REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ SAAD DAHLEB DE BLIDA

Faculté des Sciences de l'ingénieur
Département d'électronique

Projet de fin d'étude pour l'obtention du diplôme de
Master en Réseaux et Télécommunications

Étude et conception d'un Firewall

Réalisé par :

BELALIA Mohamed cherif
MAACHE Khaled

Promoteur :

Mr.MEHDI Merouane

Année Universitaire 2010 – 2011

Dédicaces

Je dédie ce travail a ma mère et mon père qui mon poussés à être parmi les meilleurs et qui m'ont encouragé et soutenue tous le long de ma vie ;

Ma petite sœur ;

Mes frères ;

Mes amis ;

Et a toutes ma famille.

Khaled

Dédicaces

Je dédie ce travail a toute ma famille qui m'a été toujours la garante d'une existence paisible et d'un avenir radieux ;

A tous mes amis ;

A tous ceux qui nous ont aidés de près ou de loin à la réalisation de ce travail.

Cherif

Remerciements

C'est avec un grand plaisir que nous réservons cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui nous ont aidés de près ou de loin à la réalisation de ce travail.

On remercie particulièrement notre promoteur« Mr. MEHDI Merouane » pour sa précieuse assistance, sa disponibilité et l'intérêt qu'il a manifesté pour ce travail.

On remercie vivement tous les enseignants qui ont participé à notre formation, plus particulièrement ceux de l'équipe réseaux et télécommunications.

On remercie également les membres du jury pour l'honneur qu'ils nous font en jugeant ce modeste travail.

Résumé

Durant la dernière décennie, le réseau informatique mondial Internet a connu une croissance exponentielle. En effet, il permet d'échanger de très grandes quantités d'informations dans des délais extrêmement courts, ce qui permet, notamment, d'augmenter la productivité des entreprises. Cependant, le raccordement de ces dernières au réseau mondial impose toute une politique de sécurité au niveau de l'entreprise qui s'y connecte, et des protections matérielles et logicielles suffisantes pour éviter tout risque de fuites ou piratage.

Ce projet a pour objectif d'étudier d'abord les concepts de la sécurité informatique en générale et en particulier les pare-feux, ensuite la conception et la mise en œuvre d'un pare-feu basé sur le filtrage de paquets, le scan de ports, le blocage de ports et le blocage d'adresses IP.

Mots clés : *sécurité réseau, pare-feu, filtrage de paquets, blocage de ports, blocage d'adresses IP.*

Abstract

During the last decade, the Internet knew an exponential increase. In fact, it allows exchanging huge quantity of information within extremely small periods, witch permits increasing the productivity of the companies. However the connection of these last to the network imposes a whole security policy on the company level witch is connected to, and sufficient materiel and software protections to avoid any risk of hackings.

The aims of this project are studying the concepts of the computer security in general and the firewalls in particular, then the design and the implementation of a firewall based on packet filtering, port scan, blocking ports, blocking of IP addresses.

Key words: *computer security, firewall, packet filtering, port scan, blocking ports, blocking of IP addresses.*

SOMMAIRE

Chapitre 2 : Firewall et sécurité réseaux

Chapitre 3 : Conception et mise en Œuvre

Chapitre 4 : Tests par simulation d'attaque

Liste des illustrations

Introduction Générale :

De nos jours, toutes les entreprises possédant un réseau local possèdent aussi un accès à Internet, afin d'accéder à la manne d'information disponible sur le réseau des réseaux, et de pouvoir communiquer avec l'extérieur. Cette ouverture vers l'extérieur est indispensable et dangereuse en même temps. Ouvrir l'entreprise vers le monde signifie aussi laisser place ouverte aux étrangers pour essayer de pénétrer le réseau local de l'entreprise, et y accomplir des actions douteuses, parfois gratuites, de destruction, vol d'informations confidentielles,.... Les mobiles sont nombreux et dangereux.

Pour parer à ces attaques, une architecture sécurisée est nécessaire. Pour cela, le cœur d'une telle architecture est basé sur un firewall. Cet outil a pour but de sécuriser au maximum le réseau local de l'entreprise, de détecter les tentatives d'intrusion et d'y parer au mieux possible. Cela représente une sécurité supplémentaire rendant le réseau ouvert sur Internet beaucoup plus sûr. De plus, il peut permettre de restreindre l'accès interne vers l'extérieur. En effet, des employés peuvent s'adonner à des activités que l'entreprise ne cautionne pas. En plaçant un firewall limitant ou interdisant l'accès à ces services, l'entreprise peut donc avoir un contrôle sur les activités se déroulant dans son enceinte.

Le firewall propose donc un véritable contrôle sur le trafic réseau de l'entreprise. Il permet d'analyser, de sécuriser et de gérer le trafic réseau, et ainsi d'utiliser le réseau de la façon pour laquelle il a été prévu et sans l'encombrer avec des activités inutiles, et d'empêcher une personne sans autorisation d'accéder à ce réseau de données.

C’est dans ce cadre que s’inscrit notre projet intitulé « Etude et Conception d'un Firewall », réalisé au sein du centre des systèmes et réseaux d’information et de communication, de télé-enseignement et d’enseignement à distance (ex. Centre de Calcul) au niveau de l’université de BLIDA, en vue de l'obtention d’un diplôme de Master en Réseaux et Télécommunications.

Notre travail consiste à étudier et concevoir un firewall implémenté sous la plate-forme Windows, avec une interface graphique facilitant sa gestion et sa configuration.

Le premier chapitre de ce mémoire présente les concepts de bases de la sécurité des réseaux informatiques. Il présente aussi les types, les motivations et les méthodes des attaques informatiques, ainsi que différents moyens permettant de contrer ces attaques.

Le second chapitre traite le concept du firewall dans la sécurité informatique, son apport, ses limites, et les types des firewalls.

Le troisième chapitre concerne la modélisation conceptuelle de notre travail, la réalisation de notre application, les outils utilisés et la description de l'interface.

Pour finir, dans le dernier chapitre, on effectuera une série de test pour valider l'application développée, en utilisant un cheval de Troie (**BO2K** « **B**ac**k** **O**rifice **2**000 »).

