- Stratégie de sécurité définie de manière centralisée	- Stratégie de sécurité définie de manière centralisée	- Stratégie de sécurité définie de manière centralisée	- Stratégie de sécurité définie de manière centralisée ou par groupe	- Stratégie de sécurité définie de manière centralisée
	- Ne font confiance à Personne	- Font confiance aux Internes, mais ne font pas confiance aux Externes				
		- Encombrant		-Moins encombrant	-Moins encombrant	-Moins encombrant
	- Peuvent utiliser des adresses IP ou des certificats pour identifier les hôtes	- Utilisent uniquement des adresses IP pour identifier les hôtes	- Peuvent utiliser des adresses IP ou des certificats pour identifier les hôtes			
		- Protègent-les applications provenant des anciens systèmes de fichiers				
	- Peuvent contenir des hôtes distants sans aucun risque	- Peuvent contenir des hôtes distants, mais avec beaucoup de risque	- Peuvent contenir des hôtes distants sans grand risque			

Inconvénients	- Ne protègent pas les applications provenant des anciens systèmes de fichiers - Ne protègent pas contre toutes les attaques de déni de services (smurf par exemple) - Peuvent également être corrompus par un interne	- Ne protègent pas contre toutes les attaques de dénie de services (smurf par exemple) - Sont facilement corrompus par un Interne - Conflit avec IPsec - Font confiance aux internes - Sont l'unique point de vulnérabilité et de congestion du réseau	- Ne protègent pas contre toutes les attaques de déni de services (smurf par exemple) - Peuvent également être corrompus par un interne	-Coûtent généralement cher - Sont vulnérables aux attaques de déni de service	-Coûtent généralement cher - Sont vulnérables aux attaques de déni de service	-Coûtent généralement moins cher - Sont vulnérables aux attaques de déni de service -Filtrage de bas niveau

A partir de ce tableau, nous voyons que les différents systèmes de firewalls présentent des similitudes et des différences. Comme similitudes principales, les stratégies de sécurité des firewalls sont définies et contrôlées de façon centrale.

Du point de vue fonctionnalité, de tous les systèmes de firewalls étudiés, les firewalls bridges semblent être les plus basiques. Ce sont des dispositifs qui ne font que transférer des paquets d'une interface à une autre en leur appliquant les règles prédéfinies (filtrage sur adresse IP, port, etc.).

En général, les systèmes de firewalls traditionnels et matériels disposent de mécanismes de sécurité assez fiables permettant de bien gérer les hôtes internes. Ils permettent de surveiller le trafic réseau grâce à des statistiques (log files), de restreindre l'accès à certains services extérieurs aux utilisateurs internes, etc. Ils protègent la plupart des applications des connexions hostiles. Ces systèmes s'appuient le plus souvent sur la topologie du réseau pour fonctionner. Cependant leur aptitude à gérer les hôtes externes reste à désirer.

Toutefois, les firewalls hybrides peuvent constituer une solution à cet inconvénient du fait qu'ils combinent les techniques de firewalls traditionnels et distribués. Contrairement aux firewalls traditionnels et matériels, les firewalls hybrides disposent de mécanismes efficaces permettant de bien gérer les hôtes internes et externes. Cependant, ils restent moins fiables qu'un firewall distribué du fait que ce modèle s'appuie le plus souvent sur des adresses IP pour identifier les hôtes.

D'un côté, les firewalls distribués offrent des mécanismes de sécurité puissants. A la différence des systèmes de firewalls évoqués précédemment, les firewalls distribués soutiennent que la menace vient de l'intérieur et que l'intrus ne doit pas avoir d'accès aux hôtes du réseau. En outre, même si un hôte du réseau (point final) tombe en panne, les autres vont fonctionner et donc le réseau entier ne sera pas affecté. Les firewalls distribués protègent les hôtes où qu'ils se trouvent et tout le temps. Ils supportent beaucoup de détails de contrôle, tels que le contrôle de niveau application. Les firewalls distribués s'appuient sur des certificats IPsec pour identifier les hôtes. Ces certificats sont difficilement falsifiables par rapport aux adresses IP.

Avec l'apparition de la connexion haut débit et du modem câble, on assiste à une nouvelle forme de firewalls distribués : les firewalls personnels. Ces derniers sont des applications logicielles utilisées pour protéger des hôtes individuellement. Les firewalls personnels utilisent toujours des connexions avec des adresses IP statiques, qui rendent le système vulnérable aux potentiels hackers. En principe, ils gèrent les mêmes tâches que les firewalls classiques. Cependant ils ont d'autres avantages : ils peuvent prendre en charge les besoins de sécurité individuels, etc.

De tous les systèmes de firewalls étudiés, les firewalls distribués semblent être les plus sûrs en matière de sécurité réseau. Ils sont invisibles et n'ont pas besoin de l'intervention de l'utilisateur. Avec un firewall distribué, l'identité de chaque hôte est cryptographiquement assurée, éliminant certaines formes d'attaques de type usurpation, dont les autres firewalls ne peuvent se défendre efficacement.

3.8 Les attaques sur les firewalls

Les firewalls distribués ont à la fois des forces et des faiblesses comparés aux systèmes de firewalls traditionnels. La plus grande différence se trouve au niveau de la topologie : les premiers sont indépendants de la topologie, tandis que les seconds en sont dépendants. Dans cette section, nous étudions quelques attaques communes à ces modèles, ce qui permet de voir dans quel cas utiliser l'un ou l'autre. Parmi ces attaques, nous avons : l'*usurpation d'adresse*, les *services inadéquats/balayage de port*, les *attaques de déni de services* et les *attaques internes* [1, 31].

3.8.1 L'usurpation d'adresse

Dans une attaque de type usurpation d'adresse, l'attaquant remplace l'adresse IP de l'expéditeur d'un paquet IP par l'adresse de sa machine. Ce qui lui permet d'envoyer anonymement des paquets. Toutefois, notons qu'il ne s'agit pas pour autant d'un changement d'adresse IP, mais d'une mascarade de l'adresse IP au niveau des paquets émis, c'est-à-dire une modification des paquets envoyés afin de faire croire au destinataire qu'ils proviennent d'une autre machine(voir Figure 3.2) [32].

Version	Longueur d'en-tête	Type de service	Longueur totale	
Identification			Drapeau	Décalage fragment
Durée de vie		Protocole	Somme de contrôle en-tête	
Adresse IP source				
Adresse IP destination				
Données				

Figure 3.2 : Structure d'un paquet ou datagramme IP

Commentaire : La figure ci-dessus nous montre la structure d'un paquet IP. Masquer l'adresse IP avec la technique du *spoofing* revient à modifier le champ source afin de simuler un datagramme provenant d'une autre adresse IP (donc d'une autre machine).
Le détail relatif à la description des éléments du datagramme n'est cependant pas pertinent pour la suite de notre étude. Il peut être trouvé dans la RFC 791 relative au protocole Internet (ou IP) [33].

Par ailleurs, comme nous l'avons vu précédemment (voir section 2.6), s'appuyer sur des adresses IP n'est pas sûr. L'utilisation des mécanismes cryptographiques empêchera très probablement que les attaques basées sur adresses sources falsifiées se produisent, à condition que le répertoire contenant toutes les informations d'authentification nécessaires n'ait pas été sujet à des attaques. Ces problèmes peuvent être résolus par les firewalls conventionnels en utilisant des règles pour rejeter les paquets au niveau du périmètre réseau, mais cela n'empêchera pas que de telles attaques originaires de l'intérieur du réseau puissent se produire.

3.8.2 Les services inadéquats

Ce type d'attaque vise à s'infiltrer dans le réseau en utilisant les services non autorisés par l'organisation. Par exemple, l'attaquant peut essayer d'accéder à une machine distante en utilisant le protocole *telnet*. Ce dernier figure généralement sur la liste des services non autorisés par une organisation car présentant des défauts de sécurité (voir Figure 3.3) [34].

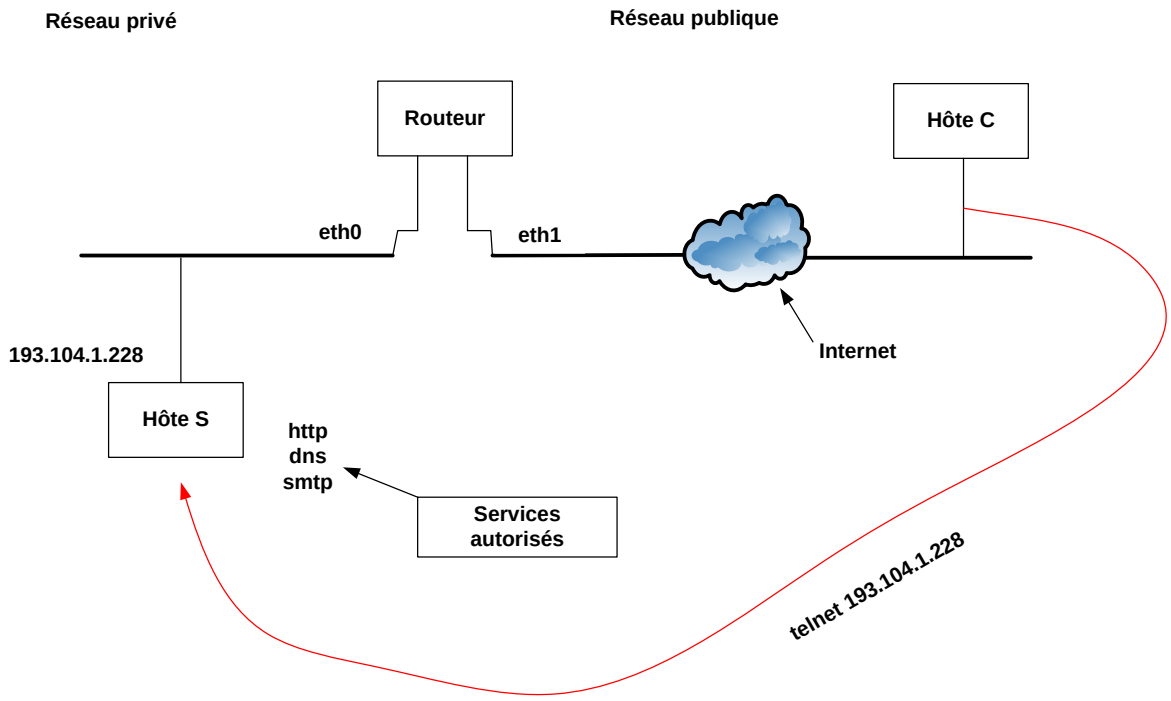

Figure 3.3 : Demande de connexion par un service non autorisé

La figure 3.3 nous montre l'exemple d'une attaque utilisant les services non autorisés par l'organisation. Ici, l'attaquant (**hôte C**) lance *telnet* sur un **hôte S** ou les seuls services autorisés sont *http*, *dns* et *smtp*.

Cependant, les deux types de firewall sont excellents pour rejeter des demandes de connexion provenant des services inadéquats (ou services non autorisés). Les firewalls conventionnels, éliminent les demandes au niveau de la frontière, tandis que les firewalls distribués le font au niveau de l'hôte. Le firewall distribué peut choisir de rejeter le paquet à condition que ses pairs légaux sachent utiliser IPsec ; alternativement, il répondrait à l'hôte en lui demandant de s'authentifier et là il informe ce dernier de son existence.

3.8.3 Le balayage de port

L'attaque par « balayage de port » ou « port scanning » consiste à détecter la liste des ports ouverts sur un hôte donné ou sur un réseau tout entier. Il est possible de détecter également le type et la version du système d'exploitation tournant sur l'hôte donné. Ces informations vont permettre à l'attaquant de préparer ses attaques futures. Précisons que le balayage se fait grâce à des requêtes permettant de déterminer les services fonctionnant sur l'hôte distant.

Nous distinguons plusieurs catégories d'attaques par « scan de port » parmi lesquelles nous citons : les scans furtifs, la détection des services en écoute sur un hôte distant, etc. Les attaques de type scans furtifs sont variées. Nous pouvons en citer le scan en connexion semi-ouverte, le scan Xmas, le scan NULL, le scan FIN, etc.

Les attaques de type scan furtifs visent à déterminer les ports en écoute sur un hôte tout en évitant les IDS et les firewalls se trouvant avant celui-ci. L'idée est de ne pas se faire identifier en contournant les IDS et les firewalls situés au niveau du réseau de l'hôte cible [35].

Toutefois, les firewalls traditionnels construits avec des filtres de paquets solides ne peuvent pas bien rejeter certaines attaques de type « scan furtif » appelé également « Stealth Scans ». Une technique de ce genre, utilise des paquets fragmentés qui peuvent traverser le firewall sans être examinés car les numéros de ports ne sont pas présents dans le premier fragment. Un firewall distribué rassemblerait le paquet et le rejetterait.

3.8.4 Les attaques par déni de service

Les attaques par déni de service ont pour but d'empêcher le bon fonctionnement des moyens de communications (voir section 1.2.2). Elles peuvent viser à saturer un routeur ou un hôte afin de le « crasher », ou en préambule à une attaque massive. Ces types d'attaques sont faciles à mettre en place et difficiles à empêcher [36].

Cependant il est impossible de réunir toutes les attaques par déni de services dans une corbeille. C'est dire qu'il y a une grande variété de ce type d'attaques et que ni l'un, ni l'autre modèle de firewalls ne peut gérer toutes les formes de ce type d'attaques. L'attaque « smurf » en est un exemple. Elle consiste à usurper l'adresse source d'une requête ICMP et à envoyer ce paquet en broadcast à des machines d'un réseau qui vont répondre en masse à l'adresse victime usurpée, saturant totalement la bande passante que lui délivre son fournisseur d'accès à Internet (FAI). Aucun des deux types de firewalls n'offre une défense efficace, face à ce type d'attaque.

Cependant, en changeant la topologie, les deux peuvent être moyennement efficaces. En protégeant sa ligne d'accès avec un firewall traditionnel, il est possible de bloquer cette

attaque. Pour résoudre ce problème, les firewalls distribués permettent aux hôtes de se connecter via différentes liaisons d'accès. Aujourd'hui, il est possible d'utiliser des systèmes de détection d'intrusion distribués pour faire face à ce type d'attaque.

3.8.5 Les attaques internes

Les attaques internes se passent à l'intérieur du réseau local. Elles sont mises en œuvre par des employés mal intentionnés et peuvent être actives ou passives. Ce type d'attaques peut revêtir différentes formes. Tout dépend de l'objectif de l'attaquant. Les attaques internes visent généralement des informations spécifiques et exploitent soit des points d'entrée établis, soit des failles méconnues. Les attaques internes peuvent être plus difficiles à détecter que les tentatives d'intrusions provenant de l'extérieur [37].

L'une des plus grandes faiblesses des firewalls distribués, est de ne pas faire confiance aux utilisateurs. Que se passerait-il, si quelqu'un change la stratégie des fichiers au détriment de celle qui a été établie pour ces fichiers?

Bien qu'il y ait des mesures techniques qui peuvent être prises, elles sont limitées dans leurs capacités à faire face aux esprits mal intentionnés. De même, les firewalls traditionnels sont facilement corrompus par un interne non coopératif. SSH peut être utilisé pour percer le tunnel des ports TCP, le chiffrement de bout en bout peut masquer le trafic, etc. En d'autres termes, un interne qui souhaite violer la stratégie du firewall, peut le faire avec n'importe quel type de firewall. Comme l'a dit Marcus Ranum [1] qui est considéré comme un des pionniers du concept de firewall, « Vous ne pouvez pas résoudre les problèmes mondains avec des logiciels ».

D'un autre côté, on peut réduire ces menaces ou attaques avec les firewalls distribués, en créant des petits groupes d'utilisateurs. Ainsi, on peut restreindre l'accès à un fichier serveur seulement aux utilisateurs qui le sollicitent, plutôt que de laisser n'importe quel utilisateur interne l'utiliser.

3.9 Etude de quelques firewalls distribués existants

Le développement et la recherche continus dans le domaine des technologies de firewalls, ont montré de façon progressive, l'ajout de mécanismes et services aux systèmes de firewalls

traditionnels. Cependant, le concept de *firewall distribué* [1] introduit pour la première fois par Bellovin en 1999, constitue une des principales innovations en sécurité réseau. Depuis, plusieurs instituts, constructeurs de logiciels/matériels et chercheurs, ont travaillé pour faire des firewalls distribués une réalité. Notons qu'actuellement il n'y a pas de méthode standard pour implémenter un firewall distribué. Ainsi, plusieurs implémentations ont été proposées. Celles-ci sont disponibles sous forme de solutions logicielles ou matérielles. Des implémentations logicielles de firewall distribué existent depuis, sous forme de projets de recherche. Le *prototype utilisant Keynote* qui a été envisagé dans [1] et implémenté dans [22], en est un exemple. Comme autres projets de recherche, nous pouvons citer le *prototype utilisant les filtres IP* [38]. Il existe cependant peu d'implémentations logicielles propriétaires : *TheGreenBow firewall distribué* et celui proposé par *F-Secure* en sont des exemples [39, 40]. Les seules solutions matérielles connues, sont les EFW de 3Com qui sont disponibles sur le marché à partir de 3Com et l'ADF, un projet de recherche qui se base sur le concept EFW. Dans la suite nous présentons l'implémentation de référence [22] à savoir *le prototype qui utilise Keynote*, puis *le prototype utilisant les filtres IP*, les deux implémentations matérielles que sont *EFW* et *ADF*, et enfin les deux implémentations logicielles que sont F-Secure Distributed Firewall et THEGREENBOW Disttributed Firewall.

3.9.1 Le prototype utilisant Keynote

Ce prototype de firewall distribué a été implémenté dans le cadre du projet STRONGMAN de l'université de Pennsylvanie. Dans ce prototype OpenBSD fut choisi comme système d'exploitation, parce que c'est une plateforme attrayante pour développer des applications sécurisées, car facilitant une bonne intégration des mécanismes et des librairies de sécurité (IPsec, SSL, Keynote, etc.). *Keynote* fut choisi comme langage de spécification de stratégie. Il fut aussi utilisé pour envoyer des informations d'authentification à travers un réseau non fiable. *IPsec* fut utilisé pour authentifier les utilisateurs et les hôtes, protéger le trafic et distribuer la stratégie. Ce prototype comprend trois composants :

- Un *ensemble d'extensions du noyau*, qui implémente les mécanismes d'application de la stratégie ;
- Un démon *de stratégie*, qui tourne au niveau de l'espace utilisateur, qui implémente les stratégies du firewall distribué ;
- Un *module de gestion de périphériques*, utilisé pour la communication dans les deux sens entre le noyau et le démon de stratégie.

Ce prototype a été mis en place pour contrôler uniquement les connexions TCP. Dans le système d'exploitation UNIX, les utilisateurs créent et autorisent des connexions TCP sortantes et entrantes, en utilisant les procédures systèmes à savoir *connect* et *accept*. Lorsqu'un hôte initialise une demande de connexion, il invoque la procédure système *connect* et quand il reçoit une demande de connexion, il invoque la procédure système *accept*. Puisque tout utilisateur a accès à ces procédures système, un mécanisme de filtrage est nécessaire pour s'assurer de la validité de la connexion établie ou acceptée.

Les auteurs ont choisi d'implémenter ce mécanisme de filtrage dans l'espace noyau. Les éléments d'entrée, qui peuvent être l'identifiant de l'utilisateur qui a initialisé la connexion ou l'adresse et le port de destination, sont fournis au démon de stratégie. Ce dernier fait appel au système Keynote, pour évaluer l'entrée en utilisant un ensemble de stratégies et d'informations d'authentification, puis renvoie sa réponse. Cette réponse positive ou négative, est envoyée au noyau, qui décide si la connexion TCP doit être établie ou pas.

Si par exemple, le système Keynote n'autorise pas la connexion en utilisant l'entrée qu'il a reçue, le démon de stratégie contactera un serveur distant (serveur web, serveur de bases de données distribuées, etc.) pour acquérir les informations d'authentification requises. Celles-ci seront utilisées en conjonction avec des informations fournies par le noyau, pour réexaminer la demande. Si elle est refusée une seconde fois, alors la connexion est immédiatement rejetée.

3.9.1.1 Avantages du prototype utilisant Keynote

Le système de firewall distribué évoqué précédemment n'est pas un système complet, c'est juste un prototype. Cependant il présente des avantages [22, 41, 42, 43] :

- Il a permis d'examiner l'application des langages de stratégies de haut niveau pour la gestion des réseaux de longue portée. Keynote est utilisé comme langage commun pour exprimer des stratégies qui peuvent être distribuées dans des applications et des systèmes variés. Ainsi il peut être utilisé pour exprimer les stratégies pour des applications particulières et les informations d'authentification correspondantes peuvent être distribuées à travers les mêmes canaux (IPsec, serveur web, etc.) ;

- Tel qu'évoqué précédemment, le démon de stratégie tourne comme un processus dans l'espace utilisateur et communique avec le noyau via un module de gestion de périphériques. Ce concept élargit la flexibilité du système actuel et permet une expérimentation plus facile ;

- Ce prototype permet de définir des stratégies pour plusieurs applications, plutôt que juste des règles de filtrage de paquets ;

- Il permet la délégation d'autorité, qui est adaptée à l'administration décentralisée (Puisque Keynote permet de construire des arborescences de confiance arbitraire) ;

- Les travaux de R. Chinchani et Derek Atkens [30] ont permis d'étendre les fonctionnalités de ce prototype quelques années plu tard. En effet, ceux-ci ont ajouté des modules supplémentaires dans l'architecture du *prototype utilisant keynote* afin que ses hôtes puissent adapter leurs stratégies de sécurité en fonction des intrusions ou tentatives d'intrusion qui s'opèrent dans leurs environnements.

3.9.1.2 Inconvénients du prototype utilisant Keynote

Malgré ces avantages considérables, ce prototype présente aussi des inconvénients :

- Le système actuel se focalise uniquement sur les connexions TCP. En d'autres termes, le contrôle est effectué selon la connexion et non selon le paquet. Les concepteurs prévoient d'ajouter un filtre IP comme mécanisme pour un contrôle plus fin c'est à dire au niveau paquet (peut être basé sur un logiciel de filtrage existant, comme IPF) ;

- Dans le système actuel, les mises à jour de stratégies affectent seulement les nouvelles connexions et non celles déjà existantes. Les concepteurs aimeraient ajouter un mécanisme d'annulation qui va parcourir la liste de toutes les connexions et ré-appliquer les stratégies. Les connexions qui ne se conformeraient pas à ces changements, seront interrompues par le noyau ;

- La méthode de recherche des informations d'authentification mérite d'être améliorée. Les utilisateurs ont besoin d'une méthode pour rechercher quelles informations d'authentification ils doivent accompagner avec leurs demandes pour une connexion.

Les concepteurs prévoient de résoudre ce problème en utilisant la procédure système déjà existante, *setsockopt* [22, 41, 42, 43].

3.9.2 Le prototype utilisant les filtres IP

Ce prototype a été proposé par Aydin Eraslan et al. [38] pour un environnement ou les hôtes tournent principalement sous Linux. Dans ce firewall distribué, le filtre *iptables* est utilisé pour définir la stratégie de sécurité des hôtes. Ensuite, celle-ci est distribuée à tous les hôtes via *ssh et sudo*. Ce prototype de firewall distribué est constitué d'une architecture à trois composants :

- Une console d'administration qui permet d'administrer tous les hôtes y compris les routeurs. Cette console permet également de déléguer des droits d'administration aux utilisateurs ;

- Un serveur central qui se connecte sur chaque hôte de façon périodique et déploie les stratégies de sécurité spécifiques à chacun et coordonne avec les points d'application des stratégies. Ce serveur se charge également de centraliser les logs des hôtes de façon périodique et les stocke dans une base de données ;

- Des hôtes qui sont déployés sur le réseau. Ces hôtes appliquent la stratégie de sécurité qui leur est délivrée à partir de la console d'administration.

Toutefois, ce prototype dispose de fonctionnalités permettant de prendre en compte d'autres filtres tels qu'ipf pour la définition des stratégies de sécurité pour d'éventuels hôtes tournant sur des systèmes d'exploitation Unix tels que *Solaris* et/ou OpenBSD.

3.9.2.1 Avantages du prototype utilisant les filtres IP

Le prototype utilisant les filtres IP n'est pas un firewall distribué complet. Il présente les avantages ci-après :

- Administration centralisée de la stratégie de sécurité. Cela permet d'obtenir une cohérence dans la spécification de la stratégie de sécurité pour l'ensemble des hôtes du firewall distribué et également de diminuer le nombre d'attaques internes ;

- Définition de l'interopérabilité entre les systèmes. En effet, plusieurs hôtes tournant avec des systèmes d'exploitation différents peuvent interagir entre eux et de façon cohérente. Ce qui constitue un système distribué hétérogène adéquat ;

- Possibilité d'analyser le trafic réseau à l'aide de la base de données située sur le serveur central. Cela permettra de connaître les différentes intrusions et/ou tentatives d'intrusions à l'endroit des hôtes du firewall distribué. Ce qui donne à l'administrateur réseau, la possibilité de définir de nouvelles stratégies de sécurité pour les hôtes concernés ;

- Architecture extensible permettant de prendre en compte d'autres filtres autres que *iptables* et *ipf*.

3.9.2.2 Inconvénients du prototype utilisant les filtres IP

Le firewall distribué proposé par Aydin Eraslan et al. [38] présente un certain nombre d'inconvénients parmi lesquels nous citons les principaux :

- Absence de mécanismes permettant de prendre en compte la stratégie de sécurité des hôtes de façon dynamique. En effet, si un hôte est attaqué, il n'y a pas de mécanismes dynamiques permettant de modifier sa stratégie de sécurité afin de bloquer cette attaque. Il faut attendre à ce que l'administrateur de sécurité constate cette attaque au niveau de la base de données et qu'il définisse une nouvelle stratégie de sécurité pour ce type d'attaque ;

- Dans le système actuel, les mécanismes de distribution de clés et d'authentification des utilisateurs sont élémentaires. Les auteurs prévoient l'ajout d'informations d'authentification pour les utilisateurs afin de renforcer leur processus d'authentification. Ils prévoient également l'ajout de mécanismes solides pour la distribution des clés afin de renforcer la sécurité du processus d'échange entre les hôtes.

3.9.3 Le firewall intégré

Le firewall intégré (EFW) ou « embedded firewall » est un firewall distribué réalisé sur du matériel, qui applique l'ensemble des règles sur l'interface réseau (NIC). EFW fournit un filtrage simple de paquets et un serveur de stratégies centrale [41, 44, 45]. L'architecture du

firewall intégré comprend quatre composants : la carte réseau EFW pour chaque hôte protégé, un pilote EFW, un assistant et le serveur de stratégies EFW. Cette architecture reste valable pour le firewall distribué autonome ou « ADF » sauf qu'au lieu des cartes EFW, il est utilisé dans ce cas des cartes ADF (Figure 3.4).

L'assistant génère un ensemble de messages ou « heartbeats » qu'il envoie au serveur. Ces messages permettent au serveur de stratégies de s'assurer que la carte réseau fonctionne normalement. Cet assistant n'est pas indispensable au fonctionnement de la carte EFW/ADF, car même s'il est en panne, la carte EFW/ADF appliquera la stratégie délivrée par le serveur central. Ce dernier dispose d'une interface graphique (GUI) centralisée qui permet de gérer plus de 4000 cartes EFW.

L'interface graphique gère les bases de données de stratégie et d'audit. Elle permet à l'administrateur de connaître le statut des différentes cartes réseaux, de créer et de modifier l'ensemble des règles. Lorsque les audits sont activés, le log de chaque paquet traité par la carte réseau (NIC) est envoyé au serveur de stratégies. Ces logs d'audit aident l'administrateur à générer la stratégie à appliquer. Le mécanisme d'audit peut également être utilisé pour détecter une intrusion.

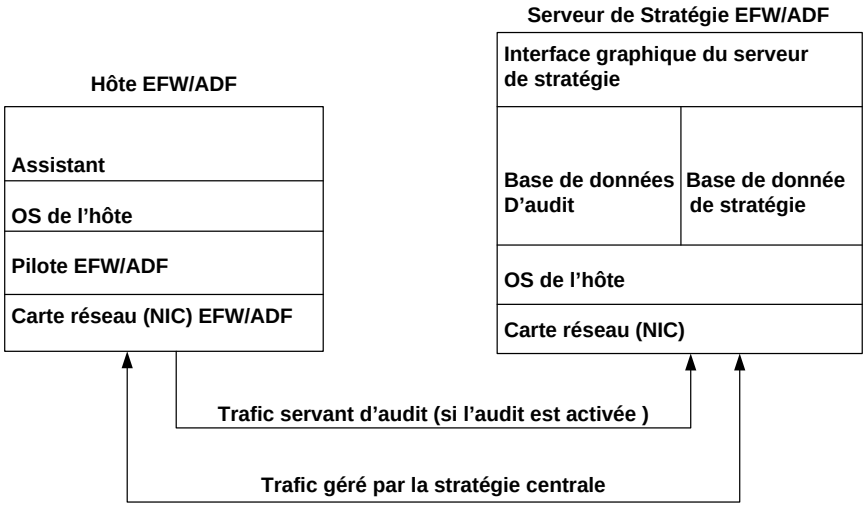

Figure 3.4 : Architecture EFW/ADF

Un des premiers buts du projet EFW est de réduire le coût effectif pour le déploiement dans les réseaux à large portée. Pour atteindre cet objectif, le périphérique doit être « rapide, simple et moins cher ». C'est pourquoi EFW est implémenté sur une carte réseau (3CR990)

qui ne coûte pas cher. Bien que cela puisse être onéreux, des équipements conçus spécialement pour le filtrage de paquets pourraient assurer une grande performance et probablement être capables de résister à des attaques de type « flood ». Ces attaques consistent à saturer un hôte dans le but de l'empêcher de bien fonctionner.

3.9.3.1 Les avantages du firewall intégré

Le firewall distribué précédemment décrit présente un certain nombre d'avantages :

- Il n'y a pas une solution complète sur la sécurité des employés mobiles. L'idée des concepteurs est de déployer une carte réseau permettant d'appliquer une stratégie de sécurité adéquate, selon que l'utilisateur soit chez lui ou en déplacement. L'utilisateur n'a pas à s'inquiéter de faire d'éventuelles configurations à chaque fois qu'il se connecte à un réseau autre que le sien. Ceci peut rassurer l'administrateur que l'utilisateur n'est pas capable de désactiver le firewall et que l'ordinateur portable n'apportera pas de virus dans le réseau interne ;

- Les administrateurs n'ont pas à s'inquiéter des nouvelles vulnérabilités rencontrées sur les systèmes d'exploitation des machines protégées, du fait que les cartes réseaux EFW sont indépendantes du système d'exploitation ;
- La stratégie ne peut pas être consciemment ou inconsciemment désactivée ou modifiée à partir de l'hôte protégé, mais seulement à partir du serveur central. Puisque les cartes réseaux sont contrôlées par ce dernier, un attaquant ou un logiciel malveillant (virus, chevaux de Troie, etc.) ne peut les reprogrammer, ni contourner la stratégie ;

- Il est facile d'établir l'ensemble des règles du firewall distribué. Ce modèle permet de gérer des milliers d'hôtes équipés de cartes EFW dans une organisation (entreprise, université, etc.).

3.9.3.2 Les inconvénients du firewall intégré

Le firewall distribué intégré présente un certain nombre de limites considérables :

- Actuellement, ni les micro-logiciels installés sur les cartes réseaux, ni le logiciel serveur de stratégies n'est disponible sur un système d'exploitation autre que Windows. En particulier dans les grands centres de traitement informatique, où l'on

trouve fréquemment des environnements mixtes. Ceci limite l'utilisation de ce produit ;

- Il n'y a pas d'adaptateurs sans fil disponibles sur la famille de ce produit. Avec l'extension rapide des points d'accès public sans fil, les employés mobiles ne seront pas à l'aise avec les adaptateurs classiques câblés ;

- Les hôtes du firewall distribué intégré ne peuvent pas modifier leurs stratégies de sécurité de façon dynamique afin de s'adapter aux intrusions ou tentatives d'intrusion qui se produisent dans leurs environnements ;

- Les cartes réseaux sont capables de filtrer seulement les paquets IP. Les autres protocoles peuvent être rejetés ou ignorés mais uniquement selon un ensemble de critères ou sur aucune base. Les cartes réseaux ne peuvent pas faire du filtrage dynamique sur les paquets. Des règles sont spécifiées pour autoriser le trafic sortant à passer par les ports non standard (ces ports ont une valeur comprise entre 1024 et 65535). L'absence de filtrage dynamique fait que certains protocoles tels que FTP, qui incluent les informations sur le port dans le champ payload du paquet, ne fonctionnent pas correctement ;

- Lorsque l'audit est activé à un niveau supérieur, et qu'il y ait un nombre important de clients, les logs qui transitent à travers les interfaces réseaux vers le serveur de stratégies consomment énormément de bande passante.

3.9.4 Le firewall distribué autonome

Le firewall distribué autonome (ADF) ou « autonomic distributed firewall » [46, 47, 48] est venu après EFW. Ayant les mêmes fonctionnalités que EFW, il dispose en plus d'un mécanisme pour créer des canaux de communication cryptés appelés groupe privé virtuel (VPG) ou « virtual private group » [46, 49] qui assurent la confidentialité, l'intégrité et l'authentification de l'émetteur. En effet, les VPGs sont des réseaux privés virtuels (VPN), qui fonctionnent au niveau de la carte réseau (NIC). Le cryptage des paquets est complètement transparent à l'hôte, permettant à n'importe quelle application de bénéficier d'une protection supplémentaire. La gestion et la distribution des clés sont contrôlées par le serveur de stratégies à cause des limites de la carte réseau en matière de traitement.

3.9.4.1 Les avantages du firewall distribué autonome

Le firewall distribué autonome présente un certain nombre d'avantages parmi lesquels nous avons :

- Tout comme les cartes EFW, les cartes ADF sont aussi indépendantes du système d'exploitation. Ainsi, les utilisateurs mal intentionnés ou les logiciels malveillants (virus, chevaux de Troie, etc.) ne peuvent pas les désactiver ;

- La stratégie de sécurité reste centralisée, mais elle est envoyée à tous les points finaux qui se chargent de l'appliquer. Cela permet de diminuer le nombre d'attaques internes. On retrouve la même fonctionnalité au niveau de l'EFW ;

- La gestion de la stratégie est simple tout comme au niveau de l'EFW car la stratégie est indépendante de la topologie. Puisque les points d'application se trouvent au niveau de l'hôte (client ou serveur), la spécification des règles ne dépend pas de l'endroit où se trouve l'hôte dans le réseau ;

- Il offre un mécanisme de sécurité plus solide que celui de l'EFW, car disposant d'une fonctionnalité supplémentaire appelée VPG. Comme les VPN, les VPGs assurent la confidentialité, l'intégrité et l'authenticité de l'émetteur. Ainsi, les utilisateurs mobiles sont mieux sécurisés durant leurs activités ;

- Le trafic réseau peut être crypté, décrypté et la vérification de l'intégrité n'influe pas sur la charge du processeur de l'hôte. Ceci est dû au fait que le traitement est effectué au niveau de la carte réseau de l'hôte ;

- Peut facilement stopper les vers avant qu'ils se propagent sur le réseau. Les vers sont des programmes autonomes, qui ne se dissimulent pas dans d'autres programmes. Leur objectif est de se reproduire d'ordinateur en ordinateur, habituellement au travers du réseau, et souvent y produire des dégâts en cascade [50].

3.9.4.2 Les inconvénients du firewall distribué autonome

Bien que cette technologie dispose d'un nombre d'avantages considérables, elle présente aussi des inconvénients :

- Tout comme EFW, ADF est vulnérable face aux attaques de déni de service tels que flood, qui consiste à envoyer des requêtes massives à destination de la cible dans le but de la saturer [51]. Ceci en vertu de la nature des cartes réseaux (3CR990) sur lesquelles ils sont implémentés ;

- On ne peut pas effectuer de modifications sur le matériel et ses pilotes sans passer par le vendeur. Ce qui peut entraîner des coûts supplémentaires. Ceci reste valable pour EFW ;

- Les fonctionnalités supplémentaires (VPGs, gestion des clés, etc.) dont il dispose, le rendent plus cher que EFW ;

- Dans les réseaux disposant d'un débit de 100mbps, EFW peut être plus performant que ADF. Cela est dû au traitement (cryptage/décryptage) supplémentaire qu'il doit effectuer sur tous les paquets VPGs [52, 53] ;

- Les hôtes du firewall distribué autonome ne peuvent pas modifier leurs stratégies de sécurité de façon dynamique afin de s'adapter aux intrusions ou tentatives d'intrusion qui se produisent dans leurs environnements ;

3.9.5 Présentation de F-Secure Distributed Firewall

Il s'agit d'un firewall distribué sous forme logicielle conçu par la société F-Secure. Ce système comprend un serveur et des clients. Au niveau du serveur, une interface appelée Policy Manager permet d'administrer les logiciels installés au niveau des clients. L'administrateur réseau configure les règles de filtrage à l'aide du Policy Manager, ces règles de filtrage sont ensuite distribuées à tous les hôtes du réseau. En plus, l'administrateur réseau ou l'utilisateur local peut définir le type de trafic autorisé à accéder ou à sortir de l'hôte client. Egalement, F-Secure Distributed Firewall offre un mécanisme de contrôle d'applications. Ce mécanisme contrôle les types d'applications autorisés à accéder au réseau. Avec ce mécanisme, les utilisateurs peuvent décider quelles sont les applications qui peuvent accéder à leurs machines. Par exemple, les utilisateurs peuvent autoriser l'accès à leurs machines uniquement aux applications considérées comme étant fiables et bloquer les applications inconnues [40].

3.9.5.1 Avantages de F-Secure Distributed Firewall

F-Secure Distributed Firewall présente beaucoup d'avantages parmi lesquels nous avons :

- Un mécanisme de filtrage de paquets pour tous les hôtes du réseau y compris les hôtes distants et mobiles ;
- Une interface d'administration (appelé Policy Manager) qui permet d'administrer tous les logiciels firewalls installés au niveau des hôtes ;
- Un mécanisme de déclenchement d'alertes au niveau des hôtes en cas d'attaques ou d'événements suspects. Ces alertes peuvent être également remontées au niveau de la console d'administration.

3.9.5.2 Inconvénients de F-Secure Distributed Firewall

Comme toute technologie, F-Secure Distributed Firewall présente également des inconvénients parmi lesquels, les principaux sont :

- Une absence de mécanismes dynamiques permettant de modifier la stratégie de sécurité des hôtes pour qu'ils puissent se protéger contre les intrusions ou tentatives d'intrusion qui s'opèrent dans leurs environnements ;
- Un hacker qui prend le contrôle de l'hôte peut désactiver le logiciel firewall installé sur l'hôte ;
- F-Secure Distributed Firewall est disponible sur la plateforme Windows uniquement.

3.9.6 Présentation de THEGREENBOW Distributed Firewall

A l'image de F-Secure Distributed Firewall, le firewall distribué TGB (c'est-à-dire THEGREENBOW) comprend également une interface d'administration et des logiciels clients installés sur chaque hôte. Cette interface d'administration permet la gestion du parc de firewalls personnels installés, la configuration des règles de sécurité de ces firewalls, indépendamment ou par groupe, la compilation des logs et des alarmes de chaque firewall. Le firewall distribué TGB sécurise toutes les communications entrantes et sortantes du poste à

protéger, que ce dernier soit connecté directement au réseau d'entreprise, ou qu'il soit connecté directement à Internet. Il implémente une technologie de filtrage à deux niveaux qui garantit le contrôle de toutes les connexions entrantes et de toutes les applications sortantes. Ces dernières sont de plus signées pour protéger l'hôte contre des programmes malicieux tels que les chevaux de Troie [39].