Chapitre 1
Sécurité des réseaux informatique

1. Introduction :

La sécurité (ou l'insécurité) informatique est tellement d'actualité qu'un rapport sur le sujet n'a pratiquement pas besoin d'introduction. En cherchant un peu dans les causes de l'inflation de ce sujet dans moins de dix ans on se rend compte que la base réside dans un mot clé : réseau. En fait, lorsqu'on dispose d'un ordinateur personnel sans connexion réseau, a-t-on besoin d'une sécurité informatique ? Normalement, la maison est assez sûre pour protéger la machine. Il n'en va plus tout à fait de même lorsque les ordinateurs sont reliés les uns aux autres via Internet. L'interconnexion des ordinateurs permet de tisser la toile du Web, et celle-ci repose d'abord sur la confiance mutuelle. Le problème est que cette confiance est régulièrement trahie. Pour comprendre la situation présente, il faut revenir un peu en arrière et examiner les faits. Il apparaît alors que le fond du problème actuel tient au fait que les gens se sont emparés de la technologie informatique en se focalisant sur ses fonctionnalités mais en négligeant les questions de sécurité.

2. La sécurité informatique :

On parle d'un système informatique sécurisé, s'il est capable d'empêcher quiconque (qu'il fasse partie du réseau ou non) d'agir sur le réseau avec des droits et privilèges que l'administrateur ne lui a pas donné. La sécurité informatique, d'une manière générale, consiste à assurer une utilisation des ressources matérielles ou logicielles d'une organisation.

La sécurité informatique consiste généralement en quatre principaux objectifs :

- L'intégrité, c'est-à-dire garantir que les données sont bien celles qu'on croit être.
- La confidentialité, consistant à assurer que seules les personnes autorisées aient accès aux ressources.
- La disponibilité, permettant de maintenir le bon fonctionnement du système informatique.
- La non répudiation, permettant de garantir qu'une transaction ne peut être niée.

Afin de pouvoir sécuriser un système, il est nécessaire d'identifier les menaces potentielles, et donc de connaître et de prévoir la façon de procéder de l'ennemi [1]. Le but de cette recherche est ainsi de donner un aperçu des motivations éventuelles des pirates, de catégoriser ces derniers, et enfin de donner une idée de leur façon de procéder afin de mieux comprendre comment il est possible de limiter les risques d'intrusions .

3. Les attaques :

3.1. Définition d'une attaque :

Une « attaque » est l'exploitation d'une faille d'un système informatique (système d'exploitation, logiciel ou bien même de l'utilisateur) à des fins non connus par l'exploitant du système et généralement préjudiciables [2].

3.2. Les « hackers » :

Le terme de hacker est souvent utilisé pour désigner un pirate informatique. Les victimes de piratage sur des réseaux informatiques aiment à penser qu'ils ont été attaqués par des pirates chevronnés ayant soigneusement étudié leur système et ayant développé des outils spécifiquement pour créer une faille dans leur système.

Le terme hacker a eu plus d'une signification depuis son apparition à la fin des années 50. Ce nom désignait convenablement les programmeurs avertis, qui pour la plupart sont devenus les fondateurs des plus grandes entreprises informatiques.

C'est au cours des années 80 que ce mot a été utilisé pour catégoriser les personnes impliquées dans le piratage de jeux vidéo, en désamorçant les protections de ces derniers, puis en en revendant des copies. Aujourd'hui ce mot est souvent utilisé pour désigner les personnes s'introduisant dans les systèmes informatiques [3].

3.3. Les différents types de pirates :

En réalité il existe de nombreux types d'attaquants catégorisés selon leur expérience et selon leurs motivations [3]:

- Les **white hat hackers**, hacker au sens noble du terme, dont le but est d'aider à l'amélioration des systèmes et technologies informatiques, sont généralement à l'origine des principaux protocoles et outils informatiques que nous utilisons aujourd'hui. Le courrier électronique en est un exemple.

- Les **blacks hat hackers (crackers)**, plus couramment appelés pirates, c'est-à-dire des personnes s'introduisant dans les systèmes informatiques dans un but nuisible.

    - ✓ Les Script Kiddies (gamins du script, parfois également surnommés crashers, lamers ou encore packet monkeys, soit les singes des paquets réseau) sont de jeunes utilisateurs du réseau utilisant des programmes trouvés sur Internet,

généralement de façon maladroite, pour vandaliser des systèmes informatiques afin de s'amuser.

- ✓ Les phreakers sont des pirates s'intéressant au réseau téléphonique commuté (RTC) afin de les utiliser gratuitement grâce à des circuits électroniques (qualifiées de box, comme la blue box, la violet box, ...) connectés à la ligne téléphonique dans le but d'en falsifier le fonctionnement.
- ✓ Les carders s'attaquent principalement aux systèmes de cartes bancaires pour en comprendre le fonctionnement et en exploiter les failles.
- ✓ Les crackers sont des personnes dont le but est de créer des outils logiciels permettant d'attaquer des systèmes informatiques ou de casser les protections contre la copie des logiciels payants.

- ❖ Les hacktivistes (contraction de hackers et activistes que l'on peut traduire en cybermilitant ou cyber-résistant), sont des hackers dont la motivation est principalement idéologique. Ce terme a été largement porté par la presse, aimant à véhiculer l'idée d'une communauté parallèle (qualifiée généralement d'underground), par analogie aux populations souterraines des films de science-fiction.

3.4. Motivations des attaques :

Comprendre les raisons pouvant motiver une attaque fournit parfois les indicateurs sur les points de vulnérabilité d'un réseau et les actions qu'un intrus pourrait entreprendre. Parmi les motivations les plus courantes, on retrouve les suivantes [4] :

Cupidité : L'intrus est payé par quelqu'un pour s'introduire sur un réseau d'entreprise afin d'y dérober ou endommager des informations relatives à l'échanges d'importantes somme d'argent.

Curiosité : L'intrus est calé en informatique et curieux, en tente d'obtenir un accès aux sites qui lui semblent intéressants.

Notoriété : L'intrus est très calé en informatique et tente de s'introduire sur des sites connus pour être difficiles à forcer pour prouver ses compétences. Une attaque réussie peut alors lui valoir le respect et la reconnaissance de ses paires.

Vengeance : L'intrus a été licencié, rétrogradé ou traité de façon déloyale par un employeur et entreprend une attaque dans le but de détruire des informations sensibles.

Ignorance : L'intrus s'intéresse à l'informatique et aux réseaux et tombe par inadvertance sur une vulnérabilité, causant involontairement des préjudices en détruisant des données ou en réalisant une action illégale.

3.5. Les types d'attaques :

Les hackers utilisent plusieurs techniques d'attaques. Ces attaques peuvent être regroupées en deux familles différentes [4] :

3.5.1. Les attaques directes :

C'est la plus simple des attaques. Le hacker attaque directement sa victime à partir de son ordinateur.