3.9.6.1 Avantages du firewall TGB

Le firewall distribué TGB présente beaucoup d'avantages parmi lesquels nous pouvons citer :

- Protection efficace contre les attaques (chevaux de Troie, déni de service, etc.) ;
- Protection contre les scans d'adresses et de ports ;
- Contrôle des applications sortantes et des connexions entrantes ;
- Signature MD5 des applications filtrées ;
- Contrôle de conformité des paquets entrants avant leur transmission au système d'exploitation de l'hôte ;
- Centralisation de la configuration des hôtes équipés du firewall TGB.

3.9.6.2 Inconvénients du firewall TGB

Le firewall distribué TGB n'échappe pas à la philosophie selon laquelle aucune technologie n'est infaillible. Ainsi, il présente également des inconvénients :

- Tout comme F-Secure Distributed Firewall, il y a une absence de mécanismes dynamiques permettant de modifier la stratégie de sécurité des hôtes pour qu'ils puissent réagir aux intrusions ou tentatives d'intrusions qui se produisent dans leurs environnements ;
- Egalement comme tout firewall logiciel, un hacker qui prend le contrôle de l'hôte peut désactiver le logiciel firewall qui tourne à son sein ;

- TGB est disponible sur la plateforme Windows uniquement à l'image de F-Secure Distributed Firewall.

3.10 Comparaison des firewalls distribués existants

Le tableau 3.2 compare les Fws distribués étudiés. Ces firewalls sont classés en trois catégories : les firewalls existants sous forme de projet de recherche (le prototype utilisant keynote étant un exemple), les firewalls matériels (EFW et ADF) et les firewalls logiciels (F-Secure Distributed Firewall et THEGREENBOW Distributed Firewall).

	Les Firewalls Distribués Existants				
	Les projets de Recherche	**Les Firewalls Matériels**		**Les firewalls Logiciels**	
	Le prototype utilisant Keynote	**EFW**	**ADF**	**F-secure Distributed Firewall**	**THEGREENBOW Distributed Firewall**
Avantages	- Stratégie de sécurité définie de manière centralisée -Possibilité de déléguer des droits aux utilisateurs -Mécanismes d'application de la stratégie se trouvent sur le noyau openBSD -Impossibilité de contourner les filtres	- Stratégie de sécurité définie de manière centralisée -Mécanismes d'application de la stratégie se trouvent sur les cartes réseaux -Impossibilité de contourner les filtres	- Stratégie de sécurité définie de manière centralisée -Mécanismes d'application de la stratégie se trouvent sur les cartes réseaux -Impossibilité de contourner les filtres	- Stratégie de sécurité définie de manière centralisée -Mécanismes d'application de la stratégie se trouvent sur les logiciels clients	- Stratégie de sécurité définie de manière centralisée -Mécanisme d'héritage de règles pour les utilisateurs d'un groupe -Mécanismes d'application de la stratégie se trouvent sur les logiciels clients
Inconvénients	-Nœuds réseaux doivent tourner sous openBSD	-Nœuds réseaux doivent être équipés de cartes 3Com	-Nœuds réseaux doivent être équipés de cartes 3Com	-Possibilité de contourner les filtres -Nœuds réseaux doivent tourner sous Windows	-Possibilité de contourner les filtres -Nœuds réseaux doivent tourner sous Windows

Tableau 3.2 : Comparaison des Firewalls distribués existants

Le *prototype utilisant keynote* permet à des utilisateurs de déléguer des privilèges à d'autres utilisateurs pour accéder aux machines et services auxquels ils ont accès. Bien que ce mécanisme soit optionnel et puisse être désactivé par l'administrateur, il offre une grande flexibilité au système et permet de décentraliser l'administration. Ce qui n'est pas le cas pour les firewalls distribués matériels (EFW et ADF) où les utilisateurs reçoivent les droits à partir du serveur central. Le firewall distribué TGB possède un mécanisme d'héritage de règles pour les utilisateurs d'un groupe. Ce mécanisme est similaire à celui de délégation de droit du prototype utilisant Keynote. F-Secure distributed firewall n'offre pas ce type de mécanisme.

Au niveau de EFW et ADF, les mécanismes d'application de la stratégie sont implémentés au niveau des cartes réseaux (NIC), alors que pour le prototype qui fonctionne sous OpenBSD ces mécanismes sont implémentés dans le noyau du système d'exploitation. F-Secure distributed firewall et le firewall TGB implémentent les mécanismes d'application de la stratégie au niveau de chaque logiciel client.

Pour s'assurer de la validité d'une connexion établie ou acceptée, Bellovin et ses collègues [22] ont implémentés le mécanisme de filtrage (basé sur la stratégie définie par l'administrateur) dans le noyau du système d'exploitation. Ainsi, un utilisateur ne peut jamais contourner les filtres. C'est dans la même lancée que le mécanisme de filtrage au niveau de EFW et ADF a été implémenté sur les cartes réseaux afin que ces dernières ne puissent pas être contournées. Cependant, les firewalls distribués logiciels que nous avons vus ne peuvent pas échapper à cette philosophie. En effet, dès qu'un hacker prend le contrôle de l'hôte client, il peut facilement désactiver le firewall logiciel qui y tourne.

Les trois catégories de systèmes délèguent l'application de la stratégie de sécurité à chaque hôte du réseau. Les stratégies de sécurité sont définies dans un certain langage de spécification de stratégie et distribuées à tous les points finaux. Ces systèmes ont des exigences sur les nœuds du réseau. ADF et EFW exigent que les nœuds du réseau soient équipés de cartes réseaux 3Com, pendant que le *prototype utilisant keynote* exige qu'ils tournent sur OpenBSD. Cependant, les firewalls distribués logiciels vus précédemment exigent à ce que les nœuds du réseau tournent sur Windows. Toutefois, notons que la mise en place du *prototype utilisant keynote* est très complexe car nécessitant des modifications des procédures systèmes, comme *connect, accept,* etc., dans le noyau du système d'exploitation OpenBSD.

3.11 Conclusion

Dans ce chapitre, nous avons abordé le concept de firewall distribué. Nous avons vu que ce concept est beaucoup plus sûr que celui de firewall traditionnel. Mais néanmoins, comme toute technologie, le firewall distribué en dépit de ces avantages, présente aussi des inconvénients parmi lesquels nous avons étudié les principaux (voir section 3.5). Ce concept n'est pas venu pour remplacer le firewall traditionnel dans toutes les architectures, mais plutôt pour le compléter. Toutefois, cette étude sur les firewalls distribués nous a permis de dresser le tableau ci-dessous, présentant les réactions des firewalls distribués décrits précédemment (voir section 3.9) face aux différents problèmes.

Firewalls Distribués	**Les problèmes**						
	Protection des anciennes applications	*Coût du déploiement d'un hôte*	*Conformité de la stratégie de sécurité*	*Détection d'intrusions*	*Contrôle des failles de la topologie*	*Adaptabilité de la stratégie de sécurité des hôtes*	*Authentification basée sur des adresses IP*
Prototype Utilisant Keynote	Non	NA	Non	Non	Non	Oui	NA
Prototype utilisant les filtres IP	Oui	NA	Non	Non	Non	Non	NA
EFW	Oui	NA	Non	Non	Non	Non	NA
ADF	Oui	NA	Non	Non	Non	Non	NA
F-Secure Distributed firewall	Oui	NA	Non	Non	Non	Non	NA
TGB	Oui	NA	Non	Non	Non	Non	NA

Tableau 3.3 : Aperçu des problèmes solubles par le firewall distribué

Au vu de ces résultats, il est impossible de concevoir un système ne comportant aucune faille. Ainsi, en guise de contribution dans ce travail nous proposons une solution à une des insuffisances de l'un des types de firewall distribué. C'est ainsi que nous avons pris l'exemple du *prototype utilisant les filtres IP*, identifié un de ces problèmes qu'est l'adaptabilité de la stratégie de sécurité des hôtes et essayé de proposer une solution pour que ses hôtes puissent modifier leurs stratégies de sécurité, afin de contrer les intrusions ou tentatives d'intrusions qui s'opèrent dans leurs environnements. Le chapitre suivant présente notre contribution.

Chapitre 4

Ebauche de proposition de solution

Dans ce chapitre, nous exposons la solution que nous proposons pour résoudre le problème identifié qu'est l'adaptabilité de la stratégie de sécurité des hôtes dans les firewalls distribués.

4.1 Analyse du problème

Une des implémentations les plus accessibles des firewalls distribués est basée sur l'utilisation d'iptables. Nous nous proposons de résoudre le problème dans le cadre d'un firewall distribué sous *iptables*.

Dans les firewalls distribués sous *iptables* tel que celui proposé dans [38], le serveur de stratégies central génère et envoie une stratégie de sécurité à chaque hôte. Cette stratégie est statique, en ce sens que l'hôte qui la reçoit ne peut pas la modifier, pour réagir et s'adapter aux changements (attaques, intrusions) qui peuvent s'opérer dans son propre environnement. Et il n'y a pas de mises à jour à partir du serveur de stratégie puisqu'il ne voit pas le trafic interne de l'hôte et par conséquent ne peut prendre aucunes décisions.

Dans beaucoup de cas le contrôle du firewall au sein de l'hôte ne peut pas le protéger totalement contre une attaque. Par exemple c'est le cas d'un hôte avec une stratégie de sécurité autorisant l'accès au port 80 du serveur web à partir de n'importe quel hôte d'Internet. Il vérifiera seulement si le port de destination du paquet est assigné au service http, mais il ne vérifiera pas si le contenu du paquet représente une attaque ou une violation de la

stratégie de sécurité de l'organisation (Université, entreprise, etc.). Cela signifie qu'il ne stoppera pas un hacker qui tente d'avoir un accès illégitime au système en exploitant une faille du serveur web.

Chaque hôte doit réagir et s'adapter aux changements qui se produisent dans son environnement. En se basant sur l'information qu'il rassemble sur le trafic qu'il voit, s'il peut prendre une décision pour modifier sa stratégie de sécurité, le nombre d'attaques qui surviennent peuvent être essentiellement réduit.

En guise de contribution à ce problème, notre solution se base sur le mécanisme de sécurité proposé par J. H. Guerrero et R. G. Cardenas [54]. Nous proposons d'implémenter ce mécanisme au niveau de chaque *point final* du firewall distribué afin d'automatiser le processus de renforcement de la stratégie de sécurité des *points finaux*. L'objectif est d'élever les *boucliers de protection* que sont les *points finaux* à un niveau supérieur en cas d'attaques ou d'intrusions.

Notre solution est basée sur des entités en interaction qui se partagent des connaissances et travaillent de manière collaborative. Ces entités alors appelées agents sont subdivisées en deux : l'IDS *Snort* et l'agent *Analyzer*. L'IDS Snort surveille le trafic du *point final* et génère des alertes sous un certain format en cas de détection d'événements suspects. L'agent *Analyzer* parcourt le fichier « *alert* » généré par Snort, l'analyse et extrait une ou plusieurs adresses IP relatives à un ou plusieurs attaquants. Ensuite, il modifie la stratégie de sécurité du point final afin de les bloquer.

Nous exposons plus en détail les interactions entre les différentes entités citées précédemment dans la section *base de notre proposition* (voir section 4.4). Au vu de ce qui précède, notre étude est développée de la façon suivante :

- D'abord, nous présentons les systèmes de détection d'intrusions de façon brève, puis nous présentons le système de détection d'intrusions *Snort* ;
- Ensuite, nous présentons le mécanisme de sécurité sur lequel nous nous appuyons ;
- Enfin, nous exposons notre proposition.

4.2 Les systèmes de détection d'intrusions

L'*intrusion* est l'ensemble des actions qui ont pour but de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource. Ces actions de franchissement d'un accès non-autorisé ou de manipulation d'une ressource, peuvent être menées par un individu externe n'ayant aucun privilège sur les ressources d'un système, ou par un individu interne qui outrepasse ses privilèges.

La *détection d'intrusion* consiste à identifier les individus ou les machines qui exécutent ou tentent d'exécuter une intrusion. Les *systèmes de détection d'intrusion* (ou IDS) sont des systèmes qui tentent de réaliser la détection d'intrusion en repérant les activités anormales ou suspectes sur la cible analysée (réseau, hôte). De ce fait, un système de détection d'intrusion analyse l'ensemble du trafic entrant et sortant du système afin d'identifier des traces suspectes, qui peuvent indiquer une tentative d'accès illégitime au système [55].

Deux approches principales permettent de caractériser un système de détection d'intrusions : celle basée sur les systèmes à observer et celle basée sur la façon dont les systèmes agissent.
La première approche est divisée en deux groupes, ceux qui analysent seulement les activités d'une machine à la recherche d'attaques possibles et ceux qui le font sur un réseau entier. Le système de détection d'intrusion réseau capture et décode toutes les trames qui transitent par le segment sur lequel il est connecté.
La deuxième approche est basée sur la manière dont les systèmes agissent ; il y a deux techniques : détection de signatures et détection d'anomalies. La première technique consiste à observer le trafic sur le réseau, généralement en observant le contenu des paquets de données qui circulent d'une machine à l'autre, afin de détecter des signatures d'attaques connues. La seconde technique consiste à définir un comportement normal du système et à rechercher tout ce qui dévie de ce comportement. Ainsi, un système de détection d'intrusions par recherche de signatures connaît ce qui est mal, alors qu'un système de détection d'intrusions par analyse de comportement connaît ce qui est bien [56].

Lorsque plusieurs systèmes de détection d'intrusions répartis sur un vaste réseau communiquent les uns avec les autres, ou avec un serveur central, qui facilite une surveillance avancée du réseau en terme d'analyse d'intrusion et des données résultant des attaques instantanées, on parle de système de détection d'intrusions distribué (DIDS) [57].

4.3 Présentation de Snort

Snort est un véritable système de détection d'intrusions réseau (NIDS) open-source qui a beaucoup de fonctionnalités. Il a été écrit en grande partie par Martin Roesch et son code source peut être téléchargé sur le site web officiel de Snort [58]. Ces fonctionnalités incluent les modes sniffer (ou renifleur réseau) et logging (ou enregistreur) de paquets en plus de la détection d'intrusion. A cela s'ajoute la possibilité de le configurer pour envoyer des alertes en temps réel.

Les fonctions de *sniffer* et de *logging de paquets* sont des fonctionnalités élémentaires de Snort. La force de Snort vient de sa fonctionnalité de détection d'intrusions.

Dans le mode *Sniffer* Snort lit les paquets circulant sur le réseau et les affiche d'une façon continue sur la sortie standard. Les options pour exécuter Snort dans ce mode sont «**-d**», «**-e** » et «**-v** ». La dernière option met Snort en mode sniffing de paquets (en-têtes TCP seulement), l'option **«-d**» inclut toutes les en-têtes de couche réseau (TCP, UDP, et ICMP), et l'option «**-e**» inclut les en-têtes de couche liaison de données.

Dans le mode *logging de paquets* Snort journalise le trafic réseau dans des répertoires sur le disque. Enfin, pour exécuter Snort en mode *détection d'intrusions*, il faut spécifier l'emplacement du fichier « **snort.conf** » et l'emplacement du fichier log. Par défaut, le fichier log se trouve dans «**/var/log/snort**». Le fichier *snort.conf* contient la configuration du réseau interne, les règles, les préprocesseurs, et les *plugins* d'entrée et de sortie. Il est possible de spécifier les paquets que l'on souhaite voir journalisés au niveau de ce fichier. Notons que les préprocesseurs permettent d'améliorer les possibilités de Snort en matière d'analyse et de recomposition du trafic capturé.
Toutefois, Snort propose une architecture distribuée permettant aux sondes d'envoyer leurs alertes dans une base de données commune qui sera ensuite analysée par une application web. Ceci permet à l'administrateur de sécurité de surveiller l'évolution des attaques via un navigateur web [59, 60, 61].
En plus de son aspect distribué, l'architecture de Snort est modulaire (voir Figure 4.1) et est composée de :

- Un module de *capture du trafic et de décodage du flux* au niveau duquel le trafic est dupliqué et les données copiées à partir du réseau. Les paquets capturés sont au format de

la couche liaison (protocoles Ethernet, ATM, etc.). Le décodage du flux est réalisé par l'extraction des paquets IP à partir de la couche liaison. Les paquets IP sont ensuite stockés dans une structure mémoire adéquate ;

- Un module dans lequel se trouvent les *pré-processeurs de paquets réseau.* Dans ce module, les paquets IP obtenus par le module de décodage doivent subir des traitements afin que l'IDS réalise une analyse plus efficace et plus fiable. Les traitements appliqués permettent de parer aux méthodes d'évasion utilisées par les trafics malveillants. Il s'agit en l'occurrence de l'étape de dé-fragmentation des paquets IP. Les paquets TCP séquencés font aussi l'objet d'un ré-assemblage. D'autres analyses peuvent y être développées (analyse des sessions TCP, etc.) ;

- Un module où se trouve *le moteur de détection et d'inspection des paquets*. C'est le coeur du système de détection d'intrusions. Il applique l'ensemble des règles de détection aux paquets. La vérification des règles requiert des opérations très exigeantes en calcul, car ces dernières sont complexes (pour pouvoir détecter des attaques toujours plus complexes). Les vérifications sont orientées, soit champ de paquet (entête), soit contenu de paquet (concernant les données utiles transportées par le paquet). L'opération de recherche dans le contenu du paquet est la plus coûteuse en temps et en ressources (CPU, mémoire, etc.) ;

- Un module constitué des *modules de sortie*. Ces modules ou *plugins* de sortie se chargent de réaliser les instructions adéquates lorsqu'une règle de détection est vérifiée. Ils permettent d'imprimer un message d'alerte à l'écran, d'écrire dans un fichier *log*, ou d'envoyer un message à un serveur distant (centralisation de données). Ce serveur distant peut être une base de données [62].

L'articulation de ces différents modules autour du moteur de détection est illustrée par la figure 4.1 :

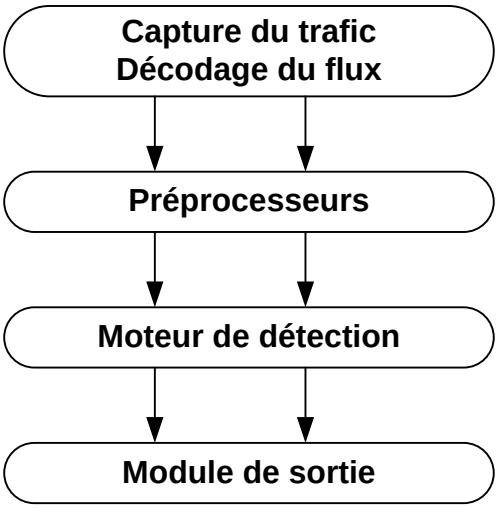

Figure 4.1 : Architecture générale de Snort

Snort dispose d'une variante open source appelée *Snort-inline.* Ce dernier s'appuie sur le firewall *Netfilter*. Il fonctionne comme un IDS. Ainsi à l'image de l'IDS, il possède une table de signatures des attaques connues. Il peut également fonctionner tout seul, tout comme il peut interagir avec Snort. Ce type de système est appelé système de Prévention d'intrusions (IPS). Ainsi quand un paquet de données entre dans le réseau, l'IPS confronte ces éléments à sa base de données de signatures. Si les données concordent, l'IPS peut supprimer le paquet ou interrompre la connexion de l'hôte suspecté (machine pirate) [63, 64].

4.3.1 Avantages de Snort

Snort est un logiciel libre qui concurrence habilement les outils de détection d'intrusions commerciaux. Ses principaux avantages sont les suivants :

- Ses signatures ont un format ouvert c'est-à-dire que l'on peut le modifier et l'adapter à ses besoins. Ce format est devenu un standard que les IDS commerciaux commencent à intégrer. En effet, les IDS commerciaux ont tendance à cacher le contenu de leurs signatures (la séquence précise que recherche le logiciel pour identifier une attaque). Cela oblige les clients à dépendre d'eux pour bénéficier de nouvelles règles de détection d'intrusions. Snort offre un format ouvert et documenté qui permet d'écrire les règles de son choix ;

- La gratuité au niveau des licences permet d'installer des sondes Snort en tout point du réseau. La prolifération de ces sondes peut tout de même rapidement devenir un handicap dans un réseau vaste en ce qui concerne l'étude des rapports ;

- Snort peut interagir avec d'autres IDS tels que Prelude. En effet Snort peut être configuré pour logguer ces alertes au niveau de la base de données de Prelude ;

- Snort est le système de détection d'intrusions le plus déployé au monde avec environ 3 million de téléchargements et plus de 100.000 utilisateurs actifs [58].

4.3.2 Inconvénients de Snort

Bien que Snort présente des avantages, comme toute technologie il présente également des inconvénients. Parmi ceux-ci, nous avons :

- Un des problèmes rencontrés avec Snort est l'obtention des rapports cités précédemment même si des interfaces tel qu'ACID (Analysis Console for Intrusion Databases) existent bien. ACID permet de générer des rapports à partir des alertes de Snort et de stocker les événements dans une base de données Mysql. ACID, en lui-même est néanmoins assez difficile à installer ;

- Les fichiers d'alertes générés par Snort sont difficiles à analyser de manière automatisée. En effet lorsque l'on décide de stocker les informations sur les attaques dans un fichier au détriment d'une base de données, il est difficile d'avoir des outils permettant d'extraire les informations nécessaires sur ces attaques au niveau du fichier alertes ;

- Les règles de détection d'intrusions utilisées par Snort peuvent être obsolètes avec l'introduction permanente de nouvelles attaques. Ceci nécessite une mise à jour périodique des règles de Snort ; cette fonctionnalité est payante en ce sens qu'il faudra s'abonner au niveau d'un fournisseur pour obtenir de nouvelles mises à jour à chaque fois que des nouvelles attaques apparaissent ;

- L'absence d'interface de configuration constitue un handicap du fait que toutes les configurations au niveau de Snort se font par édition de fichiers textes.

4.4 Les bases de la solution proposée

Dans cette section nous explorons la possibilité de proposer une solution au problème identifié (voir section 3.11). Nous nous appuyons sur le mécanisme de sécurité proposé par J. H. Guerrero et de R. G. Cardenas [54] pour l'adapter à un firewall distribué sous *Netfilter*.

4.4.1 Le mécanisme de sécurité utilisé

Ce mécanisme a été proposé par J. H. Guerrero et de R. G. Cardenas [54]. Il s'agit en fait d'un protocole de communication entre l'IDS Snort et *iptables*. Ce protocole se base sur le format d'échange de messages de détection d'intrusions (IDMEF) proposé par le groupe de travail sur la détection d'intrusions (IDWG) pour adapter un système à réagir de façon dynamique aux variétés d'attaques qui peuvent se produire dans son environnement. Le modèle IDMEF [65] est une représentation orientée objet des données d'alertes produites par un IDS et fut implémenté avec XML en raison de sa flexibilité. Les alertes générées au format IDMEF via un plugin de sortie [66] sont analysées en temps réel par un agent, qui utilisera les informations qu'il extrait pour reconfigurer la stratégie de sécurité de l'hôte firewall, dans le but de stopper une attaque ou une intrusion. Ce mécanisme peut être illustré par la figure ci-dessous.

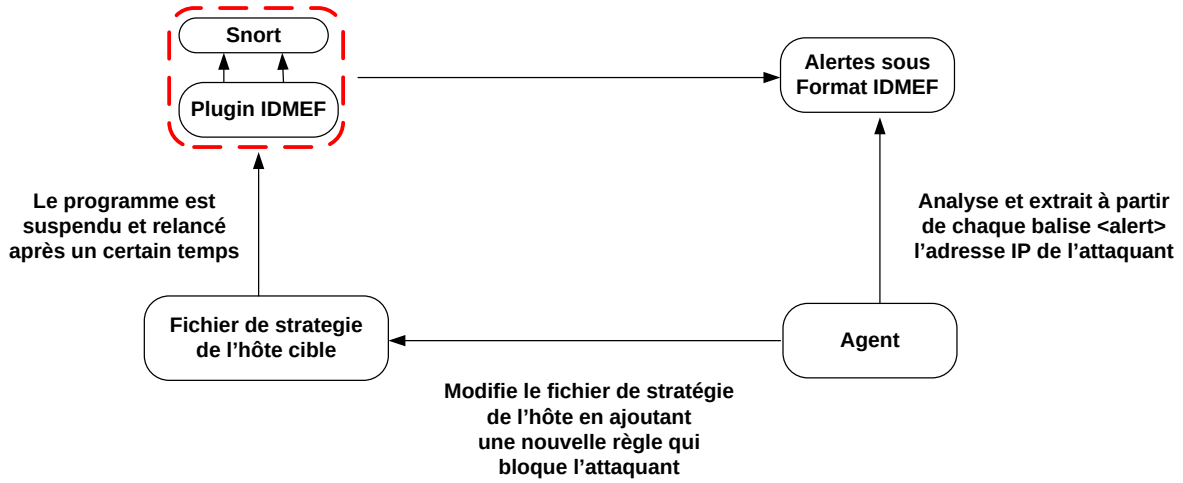

Figure 4.2 : Mécanisme de sécurité basé sur Snort et le logiciel Netfilter

Toutefois, bien que ce mécanisme présente des avantages, il présente également des inconvénients prévisibles. En effet, plusieurs programmes dont l'IDS *Snort*, le *plugin*

IDMEF, l'*Agent* tournent au niveau d'un même hôte ; cela peut consommer beaucoup de ressources au niveau de l'hôte.

En ce qui nous concerne et sur la base de ce mécanisme, notre solution utilise l'IDS Snort et un agent pour bloquer les attaques ou tentatives d'attaques qui se produisent au niveau du point final. Cela reste valable pour tous les points finaux du firewall distribué.

4.4.2 L'interaction entre les différentes entités

L'IDS Snort et le firewall logiciel *iptables* communiquent en utilisant un agent qui va analyser le fichier *alert* généré par Snort. Notons que ce fichier est stocké dans « **/var/log/snort/**».

L'*agent analyzer* a été implémenté en C et testé sous Linux (voir **annexe A**). Cet agent est également capable de reconfigurer le firewall logiciel *iptables* en ajoutant de nouvelles règles dans le fichier de configuration du firewall *iptables*.

La figure 4.3 illustre cette interaction entre les différentes entités.

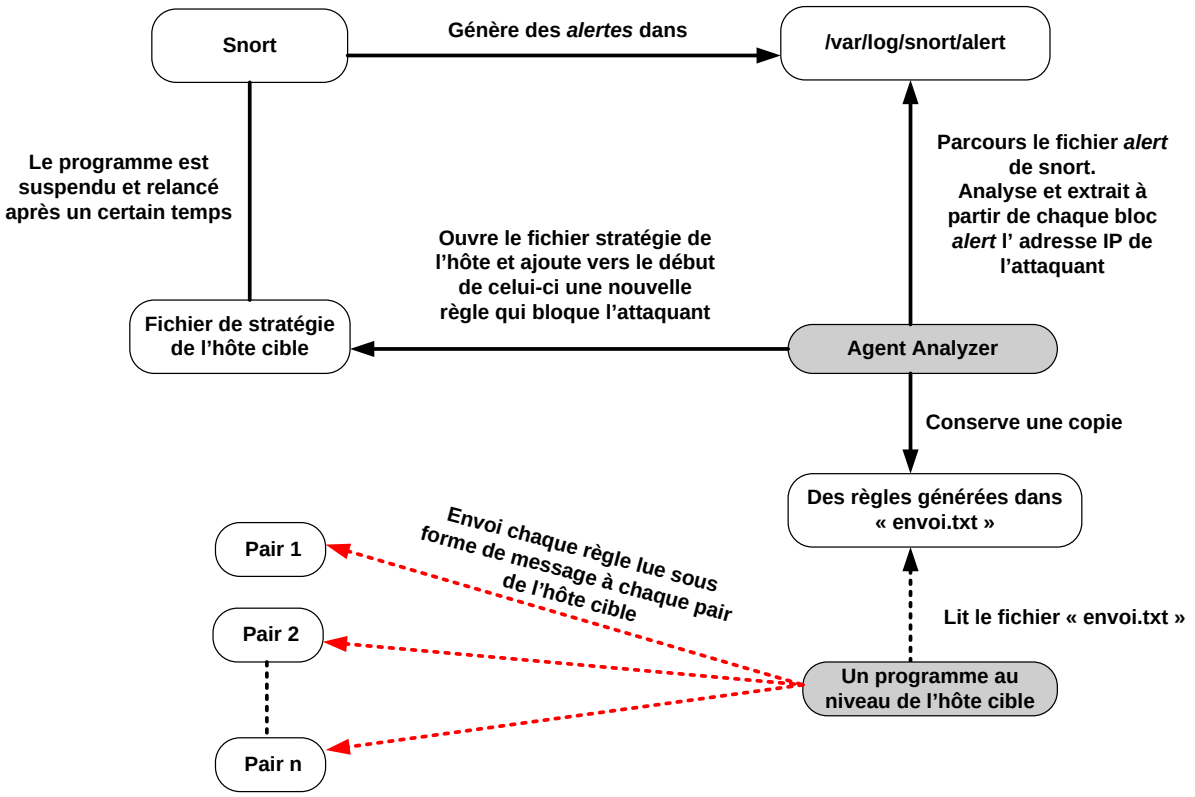

Figure 4.3 : Principe de communication de l'IDS Snort et du firewall Netfilter

L'IDS Snort capture le trafic entrant et/ou sortant du *point final* et l'examine. En cas de détection d'événements suspects, il génère des alertes sous son format log traditionnel. Ensuite il les stockent dans le fichier « **/var/log/snort/alert** ».

Les alertes générées par Snort sont examinées par l'***agent Analyzer*** évoqué précédemment. Cet *agent* parcourt le fichier « **/var/log/snort/alert** » et examine chaque bloc *alert* qu'il rencontre afin de vérifier les informations qu'il contient, car toutes les informations relatives à une attaque se trouvent à l'intérieur de ce bloc, y compris l'adresse IP de l'attaquant. Dès que cette information est trouvée, l'*agent Analyzer* la stocke en mémoire. Puis il continue la lecture du fichier « **/var/log/snort/alert** » et répète les étapes ci-dessus pour chaque bloc *alert* rencontré jusqu'à la fin du fichier.

Une fois le dernier bloc du fichier lu, l'*agent Analyzer* ouvre le fichier de stratégie du point final (voir **annexe C**) et ajoute vers le début (par défaut après la stratégie de sécurité par défaut) de celui-ci une ou plusieurs nouvelles règles pour bloquer un ou plusieurs attaquants. Les règles sont de la forme suivante :

iptables -A INPUT -s {adresse IP source} -j DROP

Après cela, l'exécution du programme *Agent* est suspendue pendant un temps prédéfini par l'administrateur réseau via le démon *crond* et est relancée à l'expiration de ce délai (voir **annexe A**). Ce délai est configurable en fonction de l'activité et des exigences du système. Ce délai fournit également un intervalle de temps, durant lequel le fichier « **/var/log/snort/alert** » est mis à jour avec de nouvelles alertes.

Toutefois, notons que l'*agent analyzer*, avant d'être suspendu, conserve une copie des règles générées dynamiquement dans un fichier local au point final (**« envoi.txt »**). Celles-ci seront ensuite envoyées à tous ses pairs sous forme de messages par un autre programme que nous présentons dans la section communication entre les points finaux (voir section 4.4.3). L'idée est de les informer sur une menace qui a été bloquée afin qu'ils puissent s'en protéger.

Le principe de communication décrit ci-dessus reste valable pour tous les *points finaux* du firewall distribué. Ce qui constitue un système de défense distribué, efficace contre les éventuelles attaques ou intrusions.

Cette solution est extensible en ce sens qu'elle peut être utilisée pour prendre en compte d'autres filtres IP autres que *iptables*, comme par exemple le filtre *ipf* des systèmes d'exploitation UNIX *OpenBSD* et *Solaris*, … etc.

4.4.3 Communication entre les points finaux

Lorsqu'un point final bloque un certain trafic supposé suspect, il en informe ses pairs du trafic bloqué en leur envoyant des messages sous forme de règles de sécurité, qu'ils doivent ajouter dans leurs fichiers de stratégies, afin de renforcer leurs mécanismes de sécurité.

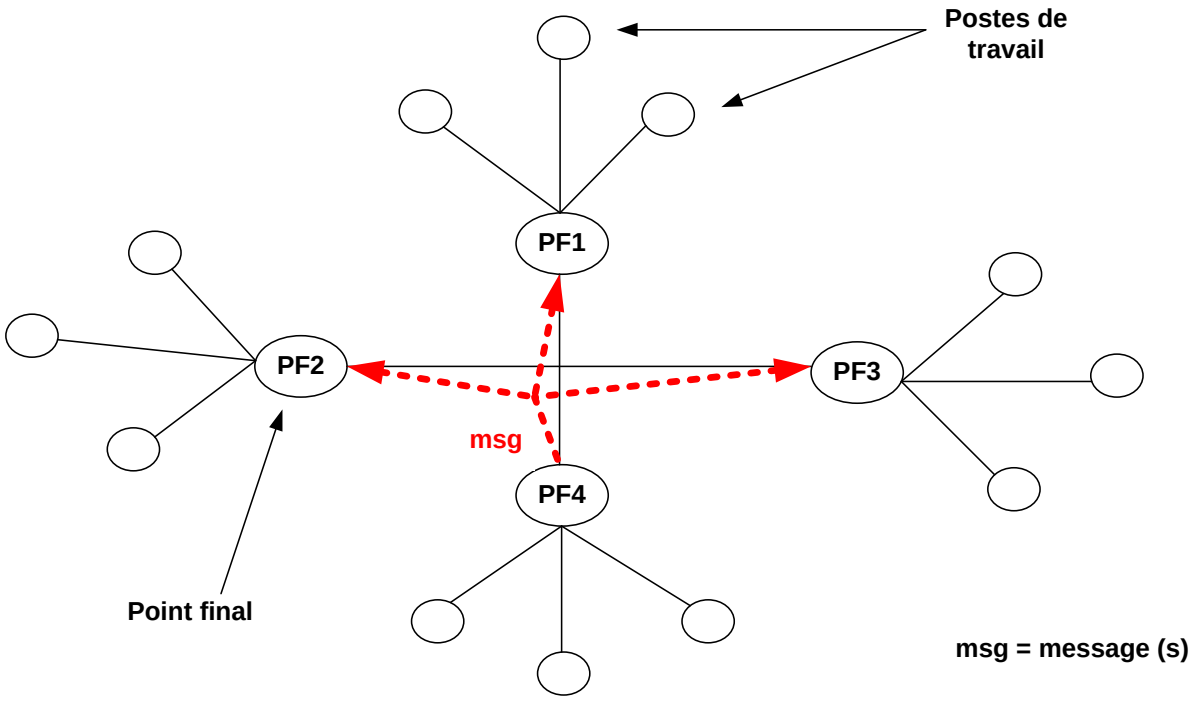

Figure 4.4 : Exemple de communication entre les points finaux

La figure 4.4 illustre un exemple de communication entre les points finaux. Le point final N°4 envoie un message de diffusion à tous ses pairs.

Nous avons simulé cette communication entre les différents points finaux du firewall distribué grâce à un programme pair à pair (P2P) en JAVA (voir **annexe D**). Celui-ci réside au niveau

de chaque point final du réseau. Ce programme fonctionne en deux modes : l'écoute et l'envoi.

Lorsqu'un point final veut envoyer un message, il parcourt le fichier « **envoi.txt** » car c'est lui qui contient les règles de sécurité générées dynamiquement par l'*agent Analyzer* concernant les activités suspectes, détectée au niveau dudit point final.

Si le fichier n'est pas vide chacune de ses lignes est envoyée à tous les pairs sous forme de message. Chaque pair, qui reçoit un message, le stocke dans un fichier en local appelé « **reception.txt** ». Une fois que l'hôte a reçu les messages provenant d'un de ses pairs, un programme lie ces derniers et les ajoutent au niveau du fichier de stratégie de l'hôte. Après les fichiers « **envoi.txt** » et « **reception.txt** » sont vidés par le programme P2P en attendant de nouvelles informations.

L'échange d'informations entre les hôtes pairs se fait via des sockets qui leurs sont associées de manière dynamique pendant qu'ils communiquent.

Grâce à ce programme P2P, Plusieurs hôtes du firewall distribué peuvent communiquer en parallèle et échanger des informations spécifiques. Ce qui nous permet d'avoir un système distribué capable de s'adapter aux différentes situations qui se produisent dans son environnement.

4.5 Algorithme de l'Agent Analyzer

Cet algorithme est constitué de deux parties : la récupération d'une ou plusieurs adresses relatives à un ou plusieurs attaquants au niveau du fichier *alert* et l'ajout de celles-ci au niveau du fichier de stratégies de l'hôte.

4.5.1 La récupération des données

Cette partie concerne la récupération des données contenues dans le fichier *alert* généré par *Snort.* Elle est constituée essentiellement de la fonction *GetIPattaquant,* qui a pour rôle principal d'extraire les données contenues dans le fichier *alert.* Il s'agit précisément d'une ou plusieurs adresses relatives à un ou plusieurs attaquants.

L'Algorithme de la fonction *GetIPattaquant ()* est décrit par la procédure suivante.

La fonction *GetIPattaquant ()* prend la variable **cheminfile** en paramètre, qui désigne le fichier *alert* y compris son emplacement, et renvoie un tableau d'informations relatives à plusieurs alertes.

Les variables utilisées par le programme sont :

- **expreg :** désigne une variable contenant une expression régulière que l'on cherche tout au long du fichier *alert*.
- **TabAdr :** désigne un tableau de caractères contenant la liste des adresses IP des attaquants.

```
GetIPattaquant (cheminfile)
DEBUT
TabAdr, tab, tab1, tab2, elts : Tableaux de chaîne de caractères
nb, j : Entiers
trouve : Booléen
nb ← 1 ;
expreg ← '/^((([0-9]{1,3}\.){3})([0-9]{1,3}\ :[0-9]{1,5})(\->)(([0-9]{1,3}\.){3})([0-9]{1,3}
        \ :[0-9]{1,5}))([ ] ?)$/' ;
Ouvrir « cheminfile » en Lecture
TANTQUE non Eof (cheminfile) FAIRE
   Ligne ← recuperer_ligne_fichier (cheminfile) ;
   SI (recherche_motif (expreg, trim (Ligne), elts))
     tab ← extraireInfo (« -> », elts [1]) ;
     tab1 ← extraireInfo («  : », tab[1]) ;
     tab2 ← extraireInfo («  : », tab[2]) ;
     j ← 1 ;
     trouve ← FAUX ;
     TANTQUE (j < Len (TabAdr) && trouve == FAUX)
        SI (TabAdr[j] == trim (tab1[1]))
           trouve ← VRAI ;
        FINSI
        j ← j + 1 ;
     FINTANTQUE
```

```
      SI (trouve == FAUX)
         TabAdr[nb] ← trim (tab1[1]) ;
         nb ← nb + 1 ;
      FINSI
   FINSI
FINTANTQUE
FermerFichier (cheminfile) ;
retourner TabAdr ;
FIN
```

La fonction *GetIPattaquant* intervient au niveau du *point final* pour parcourir le fichier *alert* généré par Snort. Ce fichier se trouve par défaut dans « **/var/log/snort/** ».
Elle le parcourt ligne après ligne (première boucle TANTQUE) et récupère chaque ligne contenue dans ce fichier. La ligne peut être vide tout comme elle peut contenir des données relatives à une alerte donnée.

Pour chaque ligne récupérée, la fonction *GetIPattaquant* recherche une expression régulière correspondant à une chaîne de caractère de la forme « **aaa.aaa.aaa.aaa :xxxxx -> bbb.bbb.bbb.bbb :yyyyy »**. Cette expression régulière commence par un motif compris entre 0 et 9 qui doit apparaître au moins 1 fois et au plus 3 fois suivit d'un point. Le sous motif obtenu doit apparaître exactement 3 fois.

Celui-ci doit être accompagné d'un motif compris entre 0 et 9, qui doit également apparaître au moins 1 fois et au plus 3 fois, suivi de « : » et d'un autre sous motif compris entre 0 et 9, qui doit apparaître au moins 1 fois et au plus 5 fois.