La plupart des pirates débutants utilisent cette technique. En effet les programmes de hack qu'ils utilisent ne sont que faiblement paramétrables, et un grand nombre de ces logiciels envoient directement les paquets à la victime.

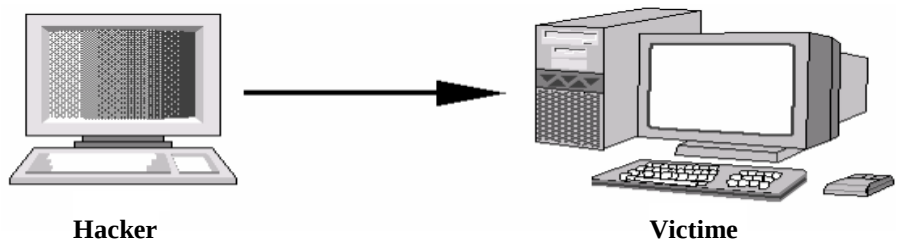

Figure 1 : Attaque directe

3.5.2. Les attaques indirectes :

Par rebond : Le principe en lui-même, est simple : les paquets d'attaque sont envoyés à l'ordinateur intermédiaire, qui répercute l'attaque vers la victime. D'où le terme « rebond ».

Avantage :

- Masquer l'identité (l'adresse IP) du hacker ;
- Eventuellement, utiliser les ressources de l'ordinateur intermédiaire pour attaquer.

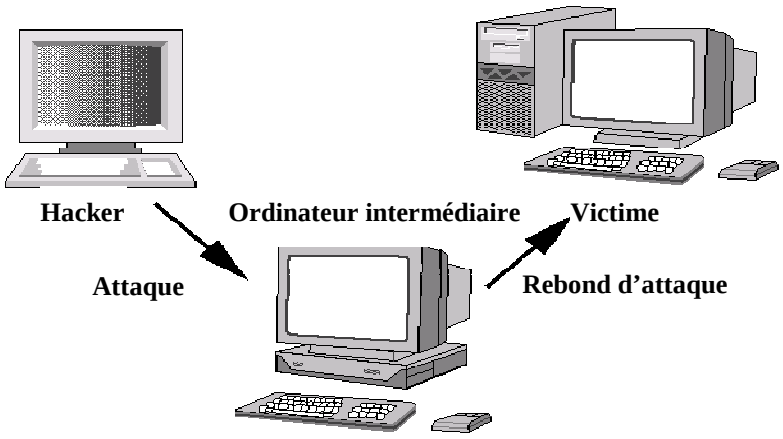

Figure 2 : Attaque indirecte par rebond

Par réponse : Cette attaque est un dérivé de l'attaque par rebond. Elle offre les mêmes avantages, du point de vue du hacker. Mais au lieu d'envoyer une attaque à l'ordinateur intermédiaire pour qu'il la répercute, l'attaquant va lui envoyer une requête. Et c'est cette réponse à la requête qui va être envoyée à l'ordinateur victime.

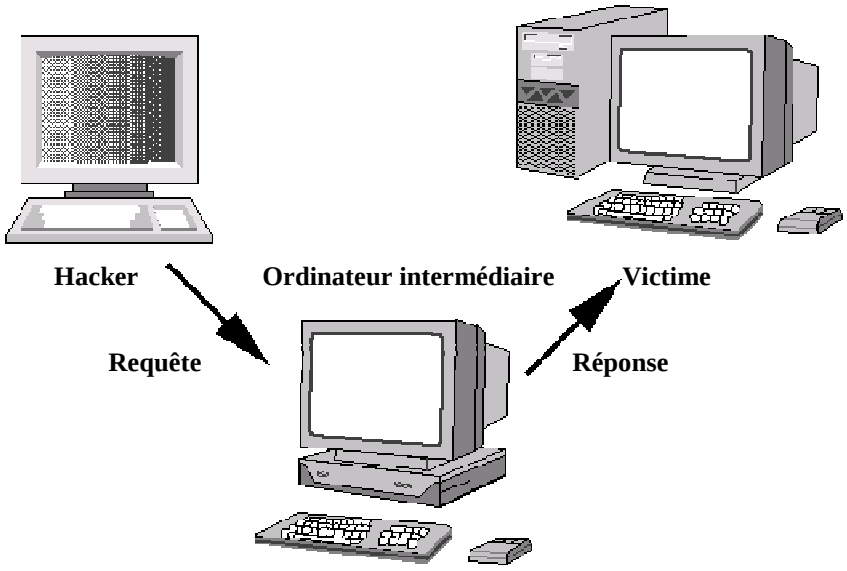

Figure 3 : Attaque indirecte par réponse

3.6. Les malwares :

Un malware est un logiciel développé dans le but de nuire à un système informatique. Il existe plusieurs familles de malwares. On va définir les plus intéressants [5]:

3.6.1. Virus :

C'est un programme malveillant introduit à l'insu des utilisateurs dans un système, il possède la capacité de se dupliquer (s'auto reproduire).

3.6.2. Cheval de Troie :

C'est un programme ou un code malveillant intégré à une application par ajout ou par modification de son code. Lors de l'exécution de ce programme inoffensif, le bout de code malveillant pourra exécuter des commandes spécifiques (récupération de fichiers de mot de passe, altération de système, etc.) à l'insu de l'utilisateur.

3.6.3. Ver :

Un ver informatique est un programme qui peut s'auto reproduire et se déplacer à travers un réseau en utilisant ses mécanismes, sans avoir réellement besoin d'un support physique ou logique (disque dur, programme hôte, fichier, etc.) pour se propager.

3.6.4. Spyware :

Les spywares ou Espiogiciels permettent de connaître l'activité exécutée sur l'ordinateur infecté. On y retrouve : les Keyloggers et les Tempests.

Key logger : Un keylogger (littéralement enregistreur de touches) est un dispositif chargé d'enregistrer les frappes de touches du clavier, à l'insu de l'utilisateur.

Tempest : C'est un dispositif électronique permettant de capter les émissions électromagnétiques générées par un appareil électrique.

3.6.5. Rootkit :

Les rootkits sont une variété de malwares apparue d'abord dans le monde Unix/Linux puis, plus récemment, dans celui de windows.

Les rootkits sont des malwares qui peuvent être très difficiles à démasquer et parfois, à éradiquer. En effet ils possèdent deux caractéristiques originales :

- D'une part ils modifient en profondeur le fonctionnement du système d'exploitation (éventuellement son noyau) ;

- D’autre part ils se rendent invisibles à des systèmes d’exploitation.