L'ensemble de ses sous motifs doit être suivit de « -> », d'un autre sous motif compris entre 0 et 9, qui doit apparaître au moins 1 fois et au plus 3 fois suivi d'un point. Ce sous motif doit apparaître exactement 3 fois.

Le tout doit être suivi d'un autre motif compris entre 0 et 9, qui doit apparaître au moins 1 fois et au plus 3 fois. Le motif obtenu doit être suivi de « : », ensuite d'un autre sous motif compris entre 0 et 9, qui doit apparaître au moins 1 fois et au plus 5 fois.

Le motif résultant constitué de tous les sous motifs précédents, doit être suivi d'un dernier sous motif caractérisé par un espace. L'expression régulière obtenue par la combinaison des sous motifs précédents est recherchée au niveau de chaque ligne parcourue du fichier *alert* par la fonction *GetIPattaquant.*

Lorsqu'elle est trouvée (première condition SI), on extrait l'adresse IP de l'attaquant. Ensuite, on parcourt le tableau TabAdr (deuxième et dernière boucle TANTQUE) afin de vérifier qu'elle ne s'y trouve pas. Au cas ou elle se trouve dans le tableau qui stocke les adresses IP des attaquants (deuxième condition SI), on ne l'inscrit pas dans ce tableau, sinon on l'inscrit dans ledit tableau (troisième et dernière condition SI).

Ce processus se poursuit pour chaque ligne contenant l'expression régulière recherchée jusqu'à la fin de la première boucle TANTQUE, c'est-à-dire lorsqu'on arrive à la fin du fichier « **/var/log/snort/ alert**». Ensuite celui-ci est fermé et supprimé. Il est recrée par Snort lors d'une prochaine mise à jour c'est-à-dire lorsque de nouvelles alertes sont remontées.

4.5.2 La modification du fichier de stratégies

Cette partie concerne l'ajout des adresses récupérées dans la première partie sous forme de règles de stratégie dans le fichier de configuration du point final. Elle est constituée essentiellement de la fonction *Modifier_strategie,* qui a pour rôle principal de récupérer les adresses IP des attaquants et de les ajouter vers le début (par défaut après la stratégie de sécurité par défaut) du fichier de stratégies du point final sous forme de règles permettant de bloquer un ou plusieurs attaquants.

Les variables utilisées par le programme sont :

- **regle** désigne une variable locale qui reçoit la règle de sécurité à ajouter vers le début du fichier de stratégie de l'hôte ;
- **TabAdr :** désigne une variable locale qui reçoit la liste des adresses IP des attaquants.

Modifier_strategie (fichier_strategie_hôte)
DEBUT
TabAdr ← GetIPattaquant (chemin_alert) ;
Ouvrir « **fichier_strategie_hôte** » en Ecriture
POUR i ←1 à **Len (**TabAdr**) PAS** 1
SI (FindIP (**fichier_strategie_hôte,** TabAdr[i])==FALSE)
regle ← "iptables –A INPUT –s TabAdr[i] –j DROP" ;

Garder une copie de la règle générée dans le fichier « **envoi.txt** »

Se placer vers le début du « **fichier_strategie_hôte** »

Ajouter (regle) ;

FINSI

FINPOUR

FermerFichier (**fichier_strategie_hôte**) ;

FIN

L'algorithme de *la fonction Modifier_strategie* intervient lorsque la fonction *GetIPattaquant* a récupérée une ou plusieurs adresses IP relatives à un ou plusieurs attaquants.

La première instruction permet de récupérer une ou plusieurs adresses IP relatives à un ou plusieurs attaquants via la fonction *GetIPattaquant* dans le tableau TabAdr.

La deuxième instruction ouvre le fichier de stratégie en écriture, parcourt le tableau TabAdr et pour chaque adresse rencontrée, on vérifie si elle n'a pas été déjà bloquée, si oui on ne fait rien et on passe à l'adresse IP suivante de TabAdr ; sinon la règle de sécurité permettant de bloquer l'attaquant est construite, ensuite stockée dans une variable locale. On garde d'abord une copie de la règle dans le fichier local « **envoi.txt** ».

Ensuite, on se met vers le début (par défaut après la stratégie de sécurité par défaut) du fichier de stratégies de l'hôte afin d'ajouter la règle permettant de bloquer l'attaquant.

Ce processus se poursuit jusqu'à la fin de la boucle POUR, c'est-à-dire lorsqu'il n'y a plus d'adresse IP à bloquer.

Enfin, la dernière instruction ferme le fichier de stratégie de l'hôte (c'est-à-dire le point final).

L'ordinogramme de l'agent Analyzer est présenté à la page suivante.

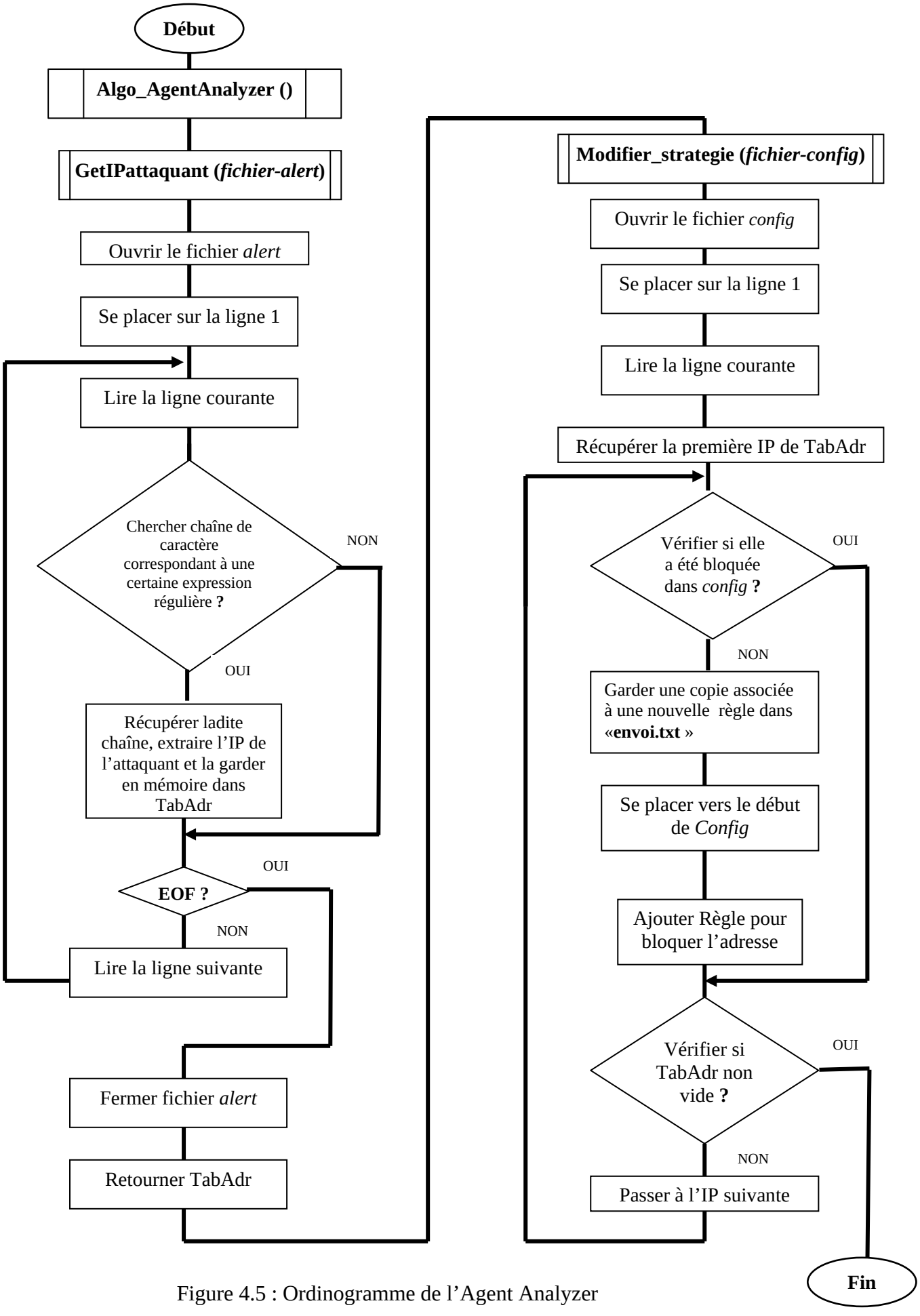

Figure 4.5 : Ordinogramme de l'Agent Analyzer

4.6 Simulation et Validation

Dans cette section, nous voulons tester et valider l'algorithme de l'*agent Analyzer* que nous avons implémenté et programmé en C sous Linux. Cet agent et l'IDS Snort sont implantés au niveau de chaque point final du firewall distribué. Ils interagissent en se partageant des connaissances et en travaillant de manière collaborative. Le système de détection d'intrusions Snort surveille le trafic entrant et/ou sortant du *point final* et génère des alertes sous un certain format en cas de détection d'événements suspects. L'agent *Analyzer* parcourt le fichier « *alert* » généré par Snort, l'analyse et extrait une ou plusieurs adresses IP relatives à un ou plusieurs attaquants présumés. Ensuite, il modifie la stratégie de sécurité du point final afin de les bloquer.

Notre objectif majeur est de simuler le comportement de l'*agent Analyzer* avec *l'IDS Snort* et le *firewall* situés au niveau du point final. En d'autres termes, nous voulons nous assurer que l'agent Analyzer est capable d'analyser effectivement les alertes remontées par Snort au niveau du fichier « **/var/log/snort/alert** », d'extraire les adresses IP des attaquants présumés et de reconfigurer le fichier de stratégie de l'hôte pour les stopper.

Notre second objectif consiste à envoyer les règles de protection générées par l'hôte cible sous forme de messages à tous ses pairs. L'idée est de les informer d'une ou plusieurs menaces qui ont été bloquées afin qu'ils puissent s'en protéger. Cette simulation a été réalisée grâce à un programme P2P que nous avons implémenté en JAVA. Celui-ci a été implanté au niveau de chaque point final du firewall distribué.

4.6.1 Les paramètres d'entrées

4.6.1.1 Agent Analyzer

L'*agent Analyzer* prend en entrée le fichier « ***alert*** » généré par Snort. Par défaut, celui-ci se trouve dans «**/var/log/ snort** ». Toutes les alertes remontées par Snort sont inscrites au niveau de ce fichier sous forme de blocs. Les différents blocs ont la même structure (voir figure 4.6).

```
[**] [1:486:5] ICMP Destination Unreachable Communication with Destination
Host is Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
10/04-11:59:50.467284 192.168.2.2 -> 192.168.2.1
ICMP TTL:64 TOS:0xC0 ID:55087 IpLen:20 DgmLen:68
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST
FILTERED
** ORIGINAL DATAGRAM DUMP:
 192.168.2.1:55652 -> 192.168.2.2:1414
TCP TTL:36 TOS:0x0 ID:38585 IpLen:20 DgmLen:40
Seq: 0xBC9BADFE
(12 more bytes of original packet)
** END OF DUMP
```

Figure 4.6 : Exemple de Bloc alerte au sein du fichier « *alert* »

Ce qui varie au niveau de chaque bloc peut être l'adresse IP de l'attaquant, le port source/destination, la taille des paquets envoyés, l'identifiant des paquets. En d'autres termes la valeur des données seule est susceptible de changer d'un bloc à un autre. Une attaque peut être constituée d'un seul bloc ou de plusieurs (cas des attaques de type *scan de port*).

Au niveau de chaque bloc alerte, l'*agent Analyzer* cherche une information du type encadré à la figure 4.6. Cette information renferme des données telles que l'adresse IP de l'attaquant, le port qu'il a utilisé, l'adresse IP de l'hôte attaqué et le port utilisé par celui-ci pour répondre à l'attaquant.

La taille du fichier « *alert* » varie en fonction de la quantité d'alertes remontées dans ce dernier. Chaque fois qu'une mise à jour survient c'est-à-dire lorsque Snort détecte de nouvelles attaques, elles son insérées dans le fichier « *alert* ». Ce qui fait que la taille de celui-ci peut devenir importante à un moment donné. Si l'administrateur réseau n'a pas pris les conditions qui s'imposent, par exemple le supprimer ou le stocker sur un hôte de sauvegarde, ce fichier est sauvegardé et renommé par Snort. Ce qui peut consommer davantage d'espace disque.

C'est pourquoi, il est recommandé à l'administrateur réseau de le supprimer de façon périodique. Lors d'une prochaine mise à jour, Snort vérifiera d'abord l'existence de ce fichier avant d'y insérer des informations sur de nouvelles alertes. S'il existe, Snort insère les nouvelles alertes remontées ; sinon il le recrée avant d'insérer de nouvelles informations [58].

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Pour faciliter le travail de l'agent *Analyzer*, nous supprimons de manière dynamique le fichier « *alert* » après consultation de celui-ci par l'agent *Analyzer*.

4.6.1.2 Le programme P2P

Il prend en entrée le fichier « envoi.txt » généré par l'*agent Analyzer* au niveau de l'hôte cible, le parcourt et envoie chaque ligne lue sous forme de message à tous ses pairs que sont les autres points finaux du firewall distribué. Les lignes du fichier «envoi.txt » se présentent sous forme de règles de sécurité du type décrit dans la section 4.4.2.

4.6.2 Mise en place d'une plateforme de test

Pour valider les programmes précités, nous avons mis en place une plateforme de test. Nous montrons les caractéristiques du matériel utilisé dans cette plateforme à savoir les machines, les types de câble utilisés, etc. Ensuite nous présentons brièvement le système d'exploitation et quelques packages installées sur les machines. Puis nous montrons l'architecture physique de notre plateforme de test, les types d'attaques que nous voulons simuler, les outils de simulation de ces attaques, et enfin les tests que nous avons effectués sur cette plateforme afin d'évaluer le comportement des programmes *Analyzer* et *P2P*.

Notre plateforme de test est constituée de trois machines Pentium IV. Les caractéristiques des trois machines sont définies dans le tableau 4.1 :

Tableau 4.1 : Caractéristiques des machines utilisées

Machines	Caractéristiques
Trois Pentium IV	RAM : 512 Processeur : 2.50 GHZ Disque Dur : 40 GO

Parmi les trois machines de la plateforme, une dispose de deux interfaces réseau, tandis que les deux autres ont une interface réseau chacune. Un Switch (commutateur) relie toutes ces machines dans un même réseau. Nous avons utilisé des câbles droits pour connecter les machines au niveau du Switch.

Comme plateforme de développement, nous avons utilisé le système d'exploitation GNU/Linux particulièrement la distribution **fedora core 6**. En effet GNU/Linux fournit un environnement open source qui nous offre l'avantage de pouvoir modifier le noyau du système d'exploitation lorsque nécessaire. Les mécanismes et librairies de sécurité (prise en charge d'IPsec, SSL, etc.) sont également pris en charge par GNU/Linux. Nous avons installé **fedora core 6** sur les trois machines. Ensuite nous avons installé et configuré le système de détection d'intrusion **Snort** au niveau de chaque hôte. Enfin, nous avons implémenté un *prototype de firewall* grâce au langage *iptables* sur chaque hôte (voir **annexe B**).

Les programmes ***agent Analyzer*** et ***pair à pair (P2P)*** que nous avons respectivement implémentés en C et en JAVA furent intégrés au niveau de chaque hôte. Le programme P2P en particulier permet aux différents hôtes de communiquer entre eux. Ce qui nous permet d'avoir un système distribué dans lequel les différents hôtes constituent des entités pairs et peuvent communiquer en s'envoyant des informations spécifiques.

4.6.2.1 Architecture de la plateforme

La plateforme est composée d'un réseau interne (192.168.1.0/24), sur lequel se trouvent les trois hôtes décrits précédemment, d'un routeur et avec Internet comme réseau externe. La structure de la plateforme est présentée à la figure 4.7.

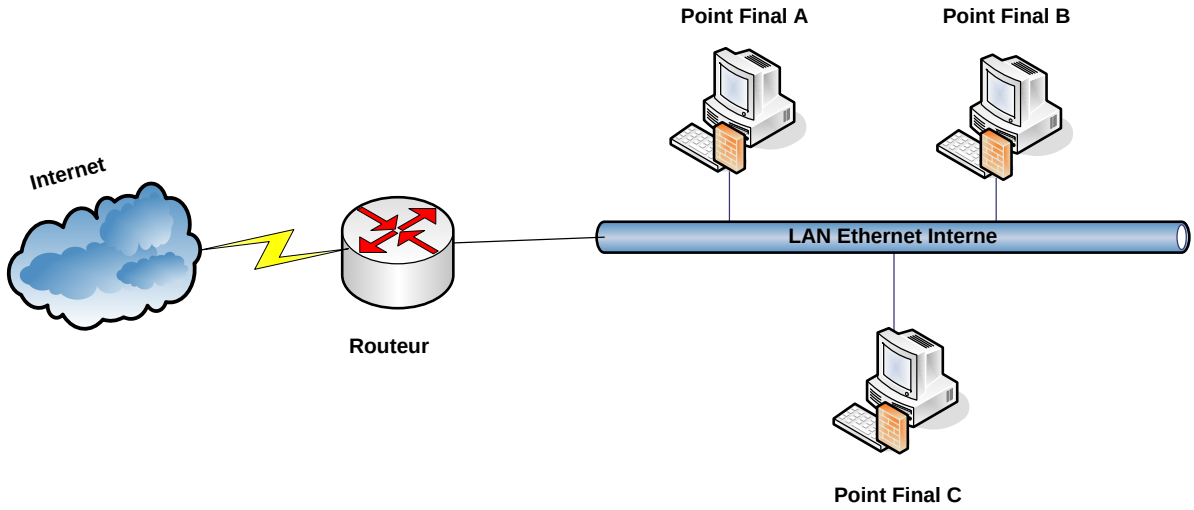

Figure 4.7 : Architecture de la plateforme de test

Les hôtes de la figure 4.7 sont des entités pairs entièrement autonomes en ce sens que chacun joue le rôle pour lequel il a été mis en place. Sur chaque hôte tourne le système de détection

d'intrusion Snort qui surveille le trafic entrant et/ou sortant de celui-ci afin de détecter toute activité suspecte. Egalement sur chaque hôte tourne un prototype de firewall qui laisse seulement passer un certain trafic supposé normal. L'IDS Snort communique avec le prototype de firewall via l'agent Analyzer décrit précédemment (voir section 4.5).

Toutefois, bien qu'étant des entités autonomes, les hôtes de la figure 4.7 peuvent communiquer entre eux de manière automatisée en s'envoyant des messages spécifiques. Cette communication se fait grâce au programme P2P (voir section 4.4.3) tournant sur chacun d'eux.

L'interaction entre les hôtes au sein de la plateforme, nous permet de simuler le comportement d'un système distribué.

4.6.2.2 Quelques attaques mises en œuvre

Il peut y avoir plusieurs types d'attaques contre les firewalls. Nous voulons en simuler principalement celles de type scan de port parce qu'elles constituent l'un des types d'attaques le plus accessible du point de vue réalisation. Dans la suite, nous simulons les attaques de types scan de port suivantes :

- Scan par détection des services en écoute sur une machine;
- Scan de protocoles IP ;
- Scan TCP connect ()

Les informations sur les caractéristiques générales de cette famille d'attaques peuvent être trouvées à la section 3.8.3.
Pour simuler ces différentes attaques, nous avons utilisé le scanner Nmap à travers sa commande *nmap*.

4.6.3 Tests sur les scans de port

Dans cette section nous décrivons les tests que nous avons effectués sur notre plate forme, ainsi que les commandes que nous avons utilisées pour lancer des attaques sur la plateforme. Afin de simuler des attaques Internet dans la réalité, nous avons utilisé un câble croisé pour relier l'interface réseau externe de la machine à double interface réseau à une machine externe qui servira de machine pirate, à partir de laquelle nous lançons les attaques.

Toutefois, nous avons remarqué que l'ensemble des attaques simulées ont été détectées par l'IDS Snort situé sur l'hôte cible, et également remontées au niveau du fichier « *alert* » de celui-ci.