Ils peuvent même rendre invisibles autres divers parasites qu’ils auront installés : spaware, cheval de Troie…et c’est ce qui constitue en général la raison majeure de leurs existence.

3.6.6. Les bombes logiques :

Les bombes logiques sont les virus dont le déclenchement s’effectue à un moment déterminé en exploitant la date du système, le lancement d’une commande, ou n’importe quel appel au système.

3.7. Méthodologie d’une intrusion sur un réseau :

Les hackers ayant l'intention de s'introduire dans les systèmes informatiques recherchent dans un premier temps des failles, c'est-à-dire des vulnérabilités nuisibles à la sécurité du système, dans les protocoles, les systèmes d’exploitation, les applications ou même le personnel d'une organisation ! Les termes de vulnérabilité, « trou de sécurité », sont également utilisés pour désigner les failles de sécurité.

Pour pouvoir mettre en œuvre un exploit (il s'agit du terme technique signifiant exploiter une vulnérabilité), la première étape du hacker consiste à récupérer le maximum d'informations sur l'architecture du réseau et sur les systèmes d’exploitation et applications fonctionnant sur celui-ci. La plupart des attaques sont l’œuvre de « script kiddies » essayant bêtement des exploits trouvés sur Internet, sans aucune connaissance du système, ni des risques liés à leur acte.

Une fois que le hacker a établi une cartographie du système, il est en mesure de mettre en application des exploits relatifs aux versions des applications qu'il a collectées. Un premier accès à une machine lui permettra d'étendre son action afin de récupérer d'autres informations, et éventuellement d'étendre ses privilèges sur la machine.

Lorsqu'un accès administrateur (le terme anglais root est généralement utilisé) est obtenu, on parle alors de compromission de la machine, car les fichiers systèmes sont susceptibles d'avoir été modifiés. Le hacker possède alors le plus haut niveau de droit sur la machine.

La dernière étape du hacker consiste à effacer ses traces, afin d'éviter tout soupçon de la part de l'administrateur du réseau compromis et de telle manière à pouvoir garder le plus longtemps possible le contrôle des machines compromises [3].

3.8. Quelques attaques connues :

3.8.1 Attaque de mot de passe :

Attaque par force brute : On appelle **« attaque par force brute** » le cassage d'un mot de passe en testant tous les cas possibles. Il existe un grand nombre d'outils, pour chaque système d'exploitation, permettant de réaliser se genre d'opération.

Attaque par dictionnaire : Les outils d'attaque par force brute peuvent demander des heures, voire des jours, de calcul même avec des machines équipées de processeurs puissants. Ainsi, une alternative consiste à effectuer une « **attaque par dictionnaire** ». Le programme utilise une liste de mots prédéfinis dans un fichier externe. Cette liste est appelée dictionnaire ; ses mots sont la plupart du temps ceux provenant d'un dictionnaire contenant les mots du langage courant. Le programme les encrypte avec l'algorithme d'encryptage adéquat un par un et les compare au mot de passe encrypté.

Attaque hybride : Le dernier type d'attaques de ce type, appelé « **Attaque hybride** », vise particulièrement les mots de passe constitués d'un mot traditionnel et suivi d'une lettre ou d'un chiffre (tel que « khaled25 »). Il s'agit d'une combinaison des attaques précédentes.

3.8.2. Attaque par déni de service :

Une « **attaque par déni de service** » (en anglais « Denial of Service », abrégé en DoS) est un type d'attaque visant à rendre indisponible pendant un temps indéterminé les services ou les ressources d'une organisation.

Ping of Death : Le principe du « **Ping of Death** » consiste tout simplement à créer un datagramme IP dont la taille totale excède la taille maximum autorisée (65536 octets). Un tel paquet envoyé à un système possédant une pile TCP/IP vulnérable, provoquera un plantage. Aucun système récent n'est vulnérable à ce type d'attaque.

Mail Bombing : Le « **Mail Bombing** » consiste à envoyer un nombre faramineux d'emails (plusieurs milliers par exemple) à un ou des destinataires. L'objectif étant de :

- Saturer le serveur de mails,
- Saturer la bande passante du serveur et du ou des destinataires,
- Rendre impossible aux destinataires de continuer à utiliser l'adresse électronique.

Attaque Tear Drop : Le principe de cette attaque consiste à insérer dans des paquets fragmentés des informations de décalage erronées. Ainsi, lors du réassemblage il existe des vides ou des recoupements (Overlapping), pouvant provoquer une instabilité du système. A ce jour, les systèmes récents ne sont plus vulnérables à ce type d'attaque.

Attaque Land : L'attaque « **Land** » consiste à envoyer un paquet possédant la même adresse IP et le même numéro de port dans les champs source et destination des paquets IP. Les systèmes récents ne sont plus vulnérables à ce type d'attaque.

Attaque Distributed denial of service (DDOS) : Le « **DDOS** » ou déni de service distribué est un type d'attaque très évolué visant à faire planter ou à rendre muette une machine en la submergeant de trafic inutile. Plusieurs machines à la fois sont à l'origine de cette attaque (c'est une attaque distribuée) qui vise à saturer des serveurs, des sous réseaux,…..etc. d'autre part, elle reste très difficile à contrer ou éviter. C'est pour cela que cette attaque représente une menace que beaucoup craignent.

Attaque SYN Flood : L'attaque « **SYN Flood** » est une attaque exploitant le mécanisme de la poignée de main en trois temps du protocole TCP. Le mécanisme de poignée de main en trois temps est la manière selon laquelle toute connexion « fiable » à Internet (utilisant le protocole TCP) s'effectue.

Une connexion TCP ne peut s'établir que lorsque ces trois étapes ont été franchies. L'attaque SYN consiste à envoyer de requêtes SYN à un hôte avec une adresse IP source inexistante ou invalide. Ainsi, il est impossible que la machine cible reçoive un paquet ACK.

3.8.3. Attaque Man In The Middle :

L'attaque « **man in the middle** » notée MITM, est un scénario d'attaque dans lequel un pirate écoute une communication entre deux interlocuteurs et falsifie les échanges afin de se faire passer pour l'une des parties.

ARP Poisoning : C'est l'attaque la plus célèbre des attaques « **man in the middle** », elle consiste à s'interposer entre deux machines du réseau et de transmettre à chacune un paquet ARP falsifié indiquant que l'adresse MAC de l'autre machine a changé, et l'adresse MAC fournie étant celle de l'attaquant. Les deux machines cibles vont ainsi mettre à jour leurs tables dynamiques appelées Cache ARP. De cette manière, à chaque fois qu'une des deux machines souhaitera communiquer avec la machine distante, les paquets seront envoyés à l'attaquant, qui les transmettra de manière transparente à la machine destinatrice.

Vol de session TCP (TCP session hijacking) : L'**ARP poisoning** permet de rediriger tout le trafic IP mais, si l'attaquant n'a besoin que du trafic TCP, il peut interférer entre une

connexion client- serveur pour rediriger le flux du client vers lui. La synchronisation TCP est assurée par les numéros de séquences TCP. Si, pendant un échange, l'attaquant envoie des paquets malformés au client avec une adresse IP correspondant à celle du serveur en y plaçant des mauvais numéros de séquences, le client va croire qu'il a perdu la connexion et stoppera ses échanges avec le serveur. Mais si l'attaquant envoie les bons numéros de séquences au serveur, il récupèrera la connexion pour lui.

3.8.4. Les Sniffers :

Un sniffer (analyseur réseau) est un dispositif permettant d'écouter le trafic d'un réseau, c'est à dire de capter les informations qui y circulent.

3.8.5. Les ports scanners :

Un port scanner est un utilitaire permettant d'effectuer un balayage de ports ouverts sur une machine donnée ou sur un réseau tout entier. Le balayage se fait grâce à des sondes (requêtes) permettant de déterminer les services fonctionnant sur un hôte distant.

4. Les menaces sur les réseaux informatiques :

- ✓ Perte d'accessibilité au système ou au réseau :
  - Le but n'est pas de récupérer ou d'altérer des données, mais de nuire à des sociétés dont l'activité repose sur un système d'information en l'empêchant de fonctionner.

- ✓ Intrusion dans les systèmes :
  - Utilisation illicite d'un accès au système par une personne « extérieure » ou utilisation d'un compte différent du sien avec plus de privilèges (par ex. ceux de l'administrateur) par une personne « intérieure ».
    - Vol d'informations ;
    - modification d'informations ;
    - Destruction d'informations ;
    - Virus.

- ✓ Interception des messages :
  - Vol d'informations
  - Prise de connaissance des mots de passes.

- ✓ Répudiation : l'émetteur nie avoir émis ou le récepteur avoir reçu un message donné :
  - dû à une erreur : disparition de la trace ou de la preuve
  - dû à un acte de mauvaise foi : refus de responsabilité d'un sinistre, acte malveillant (refus d'une commande).

- ✓ Mascarade ou déguisement

  Usurpation d'identité consistant à se faire passer pour un utilisateur du système habilité
  - mécanisme de l'intrus : le pirate lance une transaction entre A et B
    - il dit à A qu'il est B et demande à A de s'authentifier.
    - il dit à B qu'il est A et s'authentifie.

- ✓ Saturation
  - génération de messages sur le réseau de manière à rendre le réseau inutilisable
  - rendre le réseau inutilisable pour tous les membres autorisés
  - saturer certains membres du réseau avec de l'information inutile et inoffensive (perte importante de capacité de traitement pour éliminer ces messages) [6].

5. Les outils de la sécurité :

5.1. Antivirus :

Les antivirus sont des logiciels conçus pour identifier, neutraliser et éliminer les logiciels malveillants (dont les virus ne sont qu'un exemple) qui se basent sur l'exploitation de failles de sécurité [6].

5.2. Système de détection d'intrusion (IDS) :

Un système de détection d'intrusion (ou IDS : Intrusion Detection System) est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions. Il existe deux grandes familles distinctes d'IDS :

- **Les N-IDS** (Network Based IDS), qui assurent la sécurité au niveau du réseau ;
- **Les H-IDS** (Host Based IDS), qui assurent la sécurité au niveau des hôtes [6].

5.3. Firewall :

Le firewall est un logiciel et/ou un matériel, permettant de faire respecter la politique de sécurité du réseau, celle-ci définissant quels sont les types de communication autorisés sur ce réseau informatique. Il mesure la prévention des applications et des paquets. (voir chapitre 2 pour plus de détailles) [6].

5.4. Biométrie :

La biométrie est une technique globale visant à établir l'identité d'une personne en mesurant une de ses caractéristiques physiques. Il peut y avoir plusieurs types de caractéristiques physiques, les unes plus fiables que d'autres, mais toutes doivent être infalsifiables et unique pour pouvoir être représentatives d'un et un seul individu. Par exemple : empreintes digitales [4].

5.5. Réseau virtuel privé (VPN) :

Un Réseau virtuel privé « VPN » (pour Virtuel Private Network), fait référence à l'usage du protocole IPSec afin de créer un canal de communication sécurisé à usage privé, dans un réseau public non sécurisé. Souvent mis en œuvre par une organisation, pour connecter ses différents sites via Internet afin d'assurer la confidentialité des données échangées [4].

5.6. Cryptographie :

La cryptographie est une science permettant de convertir des informations " en clair " en informations codées, c'est-à-dire non compréhensibles, puis, à partir de ces informations codées, de restituer les informations originales [4].

6. Mise en place d'une politique de sécurité :

La sécurité des systèmes informatiques consiste généralement à garantir les droits d'accès aux données et ressources d'un système en mettant en place des mécanismes d'authentification et de contrôle permettant d'assurer que les utilisateurs des dites ressources possèdent uniquement les droits qui leur ont été octroyés.

La sécurité informatique doit toutefois être étudiée de telle manière à ne pas empêcher les utilisateurs de développer les usages qui leur sont nécessaires, et de faire en sorte qu'ils puissent utiliser le système d'information en toute confiance.

C'est la raison pour laquelle il est nécessaire de définir dans un premier temps une politique de sécurité, c'est-à-dire :

- Elaborer des règles et des procédures à mettre en œuvre dans les différents services de l'organisation.
- Définir les actions à entreprendre et les personnes à contacter en cas de détection d'une Intrusion.
- Sensibiliser les utilisateurs aux problèmes liés à la sécurité des systèmes d'informations.

La politique de sécurité est donc l'ensemble des orientations suivies par une organisation (à prendre au sens large) en termes de sécurité. A ce titre elle se doit d'être élaborée au niveau de la direction de l'organisation concernée, car elle concerne tous les utilisateurs du système.