Comme nous l'avons dit précédemment (voir section 4.6.2), l'objectif majeur de cette simulation est d'observer le comportement de l'*agent Analyzer* avec *l'IDS Snort* et le *firewall* situés au niveau du point final afin d'évaluer sa fiabilité.

Dans la suite, nous présentons les résultats de la simulation des attaques avant l'exécution du programme *analyzer* et ensuite les résultats après exécution de ce programme.

4.6.3.1 Résultats des scans avant l'exécution du programme Analyzer

Ici le système de détection d'intrusions Snort et le prototype de firewall sous *iptables* sont activés au niveau de l'hôte cible mais ne communiquent pas encore, car c'est l'agent *Analyzer* qui leur permet d'interagir et celui-ci n'est pas activé durant les tests suivants. Donc même si Snort détecte des activées suspectes, celles-ci seront seulement remontées au niveau du fichier « *alert* », mais le fichier de stratégie de l'hôte cible ne sera pas modifié.

4.6.3.1.1 Scan par détection des services en écoute sur une machine

Au niveau de la machine pirate la commande ***nmap 192.168.2.2***, donne les résultats de la figure 4.8.

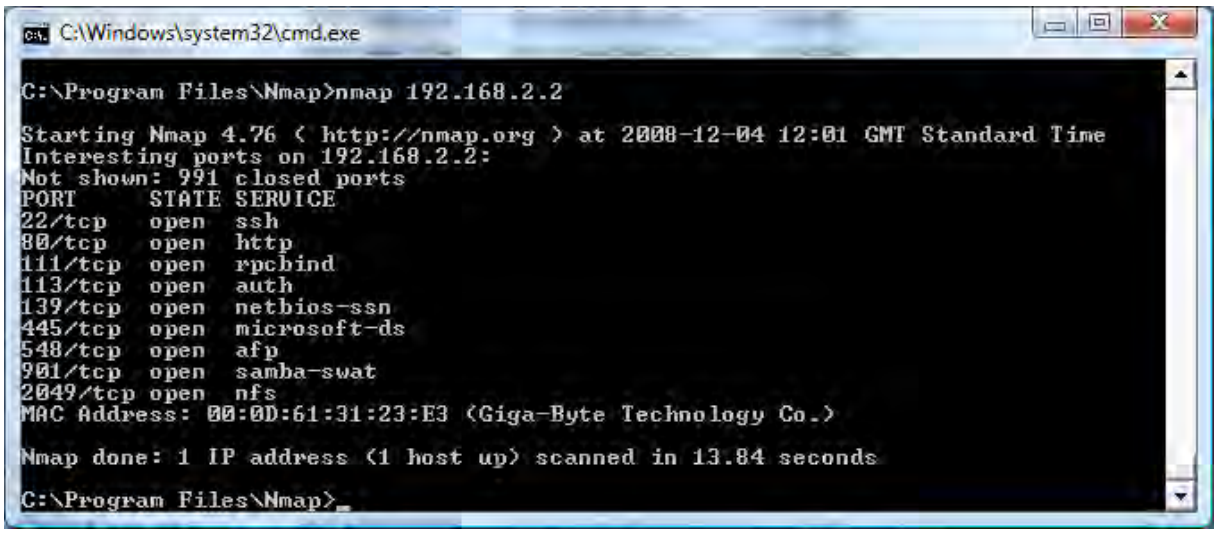

Figure 4.8 : Test Scan par détection des services en écoute sur une machine 1

Nous constatons que le pirate a obtenu des informations assez intéressantes sur quelques ports ouverts sur la machine cible. Cela signifie que le prototype de firewall implémenté au niveau de l'hôte cible a joué son rôle. L'idéal serait que l'hôte cible dispose de moyens automatisés lui permettant de priver le pirate d'informations qu'il recherche au moment ou il le désire et sans nécessiter l'intervention de l'administrateur réseau.

4.6.3.1.2 Scan de protocoles IP

Au niveau de la machine pirate la commande ***nmap -sO 192.168.2.2***, donne les résultats de la figure 4.9.

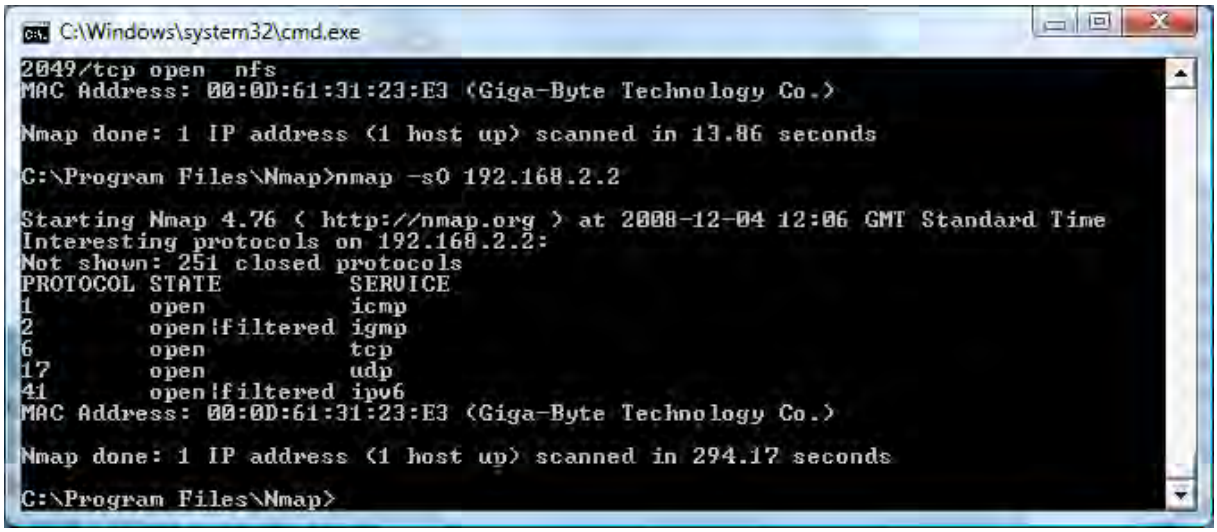

Figure 4.9 : Test Scan de protocoles IP 1

Ici, nous remarquons que le pirate a obtenu des informations sur quelques protocoles IP utilisés au niveau de l'hôte cible. Cela signifie que notre prototype de firewall a joué son rôle. Cependant par mesure de sécurité, il serait plus intéressant que le pirate ne dispose pas d'informations sur l'hôte cible. Cela l'empêcherait de préparer ses attaques futures, étant donné qu'il n'a pas eu assez d'information sur l'hôte ciblé.

4.6.3.1.3 Scan TCP connect ()

Au niveau de la machine pirate la commande ***nmap -sT 192.168.2.2***, donne les résultats de la figure 4.10.

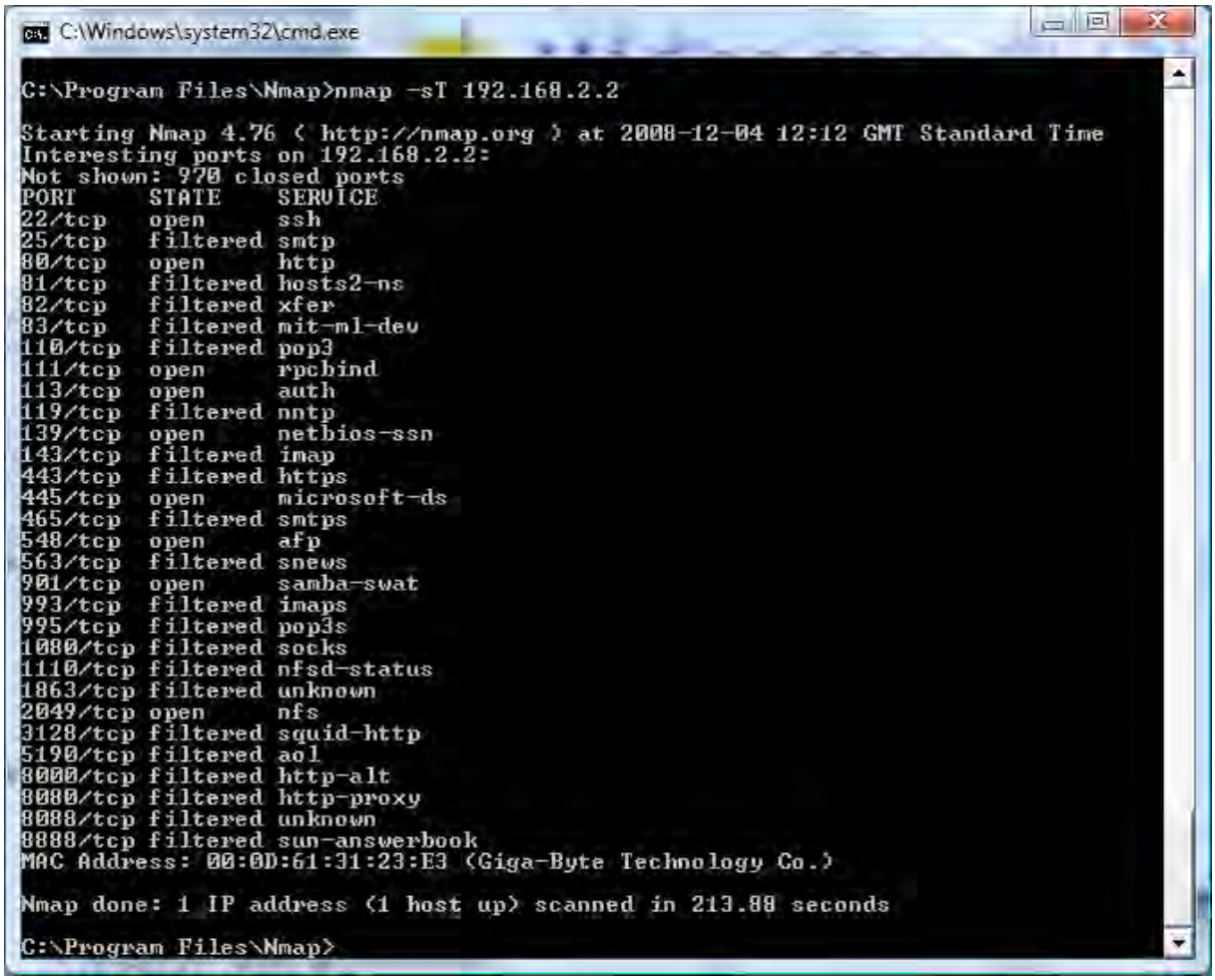

Figure 4.10 : Test Scan TCP connect () 1

Nous constatons que ce type d'attaques a procuré des informations intéressantes au pirate. Ce dernier a obtenu des informations sur les ports filtrés et les ports ouverts au niveau de la machine cible. Parmi les ports renseignés, nous remarquons que seuls quelques un sont ouverts ; la plupart étant filtrés. Ce qui veut dire que le prototype de firewall implémenté au niveau de l'hôte cible a joué son rôle. Cependant l'attaquant connaît les ports ouverts et fermés au niveau de l'hôte cible. Il serait plus intéressant qu'il ne connaisse pas les ports ouverts et/ou fermés de l'hôte cible.

4.6.3.2 Résultats des scans après l'exécution du programme Analyzer

Ici le système de détection d'intrusions Snort et le prototype de firewall sous *iptables* sont activés au niveau de l'hôte cible, et communiquent grâce à l'agent *Analyzer*. Ainsi, lorsque Snort détectera des activées suspectes, celles-ci seront non seulement remontées au niveau du fichier « *alert* », mais également le fichier de stratégie de l'hôte cible sera modifié.

4.6.3.2.1 Scan par détection des services en écoute sur une machine

Au niveau de la machine pirate la commande ***nmap 192.168.2.2***, donne les résultats de la figure 4.11.

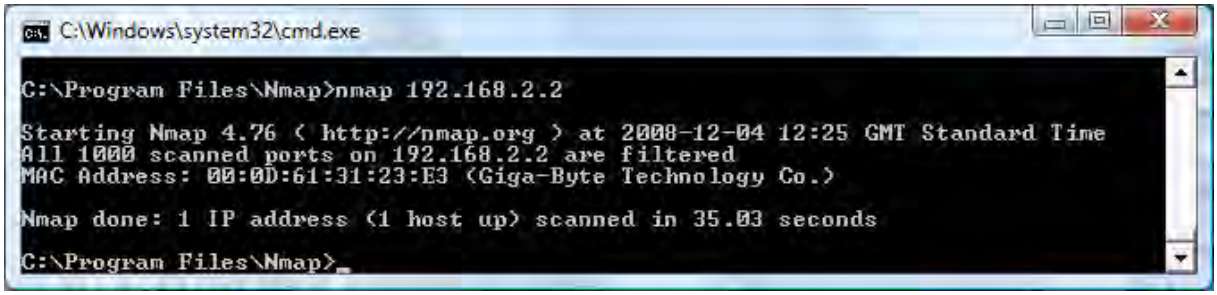

Figure 4.11 : Test Scan par détection des services en écoute sur une machine 2

Nous constatons que l'attaquant n'a pas eu d'informations sur les ports en écoute au niveau de la machine cible. Cela est dû au fait que l'hôte cible, par l'intermédiaire de l'*agent analyzer*, a pu modifier sa stratégie de sécurité pour bloquer l'adresse IP de l'attaquant (voir **annexe C**).

4.6.3.2.2 Scan de protocoles IP

Au niveau de la machine pirate la commande ***nmap -sO 192.168.2.2***, donne les résultats de la figure 4.12.

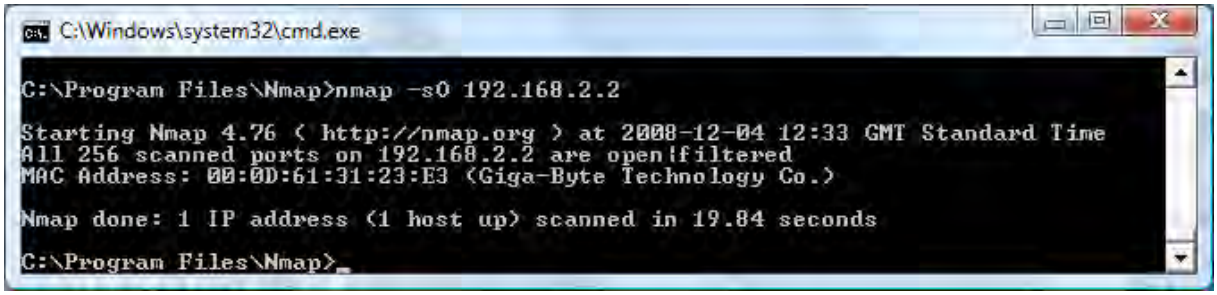

Figure 4.12 : Test Scan de protocoles IP 2

Ici, nous remarquons que l'attaquant n'a pas obtenu d'informations sur les protocoles IP utilisés au niveau de l'hôte cible contrairement au test précédent l'activation de l'*agent analyzer* (voir Figure 4.9). Ce qui veut dire que la modification du fichier de stratégie de l'hôte cible a contribué à l'obtention de ces résultats.

4.6.3.2.3 Scan TCP connect ()

Au niveau de la machine pirate la commande ***nmap -sT 192.168.2.2***, donne les résultats de la figure 4.13.

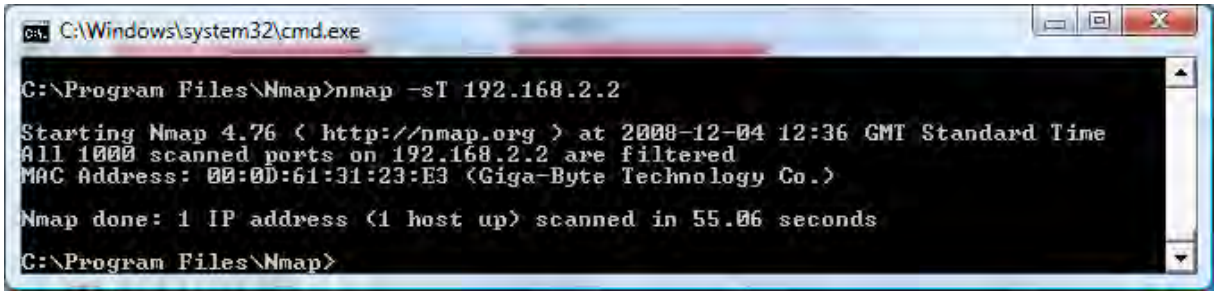

Figure 4.13 : Test Scan TCP connect () 2

A l'instar des deux tests précédents, ici nous constatons également que le pirate n'a pas obtenu d'informations sur les ports en service au niveau de l'hôte cible. Ce qui nous permet de conclure que la modification de la stratégie de sécurité a été bénéfique pour l'hôte cible, car ce mécanisme lui a permis de refuser des informations à un intrus présumé.

Ce mécanisme est général et peut s'appliquer pour n'importe quel type d'attaque. Tout dépend de la manière dont l'administrateur réseau configure l'IDS Snort au niveau d'un point final donné, puisque la configuration de l'IDS Snort peut varier en fonction de l'hôte qu'il est sensé surveiller. Par exemple la manière de configurer l'IDS au niveau d'un serveur web peut ne pas être la même au niveau d'un serveur DNS. Donc l'on configure l'IDS pour détecter des types de trafic supposés suspects pour un hôte donné.

Après la simulation de l'interaction de l'agent *Analyzer* avec l'IDS Snort et le prototype de firewall sous *iptables*, nous avons simulé le comportement du programme P2P. L'objectif est de s'assurer que les différents points finaux pouvaient communiquer entre eux en s'échangeant des informations spécifiques.

Ainsi nous avons lancé le programme P2P sur chacune des machines de la plateforme. Ce programme comprend deux classes principales : la classe *Ecoute* et la classe *Envoi*.

La classe *Ecoute* permet au point final d'être en position d'écoute de manière continue tant qu'il est sous tension. Ce qui veut dire que le point final reste à l'écoute d'éventuelles informations que ses pairs pourraient lui envoyer tant qu'il est allumé. C'est cette classe qui traite les informations reçues des autres pairs en les stockant dans le fichier « **reception.txt** » situé au niveau du point final. Ensuite, ces informations sont inscrites au niveau du fichier de stratégie du point final.

La classe *Envoi* est sollicité lorsque le point final a des informations à envoyer à ses pairs. Elle consulte régulièrement le fichier « **envoi.txt** » généré par l'agent *Analyzer* et lorsque celui-ci n'est pas vide, elle envoie chaque ligne lue sous forme de message aux pairs du point final.

L'échange d'informations entre les hôtes pairs se fait via des sockets qui leurs sont associées de manière dynamique pendant qu'ils communiquent.

Après le lancement du programme P2P, nous avons remarqué que l'hôte cible a partagé les informations contenues dans son fichier « **envoi.txt** » avec ses deux autres pairs. En effet, ces derniers ont réceptionnés et stockés les informations envoyées par l'hôte cible dans leurs fichiers « **reception.txt** » respectifs. Ensuite, ces informations ont été ajoutées au niveau des fichiers de stratégies respectifs des pairs de l'hôte cible. Ce qui nous permet d'obtenir un système de défense distribué capable de résister à différentes types d'attaques tels que les attaques distribuées de type DDOS (Distributed Denial of Service).

4.7 Conclusion

Dans ce chapitre, nous avons vu que les firewalls distribués présentent des insuffisances liées à l'adaptabilité de la stratégie de sécurité des hôtes. Nous avons proposé une solution basée sur l'utilisation d'un IDS open source moyennant un agent au niveau de chaque point final du firewall distribué afin de résoudre ce problème. Cette solution a été implémentée sur un réseau de trois machines dans lequel nous nous sommes employé à faire des simulations sur certaines attaques avec des tests comparatifs afin d'évaluer la fiabilité de la solution proposée. Les résultats des tests effectués montrent que notre solution peut bien convenir aux systèmes de firewalls distribués sous *iptables* en dotant leurs hôtes de mécanismes leurs permettant d'adapter dynamiquement leurs stratégies de sécurité aux changements qui se produisent dans

leurs environnements. La solution proposée est également extensible en ce sens qu'elle peut être adaptée à des systèmes de firewalls distribués mixtes dans lesquels les hôtes tournent avec des systèmes d'exploitation différents tels que Linux et Unix.