Le rôle de l'administrateur informatique est donc de faire en sorte que les ressources informatiques et les droits d'accès à celles-ci soient en cohérence avec la politique de sécurité retenue. De plus, étant donné qu'il est le seul à connaître parfaitement le système, il lui revient de faire remonter les informations concernant la sécurité à sa direction, éventuellement de la conseiller sur les stratégies à mettre en œuvre, ainsi que d'être le point d'entrée concernant la communication aux utilisateurs des problèmes et recommandations en termes de sécurité [3].

7. Conclusion :

Nous avons vu dans ce chapitre les différents types d’attaque ainsi que les outils à utiliser et la stratégie à entreprendre afin de mettre en place une politique de sécurité.

Malgré tous cela les hackers ne manquent pas d’ingéniosité pour inventer d’autres outils et méthode d’attaque, ce qui pousse à être à jours en s’informant sur les nouvelles techniques d’attaques et les nouveaux outils et mécanismes concernant la technologie de sécurité.

Chapitre 2
Firewall et Sécurité réseaux

1. Introduction :

Même si la sécurité d'un ordinateur et les informations qu'il contient sont bien assurées, cela reste insuffisant lorsqu'on veut communiquer avec d'autres ordinateurs que ce soit à partir d'un réseau local ou d'un réseau externe tel que internet.

Pour une sécurité accrue de notre système, nous devons augmenter le niveau de sécurité en utilisant les pare-feu (Firewall). Le pare-feu se situe entre votre ordinateur et le réseau et définit quelles ressources sont accessibles aux utilisateurs distants du réseau. S'il est correctement configuré, un pare-feu peut augmenter considérablement la sécurité du système.

2. Définition d'un firewall :

Un firewall est un dispositif séparant un réseau considéré comme "sûr" d'un réseau en principe "hostile". Le cas le plus fréquent est l'accès à Internet, mais cela permet de séparer aussi différents départements d'une même société. Ce dispositif est constitué de matériels, routeurs et ordinateurs, et de logiciels pour la partie active et la configuration.

Les pare-feu ont pour fonction de séparer un réseau de telle sorte que le trafic échangé entre ce réseau et l'extérieur soit contrôlé et que d'éventuelles attaques soient ainsi empêchées.

Un pare-feu est un ensemble de composants placé entre deux réseaux ayant les propriétés suivantes :

- ✓ tout le trafic transitant entre les deux réseaux passe nécessairement par le pare-feu.
- ✓ seul le trafic explicitement autorisé par la politique de sécurité appliquée localement est autorisé à passer au travers du pare-feu.

En théorie, il serait possible de filtrer les communications à partir des postes de travail.

Cependant, ces derniers hébergent généralement un grand nombre de logiciels qui sont autant de failles de sécurité potentielles car ces logiciels sont plus ou moins bien développés et configurés.

De plus, la gestion d'un parc d'ordinateurs auxquels ont accès une multitude d'utilisateurs est difficile. C'est pourquoi il est préférable, en plus de la sécurité mise en place

au niveau des postes de travail, de centraliser la sécurité dans un équipement spécialisé dédié à la sécurité et de placer cet équipement à la frontière du site.

Pour pénétrer dans le site, la sécurité de cet équipement devra d'abord être déjouée. Bien évidemment, pour assurer le meilleur niveau de sécurité, il est nécessaire de faire tourner un minimum de logiciels sur le pare-feu, de le configurer avec la plus grande précaution et de bien suivre les évolutions de ces logiciels afin d'éviter d'utiliser des versions obsolètes pour lesquelles des failles de sécurité auraient été découvertes [7].

3. Les différents types de filtrages :

3.1. Le filtrage simple de paquet (Stateless) :

3.1.1. Le principe :

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

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

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

3.1.2. Les limites :

Le premier problème vient du fait que l'administrateur réseau est rapidement contraint à autoriser un trop grand nombre d'accès, pour que le Firewall offre une réelle protection. Par exemple, pour autoriser les connexions à Internet à partir du réseau privé, l'administrateur devra accepter toutes les connexions TCP provenant de l'Internet avec un port supérieur à 1024. Ce qui laisse beaucoup de choix à un éventuel pirate [8].

3.2. Le filtrage de paquet avec état (Stateful) :

3.2.1. Le Principe :

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

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

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

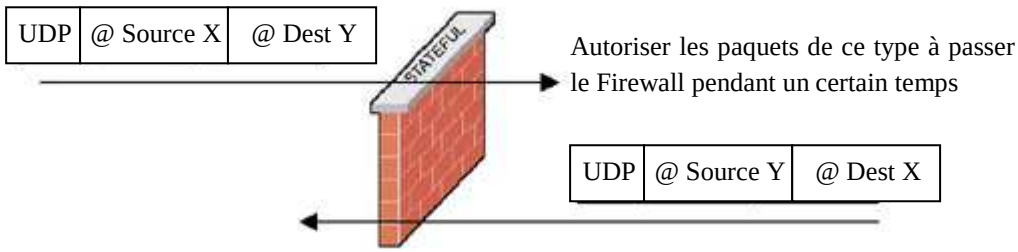

Figure 4. Filtrage de paquet avec état (UDP)

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

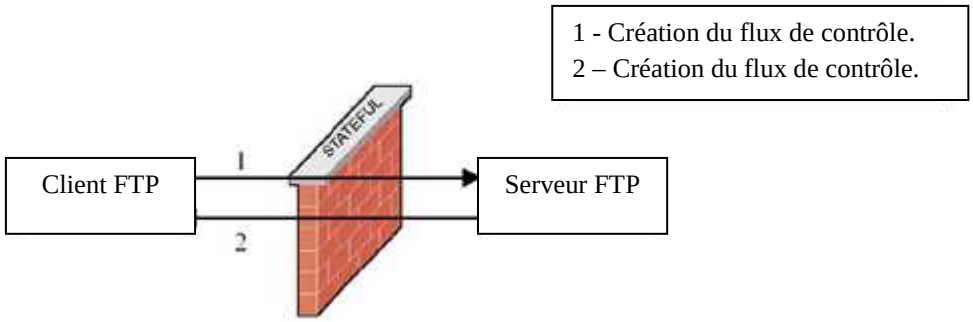

Figure 5. Filtrage de paquet avec état (FTP)

3.2.2. Les limites :

Tout d'abord, il convient de s'assurer que les deux techniques sont bien implémentées par les Firewalls, car certains constructeurs ne l'implémentent pas toujours correctement. Ensuite une fois que l'accès à un service a été autorisé, il n'y a aucun contrôle effectué sur les requêtes et réponses des clients et serveurs. Un serveur HTTP pourra donc être attaqué impunément. Enfin les protocoles maisons utilisant plusieurs flux de données ne passeront pas, puisque le système de filtrage dynamique n'aura pas connaissance du protocole [8].

3.3. Le filtrage applicatif (ou pare-feu de type proxy ou proxying applicatif) :

3.3.1. Le principe :

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

3.3.2. Les limites :

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

Mais il est indéniable que le filtrage applicatif apporte plus de sécurité que le filtrage de paquet avec état, mais cela se paie en performance. Ce qui exclut l'utilisation d'une technologie 100 % proxy pour les réseaux à gros trafic au jour d'aujourd'hui. Néanmoins d'ici quelques années, le problème technologique sera sans doute résolu [8].

4. Les différents types de firewall :

4.1. Les firewall bridge :

Ces derniers sont relativement répandus. Ils agissent comme de vrais câbles réseau avec la fonction de filtrage en plus, d'où leur appellation de bridge. Leurs interfaces ne possèdent pas d'adresse IP, et ne font que transférer les paquets d'une interface a une autre en leur appliquant les règles prédéfinies. Cette absence est particulièrement utile, car cela signifie que le firewall est indétectable pour un hacker lambda. En effet, quand une requête ARP est émise sur le câble réseau, le firewall ne répondra jamais. Ses adresses Mac ne circuleront jamais sur le réseau, et comme il ne fait que « transmettre » les paquets, il sera totalement invisible sur le réseau. Cela rend impossible toute attaque dirigée directement contre le firewall, étant donné qu'aucun paquet ne sera traité par ce dernier comme étant sa propre destination. Donc, la seule façon de le contourner est de passer outre ses règles de drop. Toute attaque devra donc « faire » avec ses règles, et essayer de le contourner [7].

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

4.1.1. Avantages :

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

4.1.2. Inconvénients :

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

4.2. Les firewalls matériels :

Ils se trouvent souvent sur des routeurs achetés dans le commerce par de grands constructeurs comme Cisco ou Nortel. Intégrés directement dans la machine, ils font office de « boite noire », et ont une intégration parfaite avec le matériel. Leur configuration est souvent relativement ardue, mais leur avantage est que leur interaction avec les autres fonctionnalités du routeur est simplifiée de par leur présence sur le même équipement réseau. Souvent relativement peu flexibles en terme de configuration, ils sont aussi peu vulnérables aux attaques, car présent dans la « boite noire » qu'est le routeur. De plus, étant souvent très liés au matériel, l'accès à leur code est assez difficile, et le constructeur a eu toute latitude pour produire des systèmes de codes « signés » afin d'authentifier le logiciel (système RSA ou assimilés). Ce système n'est implanté que dans les firewalls haut de gamme, car cela évite un remplacement du logiciel par un autre non produit par le fabricant, ou toute modification de ce dernier, rendant ainsi le firewall très sûr. Son administration est souvent plus aisée que les firewalls bridges, les grandes marques de routeurs utilisant cet argument comme argument de vente. Leur niveau de sécurité est de plus très bon, sauf découverte de faille éventuelle comme tout firewall. Néanmoins, il faut savoir que l'on est totalement dépendant du constructeur du matériel pour cette mise à jour, ce qui peut être, dans certains cas, assez contraignant. Enfin, seules les spécificités prévues par le constructeur du matériel sont implémentées. Cette dépendance induit que si une possibilité nous intéresse sur un firewall d'une autre marque, son utilisation est impossible. Il faut donc bien déterminer à l'avance ses besoins et choisir le constructeur du routeur avec soin [7].

4.2.1. Avantages :

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

4.2.2. Inconvénients :

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

4.3. Les firewalls logiciels:

Ils sont assez souvent commerciaux et ont pour but de sécuriser un ordinateur particulier, et non pas un groupe d'ordinateurs. Souvent payants, ils peuvent être contraignants

et quelque fois très peu sécurisés. En effet, ils s'orientent plus vers la simplicité d'utilisation plutôt que vers l'exhaustivité, afin de rester accessible à l'utilisateur final [7].

4.3.1. Avantages :

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

4.3.2. Inconvénients :

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

5. Configuration théorique des défenses :

La configuration d'un firewall est l'élément clef de son efficacité. Un firewall mal configuré peut être tout aussi efficace qu'aucun firewall du tout. C'est la clef de son bon fonctionnement et de son efficacité.

Il existe deux politiques de configurations différentes en ce qui concerne la « base » du firewall :

- ❖ Tout autoriser sauf ce qui est dangereux : cette méthode est beaucoup trop laxiste. En effet, cela laisse toute latitude à l'imagination des intrus de s'exprimer. Et à moins d'avoir tout prévu de façon exhaustive, on laissera forcément des portes ouvertes, des failles béantes dans notre système. A éviter absolument.
- ❖ Tout interdire sauf ce dont on a besoin et ce en quoi on a confiance : cette politique est beaucoup plus sécuritaire. En effet, les services sont examinés avant d'être autorisés à passer le firewall, et sont donc tous soumis à un examen plus ou moins approfondi. Ainsi, pas de mauvaise surprise sur un service que l'on pensait ne pas avoir installé, plus d'oubli : tout service autorisé est explicitement déclaré dans le firewall. Cette politique s'accompagne de la création de deux zones : une zone interne et l'extérieur. On peut considérer que tout ce qui est dans notre réseau local est autorisé, sans prendre de trop gros risques : le firewall est là pour nous protéger des attaques extérieures, pas des attaques internes pour lesquelles il ne peut rien. Cette facette peut changer suivant la politique de l'entreprise (interdire l'accès à certains, jeux,….etc.). La zone externe est par contre considérée comme « non sûre », et donc toute requête

envoyée sur un service non explicitement déclaré comme accessible de l'extérieur sera interceptée et ignorée. La configuration de la DMZ est ici très importante, et sa gestion aussi.

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

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

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

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

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

- ✓ Interdire le FTP.

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

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

La configuration efficace d'un firewall n'est pas chose évidente, et implique une grande rigueur, la moindre erreur ouvrant une brèche exploitable par les hackers [7].