Conclusion Générale

D'une manière générale, la sécurité réseau est un domaine très complexe. Aussi longtemps que les topologies réseaux seront modifiées et élargies, des nouveaux outils seront nécessaires pour les sécuriser. Les firewalls distribués proposent une nouvelle approche en sécurisant chaque point final sur le réseau.

Dans ce mémoire, nous avons abordé la notion de firewall en général, en montrant à quel niveau il se situe parmi les services de sécurité de base, puis en le présentant dans son aspect traditionnel ainsi que son évolution en tant que concept distribué. A ce niveau, nous avons vu que malgré les multiples avantages qu'il offre, il présente également des inconvénients. Et parmi ces derniers, nous avons étudié le problème lié à l'adaptabilité de la stratégie de sécurité des hôtes dans un tel système. Enfin, nous avons proposé une solution à un tel problème, qui se base sur des agents pour modifier la stratégie de sécurité des hôtes afin d'obtenir un système de défense distribué efficace contre les attaques ou intrusions.

Notre solution présente un certain nombre d'avantages parmi lesquels nous avons :

- Les points finaux disposent d'un mécanisme de communication leur permettant de repousser conjointement plusieurs attaques distribuées, comme les attaques de type déni de service distribué –DDOS ;

- Le mécanisme supplémentaire de détection d'intrusion implanté dans chaque point final permet de détecter des attaques que le firewall laisserait passer. Il permet également de protéger le point final qui recevrait une stratégie de sécurité inadéquate. Enfin il permet au point final de facilement détecter les attaques internes ;

- Les points finaux peuvent se conformer à répondre de manière automatisée aux divers types d'attaques sans nécessiter l'intervention de l'administrateur de sécurité. Ce qui permet d'obtenir un système de défense distribué, efficace contre les éventuelles attaques ou intrusions.

Notons toutefois, que de nombreux problèmes restent encore non résolus dans le domaine des firewalls distribués. Le problème de conformité de la stratégie de sécurité des utilisateurs internes en constitue un exemple.

Nous nous proposons comme perspectives de recherche d'implémenter la solution proposée pour le filtre *iptables* dans un firewall distribué afin d'évaluer son efficacité en temps réel. D'autre part, cette approche est extensible puisqu'elle peut être utilisée pour prendre en compte d'autres filtres IP.

La stratégie de réponse dynamique implémentée dans ce travail est de bloquer l'adresse IP du présumé attaquant; elle est réalisée en ajoutant une règle de sécurité. Bien que cela présente des avantages du fait qu'elle bloque totalement le présumé attaquant, il serait plus intéressant de lui refuser l'accès à un ou plusieurs services donnés.

La sécurité, la scalabilité et la fiabilité sont trois principaux défis dans la conception de futurs Intranets, d'hôtes Internet et de clusters. En sécurisant par exemple des Intranets de longue portée, la surcharge dans le changement dynamique de la stratégie de sécurité pourrait s'accroître. Un cluster réseau pourrait également être encombré par le trafic avec des mises à jour de stratégies fréquentes. Ces problèmes n'ont pas encore de réponses dans notre contexte. Pour bien répondre à ces questions liées à la scalabilité, nous aurons besoin de tester notre solution dans un firewall distribué constitué d'Intranets, ou de clusters de grande taille où les hôtes sont exposés par exemple à un accès public.

Une autre perspective concerne l'étude des performances des firewalls distribués matériels. Ces derniers constituent un domaine en plein essor dans les systèmes de firewalls distribués et posent un certain nombre de problèmes.

Références

[1] Bellovin Steven M., **Distributed Firewalls**, AT&T, November 1999.
URL: http://www.cs.columbia.edu/~smb/papers/distfw.pdf

[2] Konaté Karim, Cours de **Sécurité Réseaux**. Université Cheikh Anta Diop Dakar, AEA 2005-2006.

[3] Tanenbaum Andrew S., **Computer Networks**, Fourth Edition, Prentice Hall March 17, 2003.

[4] El Kadhi Nabil, **Théorie de la Cryptographie**, pages 1-121, 28 Mars 2005.
URL: http://www.smsi.rnu.tn/html/manifes/jip05/presentations/sec-trans.pdf

[5] Pujolle Guy**, Les Réseaux**, pages 880-890, Edition 2003.

[6] Shirey R., **Internet Security Glossary**, May 2000. RFC2828.
URL: http://www.faqs.org/rfcs/rfc2828.html

[7] **SECURITY ARCHITECTURE FOR OSI FOR CCITT APPLICATIONS**. ITU-T, Recommendation X.800, Geneva 22nd of March 1991.
URL: http://fag.grm.hia.no/IKT7000/litteratur/paper/x800.pdf

[8] Cheynet Jérôme, Da Silva Miguel, Sebban Nicolas, **FIREWALL**, Exposé NT Réseaux. Janvier 2007.
URL: http://www-igm.univ-mlv.fr/~duris/NTREZO/20032004/Cheynet-DaSilva-Sebban-Presentation-Firewalls.ppt

[9] Stallings William, **Cryptography and Network Security Principles and Practices**, Fourth Edition. pages 6-22, Prentice Hall November 16, 2005.

[10] CERTA, **Filtrage et pare-feux**, N° CERTA-2006-INFO-001, Paris le 10 janvier 2006.
URL: http://www.certa.ssi.gouv.fr/site/CERTA-2006-INF-001/index.html

[11] Cheswick William R., Bellovin Steven M., Rubin Aviel D., **Firewalls and Internet Security**, Second Edition. Repelling the Wily Hacker. Addison-Wesley Professionnal Computing Series, February 2003.

[12] Désandré Amélie, Kittler Benjamin, Loutrel Romain, Renaudin Thomas. **Sécurité Informatique**. Novembre 2004.
URL: http://www.kittler.fr/docs/Securite_Reseaux.pdf

[13] Parmar S.k., **An Introduction to Security**. Information Ressources Guide, Computer, Internet and Network Systems Security, Security Manual, pages 138-139, February 2000.
URL: http://www.apnilife.com/E- Books_apnilife/Books%20in%20PDF%20Format_apnilife/Computer.Network.&.Internet.Security_apnilife.pdf

[14] Joye Philippe, **Architecture Firewalls Intranet**, Ecole d'Ingénieurs et d'Architectes de Fribourg, 2002. URL: http://cui.unige.ch/tios/cours/TechInternet/Firw1_phil.pdf

[15] Srisuresh P. and al. **Traditional IP Network Address Translator (Traditional NAT).** RFC 3022, Network Working Group, January 2001.
URL: http://www.isi.edu/in-notes/rfc3022.txt

[16] Rekhter Y. and al**. Address Allocation for Private Internets**. RFC 1918, Network working Group, February 1996. URL: http://www.faqs.org/rfcs/rfc1918.html

[17] Sohm Jean-Claude, **Serveur proxy, firewall et sécurité**. 28 Février 2002. URL: http://cerig.efpg.inpg.fr/Note/2002/proxy2.htm

[18] Jacquemin Alban, Mercier Adrien. **Les Firewalls**. 15 Février 2004. URL: http://www.frameip.com/firewall/

[19] M'Bata Amakou, Persent Olivier. **Firewall, Pare-feux, Mur de Feu**. 11 Décembre 2006. URL: http://master-info.univ-lyon1.fr/M2SIR/2006/firewall.pdf

[20] Wan Daniel, **Distributed Firewall**. GSEC Pratical Assignment version 1.2C, May 2001. URL: www.giac.org/practical/gsec/Daniel_Wan_GSEC.pdf

[21] Strand Lars, **Adaptative distributed firewall using intrusion detection**. UniK University Graduate Center University of Oslo, Kjeller, Novembre 1st, 2004. URL: http://gnist.org/~lars/studies/master/StrandLars-master.pdf

[22] Ioannidis Sotiris, Keromytis Angelos D., Bellovin Steven M. and Smith Jonathan M., **Implementing a Distributed Firewall**", ACM Conference on Computer and Communications security, Athens, Greece, Novembre 2000. URL: http://www1.cs.columbia.edu/~angelos/Papers/df.pdf

[23] Fogel Avi, **Pushing Security to Network Endpoints**. NetworkWorld, 22 January 2001. URL: http://www.nwfusion.com/news/tech/2001/0122tech.html

[24] Francis Neetha. **Distributed Firewall Technology**. 590266. CSC581. URL : http://filebox.vt.edu/users/rjoseph/csc581/

[25] Chamoun Maroun. **Thèse de doctorat : Intégration de l'internet 3G dans une plateforme Active**. Ecole Nationale Supérieure des Télécommunications de Paris, 15 Mai 2006. URL: http:// pastel.paristech.org/bib/archive/00001749/01/theseMarounChamounSoutenue 15Mai2006.pdf

[26] Paul Olivier. **Thèse de doctorat : Le contrôle d'accès dans les réseaux ATM**. Université de Rennes 1, Département Réseaux et Services Multimédia, ENST de Bretagne, 27 Février 2001. URL: http://www-lor.int-evry.fr/~paul_o/these.pdf

[27] Grant James, Attfield Philip, Armstrong Ken, **DISTRIBUTED FIREWALL TECHNOLOGY**, EWA Canada, March 2001. URL: http://www.ewa-canada.com/Papers/distributed_firewall_technology.pdf

[28] Li Wei. **Distributed Firewall.** December, 5, 2000. URL: http://citeseer.ist.psu.edu/cache/papers/cs/21533/http:zSzzSzwww.cs.helsinki.fizSzu zSzasokanzSzdistseczSzdocumentszSzli.pdf/li00distributed.pdf

[29] Kaul A., Gami D. and Khanna P., **Distributed/Dynamic Software Firewalls**. University of North Carolina, Charlotte, 2003. URL: http://www.d10systems.com/gami/articles/dyn_dist_fw.pdf

[30] Chinchani R., Atkins Derek. **A Proactive Enhancement to the Distributed Firewall**. Proceedings of the 2002 IEEE. Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 17-19 June 2002. URL: http://www.cse.buffalo.edu/~rc27/publications/ia02-distfirewall.pdf

[31] Stepanek Robert, **Distributed Firewalls.** Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory. T-110.501 Seminar on Network Security 2001 ISBN 951-22-5807-2.
URL: http://www.tml.tkk.fi/Studies/T-110.501/2001/papers/robert.stepanek.pdf

[32] Comment ça marche. **Usurpation d'adresse IP (Mystificatio/ Spoofing).**
URL : http://www.commentcamarche.net/attaques/usurpation-ip-spoofing.php3

[33] RFC 791.**Internet Protocol.** DARPA internet program protocol specification. Septembre 1981.
URL : http://intranet.ccb.ac.uk/acdept/technology/computing/resources/internet/CIE/RFC /Orig/rfc791.txt

[34] Laissus François. **Cours d'introduction à TCP/IP.** Ecole centrale Paris MS SIO Septembre 2005. URL: http://www.msio.ecp.fr/IMG/pdf/cours.pdf

[35] Viardin Alexandre. **Un petit guide pour la sécurité**. Version Revue du 24 Septembre 2008.
URL: http://www.linux-france.org/prj/inetdoc/telechargement/tutoriel.securite.pdf

[36] Olivier Jean G. et al. **Attaques Informatique**. Centro Brasileiro de Pesquisas Fisicas.
URL: http://www.rederio.br/downloads/pdf/nt00700.pdf

[37] IBM. **Mettre fin aux attaques internes : comment protéger les informations sensibles de l'entreprise.** Sécurité et protection des données sensibles, Septembre 2006.
URL: http://www-935.ibm.com/services/fr/igs/pdf/securite_attaques_internes.pdf

[38] Eraslan Aydin et al. **A Distributed Firewall Appliance**. Pro-G, Turkcell 2006
URL: http://www1.webng.com/xaydin/wp/distributedFW2006.pdf

[39] TheGreenBow, **THEGREENBOW FIREWALL DISTRIBUE**. SYSTECH SA specifications Techniques, 2002.
URL: http://www.ossir.org/windows/supports/2002/2002-07-08/The%20GreenBow%20-%20Firewall%20Distribue.PDF

[40] F-Secure, **F-Secure Distributed firewall**. User's Guide, F-Secure Corporation, 2003.
URL: http://www.f-secure.com/download-purchase/manuals/docs/manual/dfw/fra/dfw.pdf

[41] Kapoor Saket. **Distributed Firewalls.** CS-743, Assignment 1, July 2004.
URL: http://www.mcs.drexel.edu/~vp/CS743/Assign1/Kapoor_DistributedXFirewalls.pdf

[42] Blaze M., Feigenbaum J., Lacy J., **Decentralized Trust Management**. In Proc. Of the 17th Symposium on Security and Privacy, pages 164-173. EEE Computer Society Press, Los Alamitos, 1996.
URL: http://www.cs.utsa.edu/~winsboro/teaching/CS6463-S06/Papers/BFL96.pdf

[43] Blaze M., Feigenbaum J., Ioannidis J., and Keromytis A. D., **The keynote trust management system version 2**. Internet RFC 2704, September 1999.
URL: http://www.faqs.org/rfcs/rfc2704.html

[44] Markham Tom, Payne Charlie, **Security at the Network Edge: A Distributed Firewall Architecture**, Secure Computing Corporation, DARPA Information Survivability Conference and Exposition (DISCEX II'01) I-Volume 1 2001 p.0279.
URL: http://www.cs.plu.edu/courses/CompSec/arts/0279.pdf

[45] Payne C. and Markham T., **Architecture and Applications for a distributed embedded firewall**. In Proceedings of the 17th Annual Computer Security Applications Conference, December 2001, pages 73-80.
URL: http://www.acsac.org/2001/papers/73.pdf

[46] Meredith L. M., **A Summary of the autonomic distributed firewalls (ADF) project**. In Proceedings of the Third DARPA Information Survivability Conference and Exposition, vol. 2, April 2003, pages 260-265.
URL: http:// www.mnlab.cs.depaul.edu/~mtaibah/papers//Others/A%20summary%20of%20the %20Autonomic%20Distributed%20Firewalls%20(ADF)%20project.pdf

[47] Kelliher Kyle. **3Com Distributed Embedded Firewall**. A Review. GSEC Assignment, Version 1.4b Option1, April 14, 2004.
URL: http://www.sans.org/reading_room/whitepapers/firewalls/

[48] Ihde Michael. **Autonomic Distributed Firewalls**. ECE442 Presentation, February 29, 2004.
URL: http://courses.ece.uiuc.edu/ece542/inCLassPresSpring2004/ ihde_Autonomic%20 Distributed%20Firewalls.ppt

[49] Markham T., Meredith L. and Payne C. **Distributed embedded firewalls with private virtual groups**. In proceedings of the Third DARPA Information Survivability Conference and Exposition, vol. 2, April 2003, pages 81-83.
URL: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/8503/26876/01194927.pdf?temp=x

[50] Université de Liège. Les Virus informatiques. Février 2004.
URL: http://www.ulg.ac.be/segi/internet/virus/index.html

[51] Fontaine Sébastien. **Attaques DOS (déni de service)**. Version 1.0 du 13 Novembre 2006.
URL : http://www.authsecu.com/dos-attaque-deny-of-service/dos-attaque-deny-of-service.php

[52] Ihde Michael and Senders William H., **Barbarians in the Gate: An Experimental Validation of NIC-based Distributed Firewall Performance and Flood Tolerance**. Department of Electrical and Computing Engineering, University of Illinois at Urbana-Champaign Urbana, IL 61801, USA 2005.
URL : http://www.perform.csl.uiuc.edu/Papers/USAN_papers/05IHD03.pdf

[53] Ihde Michael A.. **Experimental Evaluations of Embedded Distributed Firewalls: Performance and Policy**. B. S., University of Illinois at Urbana-Champaign Urbana, USA 2005.
URL: http://www.perform.csl.uiuc.edu/Papers/USAN_papers/05IHD02.pdf

[54] Guerrero J. H., Cardenas R. G., **An example of communication between security tools: Iptables-Snort**. ITESM-CEM, Operating Systems Review, Vol. 39, n° 3, pages 34-43, Jully 2005. URL: http://cat.inist.fr/?aModele=afficheN&cpsidt=16905090

[55] Zhang Q., Janakiraman R., **Indra: A Distributed Approach to Network Intrusion Detection and Prevention**. Washington University in St. Louis, Missouri-61330-4899,

USA October 13, 2001. URL: http://www.cs.wustl.edu/cs/techreports/2001/wucs-01-30.pdf

[56] Barika F. A. M.. Mémoire de DEA : **Vers un IDS Intelligent à base d'agents mobiles**. Université de TUNIS, LERIA-EPITECH-PARIS, 2003.
URL: http://leria.epitech.net/site/tree/publications/DEA.pdf

[57] Robbins Royce. **Distributed Intrusion Detections systems : An Introduction and Review**. GSEC Practical Assignment, version 1.4b, Option 1, January 2, 2002. URL: http://www.sans.org/reading_room/whitepapers/detection/897.php

[58] Caswell, Roesch Brian Marty. **Snort: The Open Source Network Intrusion Detection System.** November 2003. URL: http://www.snort.org/docs/

[59] Winteregg J., **SIMS- Security Intrusion Management System**. Ecole d'ingénieur du Canton de Vaud, Paris Décembre 2004. URL : http://www.fullsecurity.ch/fileadmin/documents/sims/sims-webJwinteregg.pdf

[60] Brennan Michael P., **Using Snort For a Distributed Intrusion Detection System**. SANS Institute 2002. URL : http://www.sans.org/reading_room/whitepapers/detection/352.php

[61] Romdhani Mohamed. **Projet : Mise en place d'une sonde**. Université de la Manouba, Tunisie 2005. URL : http://www.snort.org/docs/setup_guides/intl/french/Premiers%20pas%20avec%20Snort.pdf

[62] Adouko Georges et Al. **Panorama des algorithmes efficaces et architectures matérielles pour le filtrage réseau haut debit et la détection d'intrusions**. In : Manifestation des Jeunes Chercheurs STIC (MajecStic), Lorient, Novembre 2006. URL : http://tony.ramard.googlepages.com/article2-2.pdf

[63] Dagom Nathalie. **Détection et Prévention d'intrusion : Présentation et limites**. Laboratoire Lorrain de recherche en Informatique et ses Applications, Université de Nancy, Juillet 2006. URL : http://hal.archives-ouvertes.fr/docs/00/08/42/02/PDF/RR_DetectionIntrusion.pdf

[64] Piotrowski Michal. **Système IPS basé sur Snort**. Hakin9 N° 1/2006. URL : https://www.securinfos.info/whitepapers-securite-informatique/snortips.pdf

[65] Debar H. et al. **The Intrusion Detection Message Exchange Format (IDMEF)**. Network Working Group, RFC 4765, Mars 2007. URL: http://www.ietf.org/rfc/rfc4765.txt

[66] McAlerney, Joe. **IDMEF XML plug in for the Snort IDS**. January 2003 URL: http://www.silicondefense.com/idwg /snort-idmef/

Annexes

Annexe A : Code en C de l'agent Analyzer & le script Shell de son exécution

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <regex.h>
#define TRUE 1
#define FALSE 0
// Expression régulière pour le recherche des adresses IP
//(On recherche tout ce qui est de la forme aaa.aaa.aaa.aaa:xxxxx -> bbb.bbb.bbb.bbb:yyyyy
//et pouvant se terminer éventuellement par des caractères d'espacement)
#define REGEXP_IP "^((([0-9]{1,3}.){3})([0-9]{1,3}:[0-9]{1,5})( -> )(([0-9]{1,3}.){3})([0-9]{1,3}:[0-9]{1,5}))([:space:]*)"
// Séparateur entre les adresses : aaa.aaa.aaa.aaa:xxxxx < SEP_ADR_ADR > bbb.bbb.bbb.bbb:yyyyy
#define SEP_ADR_ADR "->"
// Séparateur entre adresse et port aaa.aaa.aaa.aaa < SEP_ADR_PORT > xxxxx
#define SEP_ADR_PORT ":"
//A paramétrer
// Chemin du fichier source
#define PATH_FILE_SRC "./toto.txt"
// Chemin du fichier d'envoi
#define PATH_FILE_ENV "./strategie_env.txt"
// Chemin du fichier temporaire
#define PATH_FILE_TMP "./strategie_tmp.txt"
// Chemin du fichier de stratégie
#define PATH_FILE_DEST "./strategie.txt"
// Régle à écrire dans le fichier de stratégie
// Rmq : %s sera remplacé par l'@ IP et \n permet un
// retour à la ligne après chaque écriture
#define REGLE_STRATEGIE "iptables -A INPUT -s %s -j DROP\n"
// Taille maximale d'une ligne lue dans le fichier source
// (A adapter selon le contenu du fichier source !)
#define SIZE_LIGNE 255
```

```
// Entete du fichier stratégie
#define ENTETE_DEST "#/bin/bash\n\n#strategie par défaut\niptables -P INPUT DROP\n
iptables -P OUTPUT DROP\niptables -P FORWARD DROP\n\n "
// Numéro de ligne dans le fichier de stratégie à partir de
// laquelle débute la copie (les lignes précédentes sont ignorées)
#define NUM_START 8
// Prototypes des fonctions utilitaires
void FindExpReg (const char * str_request, const char * str_regex, char ** str_out);
char ** str_split (const char *cs, const char *delim, size_t *p_size);
int FindIP (char * file, char * ipAdr);
int main ()
{// Ligne lue dans le fichier source
char ligne[SIZE_LIGNE];
// Pointeurs fichier source / d'envoi / temporaire / destination
FILE * file_source = NULL, * file_env = NULL, * file_tmp = NULL, * file_dest = NULL;
// Compteurs
int i, j;
//Nombre d'@ IP trouvées
int nbIP = 0;
// Tableau contenant les @ IP distinctes
char ** TabAdr = NULL;
//Variable permettant de vérifier l'existence d'une @ IP
//dans le tableau
int trouve;
// Variables utilisées dans le découpage d'une ligne du fichier source
size_t size;
char ** tab1, ** tab2;
// Tentative d'ouverture du fichier source en mode lecture seule
if ((file_source = fopen (PATH_FILE_SRC,"r")) == NULL)
{printf ("[ERROR][Echec de l'ouverture du fichier source {%s} !]\n", PATH_FILE_SRC);
exit (0);
}  else
{// Lecture du fichier source ligne par ligne
while (fgets(ligne, SIZE_LIGNE, file_source) != NULL)
{//Recherche du motif (voir plus haut)
 char * str_out = NULL;
 FindExpReg (ligne, REGEXP_IP, &str_out);
```

```
//Motif trouvé
if (str_out != NULL)
{// Extraction de l'@ IP de l'attaquant (celle de gauche)
tab1 = str_split (str_out, SEP_ADR_ADR, &size);
tab2 = str_split (tab1[0], SEP_ADR_PORT, &size);
// Mémorisation de l'@ IP s'il elle n'existe pas déjà
j=0;
trouve=FALSE;
while (j<nbIP && trouve!=TRUE)
{if (strcmp (TabAdr[j], tab2[0]) == 0)
trouve = 1;
 j++;
}
if (trouve==FALSE)
{TabAdr = (char **) realloc (TabAdr, sizeof (char*)*(nbIP+1));
TabAdr[nbIP] = (char*) malloc(sizeof(char)*(strlen(tab2[0])+1));
strcpy (TabAdr[nbIP], tab2[0]);
nbIP++;
}}}
//Fermeture du fichier source
fclose (file_source);
// Ecriture des adresses distinctes collectées dans le fichier de stratégie
for (i=0; i<nbIP; i++)
{   printf ("[INFO][Traitement de l'@ IP <%s>]\n", TabAdr[i]);
// Vérification de l'existence de l'@ IP dans le fichier d'envoi avant écriture
if (FindIP (PATH_FILE_ENV, TabAdr[i])==FALSE)
{  // Tentative d'ouverture du fichier d'envoi en mode ajout
if ((file_env = fopen (PATH_FILE_ENV,"a+")) == NULL)
{printf ("[ERROR][Echec de l'ouverture du fichier d'envoi {%s} !]\n", PATH_FILE_ENV);
exit (0);
} else
{printf ("[INFO][Adresse écrite dans le fichier d'envoi !]\n");
// Ajout de la nouvelle règle
fprintf (file_env, REGLE_STRATEGIE, TabAdr[i]);
// Fermeture du fichier d'envoi
fclose (file_env);
}
```

```
// Ecriture de l'@IP et du contenu du fichier de stratégie dans le fichier temporaire
if ((file_tmp = fopen (PATH_FILE_TMP,"a+")) == NULL)
{printf ("[ERROR][Echec de l'ouverture du fichier temporaire {%s} !]\n", PATH_FILE_TMP);
 exit (0);
}else
{// Ecriture de l'entête dans le fichier temporaire
fprintf (file_tmp, ENTETE_DEST);
printf ("[INFO][Adresse écrite dans le fichier temporaire !]\n");
// Ajout de la nouvelle règle
fprintf (file_tmp, REGLE_STRATEGIE, TabAdr[i]);
// Tentative d'ouverture du fichier destination (fichier de stratégie) en mode lecture
if ((file_dest = fopen (PATH_FILE_DEST,"r")) == NULL)
{printf ("[ERROR][Echec de l'ouverture du fichier de stratégie {%s} en mode lecture !]\n",
PATH_FILE_DEST);
 exit (0);
}
else
{printf ("[INFO][Début copie DEST -- TMP ... ");
  j=1;
// Lecture du fichier ligne par ligne
while (fgets(ligne, SIZE_LIGNE, file_dest) != NULL)
{if (j>=NUM_START)
  {// Ecriture de la chaine lue
    fprintf (file_tmp, ligne);
   }
  j++;
}
printf ("TERMINE !]\n");
// Fermeture du fichier destination
fclose (file_dest);
}
// Fermeture du fichier temporaire
fclose (file_tmp);
}
// Copie fichier temporaire --> fichier destination
// 1. Ouverture du fichier temporaire en mode lecture
if ((file_tmp = fopen (PATH_FILE_TMP,"r")) == NULL)
```

```
{printf ("[ERROR][Echec de l'ouverture du fichier temporaire {%s} !]\n", PATH_FILE_TMP);
 exit (0);
 }else
{// 2. Suppression puis ouverture du fichier destination en mode écriture
if (remove (PATH_FILE_DEST) == -1)
{printf ("[ERROR][Echec de la suppression du fichier de destination {%s} !]\n", PATH_FILE_DEST);
exit (0);}
if ((file_dest = fopen (PATH_FILE_DEST,"w+")) == NULL)
{printf ("[ERROR][Echec de l'ouverture du fichier destination en mode écriture {%s} !]\n", PATH_FILE_DEST);
exit (0);
} else
{// 3. Copie des données
 printf ("[INFO][Début copie TMP -- DEST ... ");
 while (fgets(ligne, SIZE_LIGNE, file_tmp) != NULL)
 {// Ecriture de la chaine lue
  fprintf (file_dest, ligne);
 }
 printf ("TERMINE !]\n");
 // Fermeture du fichier destination
 fclose (file_dest);
 }
 // Fermeture du fichier temporaire
 fclose (file_tmp);
// Suppression du fichier temporaire
if (remove (PATH_FILE_TMP) == -1)
{printf ("[ERROR][Echec de la suppression du fichier temporaire {%s} !]\n", PATH_FILE_TMP);
 exit (0);
}}}else
        printf ("[INFO][Adresse trouvée. Adresse non écrite !]\n");
  }}
  return 1;
}
// Fonction qui recherche une chaine (str_out) coorespondant à un motif (str_regex)
// dans une autre (str_request)
void FindExpReg (const char * str_request, const char * str_regex, char ** str_out)
```

```
{ int err;
regex_t preg;
*str_out = NULL;
err = regcomp (&preg, str_regex, REG_EXTENDED);
if (err == 0)
{  int match;
size_t nmatch = 0;
regmatch_t *pmatch = NULL;
nmatch = preg.re_nsub;
pmatch = malloc (sizeof (*pmatch) * nmatch);
if (pmatch)
{ match = regexec (&preg, str_request, nmatch, pmatch, 0);
regfree (&preg);
if (match == 0)
{ char * tmp = NULL;
int start = pmatch[0].rm_so;
int end = pmatch[0].rm_eo;
 size_t size = end - start;
tmp = malloc (sizeof (char*) * (size + 1));
*str_out = malloc (sizeof (char*) * (size + 1));
if (tmp)
{  strncpy (tmp, &str_request[start], size);
tmp[size] = '\0';
strcpy (*str_out, tmp);
free (tmp);
}}
else if (match == REG_NOMATCH)
{ //printf ("Pas de concordance !\n");}
else
{ char *text;
size_t size;
size = regerror (err, &preg, NULL, 0);
text = malloc (sizeof (*text) * size);
if (text)
{ regerror (err, &preg, text, size);
fprintf (stderr, "%s\n", text);
free (text);
```

```
}else
{fprintf (stderr, "Memoire insuffisante\n");
  //exit (EXIT_FAILURE);
}} } else
{ fprintf (stderr, "Memoire insuffisante\n");
//exit (EXIT_FAILURE);
}}}
// Fonction qui découpe une chaine (cs) suivant un délimiteur (delim)
// et retourne l'ensemble des chaines dans un tableau (ret) et mettant à
// jour la taille (p_size)
char ** str_split (const char *cs, const char *delim, size_t *p_size)
{ size_t size = 0;
 char **ret = NULL;
 if (cs != NULL && delim != NULL)
{ int start = 0;
 int end = 0;
 size_t length = 0;
const char *t = NULL;
void *tmp = NULL;
while ((t = strstr (&cs[start], delim)))
end = t - cs;
length = end - start;
if (length > 0)
{ size++;
tmp = realloc (ret, sizeof (*ret) * size);
if (tmp != NULL)
{ret = tmp;
ret[size-1] = malloc (sizeof (**ret) * (length + 1));
if (ret[size-1])
{strncpy (ret[size-1], &cs[start], length);
 ret[size-1][length] = '\0';
}else{fprintf (stderr, "Memoire insufisante\n");
exit (EXIT_FAILURE);
}}else
{ fprintf (stderr, "Memoire insufisante\n");
exit (EXIT_FAILURE);}}
start = end + 1;
```

```
}
size += 2;
tmp = realloc (ret, sizeof (*ret) * size);
if (tmp != NULL)
{ret = tmp;
ret[size-1] = NULL;
length = strlen (cs) - start;
if (length > 0)
{ret[size-2] = malloc (sizeof (**ret) * (length + 1));
if (ret[size-2])
{strcpy (ret[size-2], &cs[start]);}
} else
{ ret[size-2] = NULL;
}
} else{ fprintf (stderr, "Memoire insufisante\n");
exit (EXIT_FAILURE); } }
if (p_size)
{*p_size = size-1; }
  return ret;
}
// Recherche de l'adresse IP (ipAdr) dans le fichier (file)
int FindIP (char * pathfile, char * ipAdr)
{FILE * file = NULL;
char ligne[SIZE_LIGNE];
char tmp[SIZE_LIGNE];
int trouve = FALSE;
if ((pathfile != NULL) && (ipAdr != NULL))
{if ((file = fopen (pathfile, "r")) == NULL)
{printf ("Echec de l'ouverture du fichier {%s} !\n", pathfile);
exit (0);
}else{// Lecture du fichier ligne par ligne
while ((fgets(ligne, SIZE_LIGNE, file) != NULL) && trouve==FALSE)
{ // On crée une chaine contenant la règle avec l'@ IP
sprintf (tmp, REGLE_STRATEGIE, ipAdr);
// Et on la compare avec la ligne courante lue dans le fichier
if (strcmp (ligne, tmp) == 0)
trouve = TRUE;}
```

```
// Fermeture du fichier
fclose (file);
}}
return trouve;}
```

#Script shell d'exécution de l'agent Analyzer

#A paramétrer selon les exigences du système

#Editer le fichier /etc/crontab est inséré la ligne suivante : Pour faciliter le test, je l'ai
#paramétré pour que le script s'exécute toutes les 15 minutes
*/15 * * * * /home/test/agent.sh

#Contenu du script Shell *agent.sh* :

```
#!/bin/sh
PATH_FILE_DEST ='/var/log/snort/alert'
# La commande suivante génère le fichier  agent.o
gcc -o  agent.o  -c agent.c
# La commande suivante génère le fichier  agent.exe
gcc -o agent.exe  agent.o
#La commande suivante exécute le fichier agent.exe
./agent.exe
#Suppression du fichier alert après consultation de celui-ci par l'agent Analyzer
remove (PATH_FILE_DEST)
#Redémarrer le script de firewall
/etc/firewall-stop
/etc/firewall-start
```

Annexe B : Prototype de firewall implémenté sous *iptables* dans chaque hôte

```
#! /bin/sh

#Stratégie par défaut-Tout refusé
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#Accepter les connexions HTTP
iptables -A INPUT -p tcp --sport www -m state --state NEW, ESTABLISHED, RELATED -j
ACCEPT

#Accepter les connexions SSH
iptables -A INPUT -p tcp --sport ssh -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT

#Accepter les connexions FTP
iptables -A INPUT -p tcp --sport 21 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A OUTPUT -p tcp --dport 20 -m state --state ESTABLISHED -j ACCEPT

#Accepter la sortie de certains protocoles
iptables -A OUTPUT -p TCP --dport http -j ACCEPT
iptables -A OUTPUT -p TCP --dport ssh -j ACCEPT
iptables -A OUTPUT -p TCP --dport smtp -j ACCEPT
iptables -A OUTPUT -p TCP --dport ftp-data -j ACCEPT
iptables -A OUTPUT -p TCP --dport ftp -j ACCEPT

#Autoriser les autres connexions établies
iptables -A FORWARD -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A FORWARD -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A OUTPUT -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A OUTPUT -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A INPUT -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A INPUT -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT


#Supprmier les mauvais paquets
iptables -N LOG_DROP
iptables -A LOG_DROP -j LOG --log-level 1 --log-prefix '[PAQUET REJETTE]:'
iptables -A LOG_DROP -j DROP
iptables -A FORWARD -j LOG_DROP
iptables -A INPUT -j LOG_DROP
iptables -A OUTPUT -j LOG_DROP
```

Annexe C : Exemple de règle générée par le programme *analyzer* au niveau de l'hôte

```
#! /bin/sh

#Stratégie par défaut-Tout refusé
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

iptables -A INPUT -s  192.168.2.1 –j DROP } Nouvelle règle ajoutée dynamiquement

#Accepter les connexions HTTP
iptables -A INPUT -p tcp --sport www -m state --state NEW, ESTABLISHED, RELATED -j
ACCEPT

#Accepter les connexions SSH
iptables -A INPUT -p tcp --sport ssh -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT

#Accepter les connexions FTP
iptables -A INPUT -p tcp --sport 21 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A OUTPUT -p tcp --dport 20 -m state --state ESTABLISHED -j ACCEPT

#Accepter la sortie de certains protocoles
iptables -A OUTPUT -p TCP --dport http -j ACCEPT
iptables -A OUTPUT -p TCP --dport ssh -j ACCEPT
iptables -A OUTPUT -p TCP --dport smtp -j ACCEPT
iptables -A OUTPUT -p TCP --dport ftp-data -j ACCEPT
iptables -A OUTPUT -p TCP --dport ftp -j ACCEPT

#Autoriser les autres connexions établies
iptables -A FORWARD -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A FORWARD -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A OUTPUT -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A OUTPUT -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A INPUT -p tcp -m state --state ESTABLISHED, RELATED -j  ACCEPT
iptables -A INPUT -p udp -m state --state ESTABLISHED, RELATED -j  ACCEPT

#Supprmier les mauvais paquets
iptables -N LOG_DROP
iptables -A LOG_DROP -j LOG --log-level 1 --log-prefix '[PAQUET REJETTE]:'
iptables -A LOG_DROP -j DROP
iptables -A FORWARD -j LOG_DROP
iptables -A INPUT -j LOG_DROP
iptables -A OUTPUT -j LOG_DROP
```

Annexe D : Code JAVA du programme P2P

```
####################Classe Ecoute#########################
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
//Classe Principale
public class Ecoute extends Thread {
//Déclarations de variables
static final int port = 33333;
PrintWriter out = null;
BufferedReader in = null;
Socket soc;
/*Cree un socket d'écoute pour chaque client qui se connecte et on initialise les deux flux d'I/O*/
public Ecoute (Socket soc)
{ try
{this.soc = soc;
 this.in = new BufferedReader (new InputStreamReader (this.soc.getInputStream()));
 this.out = new PrintWriter(this.soc.getOutputStream(), true);
 this.start();
}catch(Exception e)
{System.out.println("Pb lors de l'écoute");
  e.printStackTrace();
} }
/*Méthode qui recupère ce qu'une pair a envoyé*/
public String reception()
{String d = null;
 try { d = in.readLine(); }
 catch(Exception er)
 {System.err.println("Erreur de lecture de la requête Pair .....");
 er.printStackTrace(); }
 return  d;
 }
```

```
/*Traitement requête Pair*/
public void TraitementRequetePair()
{ try
{ //Créer un fichier qui stockera les données reçues des hôtes
 PrintWriter bw =new PrintWriter (new FileWriter (new File ("reception.txt"),true));
 //lecture du message
 String str = this.reception();//pour message=in.readLine();
 if (str !=null)
{//Ajout du contenu dans reception.txt
  bw.println(str +"\r\n") ; // \r\n marque la fin d'une ligne
  System.out.println("ECHO = " + str);   // trace locale
  soc.close() ;//fermer la socket à la fin du traitement
}
//fermer le filtre
bw.close();
}catch (Exception e)
{ System.err.println("Erreur d'écriture dans le fichier.....");
}
}//Fin méthode requete Pair
/*Méthode run*/
public void run()
{try
{//Traitement requête pair
 TraitementRequetePair();  }
 catch(Exception e) { }
 }
/*METHODE PRINCIPALE*/
public static void main(String[] args)
{try
{//Ouverture du port au niveau de l'hôte
ServerSocket socketEcoute = new ServerSocket(port);
System.out.println("Lancement du programme Ecoute");
while (true)
{ //accepter chaque Pair qui se connecte
```

```
Socket socketPair = socketEcoute.accept();
//appelle du constructeur
new Ecoute(socketPair);
}}
catch(Exception e)
{e.printStackTrace();
}}}
####################Classe Envoi#########################
import java.io.*;
import java.net.*;
public class Envoi extends Thread {
//Déclaration de variables
static final int port = 33333;
Socket soc;
//BufferedReader entree=null;
//BufferedReader sin =null;
PrintWriter out =null;
public Envoi(){}
/*Traitement pour l'envoi de la requête Pair*/
public void TraitementEnvoiPair()
{//Variables locales
String ip;
String ligne_lue;
try
{ //Fichier contenant les données à envoyer
BufferedReader buffer = new BufferedReader(new FileReader ("envoi.txt"));
while ((ligne_lue = buffer.readLine())!=null)
{//Obtenir la liste des hôtes pairs (à partir d'un fichier local "hosts.list")
BufferedReader hosts = new BufferedReader(new FileReader("hosts.list"));
while(!(ip = hosts.readLine()).equals(""))
{//Associer une socket pour chaque hôte pair du réseau
soc = new Socket(ip, port);
//Définition du out qui permet d'envoyer
out = new PrintWriter(soc.getOutputStream(), true);
```

```
//Envoi du message à tous les pairs
out.println(ligne_lue);
out.flush();
out.close();
soc.close();
}
//Fermer l'objet hosts
hosts.close();
}//fin deuxième boucle wile
buffer.close( ) ; // on ferme le filtre
}
catch (Exception e)
{System.err.println("Echec lors de la lecture du fichier host");
e.printStackTrace();
}
}//Fin méthode envoi client
public void run ()
{try {   TraitementEnvoiPair();
}
catch (Exception e){
System.err.println("ERROR D'ENVOIE " );e.printStackTrace() ;
}
}
//Décalaration de la méthode principale
public static void main(String[] args) {
new Envoi().start();
}
}
```