6. Les réactions des firewalls aux attaques classiques :

6.1. IP spoofing :

L'IP spoofing consiste à modifier les paquets IP afin de faire croire au firewall qu'ils proviennent d'une adresse IP considérée comme « de confiance ». Par exemple, une IP présente dans le réseau local de l'entreprise. Cela laissera donc toute latitude au hacker de passer outre les règles du firewall afin d'envoyer ses propres paquets dans le réseau de l'entreprise. Les derniers firewalls peuvent offrir une protection contre ce type d'attaque, notamment en utilisant un protocole VPN, par exemple IPSec. Cela va crypter les entêtes des paquets, et ainsi rendre impossible leur modification par un intrus, et surtout, l'intrus ne pourra générer de paquets comme provenant de ce réseau local, ce dernier n'ayant pas la clé nécessaire au cryptage [8].

6.2. DOS et DDOS :

Le DOS, ou Denial Of Service attack, consiste à envoyer le plus de paquets possibles vers un serveur, générant beaucoup de trafic inutile, et bloquant ainsi l'accès aux utilisateurs normaux. Le DDOS, pour Distributed DOS, implique venir de différentes machines simultanées, cette action étant le plus souvent déclenchée par un virus : ce dernier va d'abord infecter nombre de machines, puis à une date donnée, va envoyer depuis chaque ordinateur infecté des paquets inutiles vers une cible donnée. On appelle aussi ce type d'attaque « flood ». Les firewalls ici n'ont que peu d'utilité. En effet, une attaque DOS ou DDOS utilise le plus souvent des adresses sources différentes (le but n'est pas de récupérer une réponse ici) et souvent, impossible de distinguer ces paquets des autres. Certains firewalls offrent une

protection basique contre ce genre d'attaque, par exemple en droppant les paquets si une source devient trop gourmande, mais généralement, ces protections sont inutiles. Cette attaque brute reste un des gros problèmes actuels, car elle est très difficilement évitable [8].

6.3. Port scanning :

Ceci constitue en fait une « pré-attaque » (Etape de découverte). Elle consiste à déterminer quels ports sont ouverts afin de déterminer quelles sont les vulnérabilités du système. Le firewall va, dans quasiment tous les cas, pouvoir bloquer ces scans en annoncent le port comme « fermé ». Elles sont aussi aisément détectables car elles proviennent de la même source faisant les requêtes sur tous les ports de la machine. Il suffit donc au firewall de bloquer temporairement cette adresse afin de ne renvoyer aucun résultat au scanner [8].

6.4. Exploit :

Les exploits se font en exploitant les vulnérabilités des logiciels installés, par exemple un serveur HTTP, FTP,.…etc. Le problème est que ce type d'attaque est très souvent considéré comme des requêtes tout a fait « valides » et que chaque attaque est différente d'une autre, vu que le bug passe souvent par reproduction de requêtes valides non prévues par le programmeur du logiciel. Autrement dit, il est quasiment impossible au firewall d'intercepter ces attaques, qui sont considérées comme des requêtes normales au système, mais exploitant un bug du serveur le plus souvent. La seule solution est la mise à jour périodique des logiciels utilisés afin de barrer cette voie d'accès au fur et à mesure qu'elles sont découvertes [8].

7. Conclusion :

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

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

Chapitre 3
Conception et mise en Œuvre

1. Introduction :

Nous avons vu lors du chapitre précédant les différentes fonctionnalités qu'offre un système Firewall, ainsi que les types de filtrage. Notre choix s'est porté sur le premier type vu la complexité des autres types.

2. Cycle de développement d'un logiciel :

UML (en anglais Unified Modeling Language ou « langage de modélisation unifié ») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique [9].

Nous avons opté pour le modèle en cascade, les phases traditionnelles de développement sont effectuées simplement les unes après les autres, avec un retour possible vers les phases précédentes, voire au tout début du cycle.

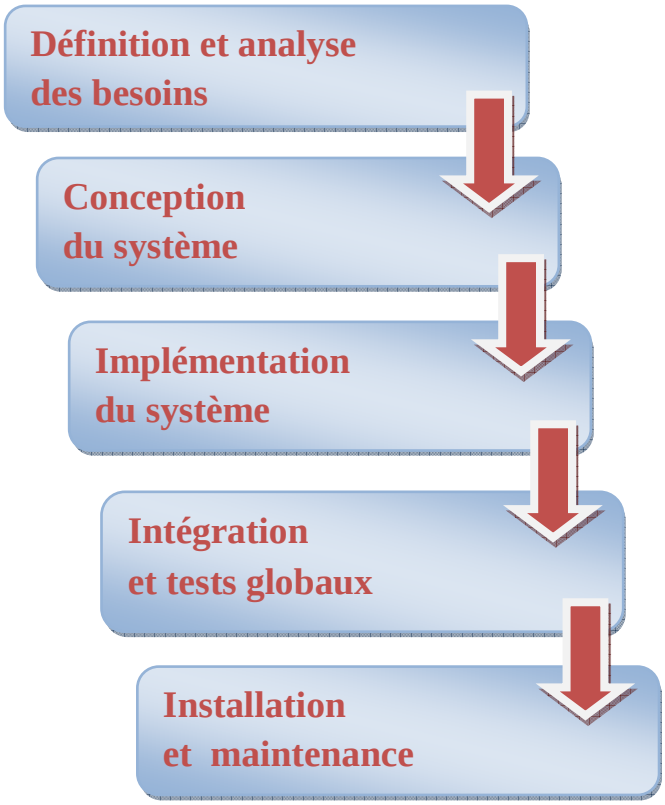

Figure 6. Modèle en cascade

2.1. Définition et analyse des besoins :

La phase de définition et d'analyse des besoins permet de décrire les fonctionnalités du futur logiciel et les contraintes sous lesquelles celui-ci doit être réalisé.

Notre application doit être en mesure d'assurer un filtrage de paquets en utilisant des règles de filtrage mais aussi doit permettre de bloquer un port (ou une plage de port) et de bloquer une adresse IP (ou une plage d'adresses IP) sans se soucier des règle de filtrage, elle intègre aussi un scan de port qui est un moyen qui nous guidera dans la prise les décisions dans la configuration de notre Firewall.

En résumer notre application se compose de quatre modules qui assurent les fonctionnalités suivantes :

- ✓ Un filtrage de paquets.
- ✓ Un blocage de port (ou plage de ports).
- ✓ Un blocage d'adresses IP (ou une plage d'adresses IP).
- ✓ Un scan de port.

On doit déterminer pour chacun des modules un diagramme de cas d'utilisation qui est une solution pour structurer les besoin des utilisateurs et les objectifs, il représente aussi une unité discrète d'interaction entre un utilisateur et un système. Dans un diagramme de cas d'utilisation, les utilisateurs interagissent avec les cas d'utilisation.

Utilisateur : Un utilisateur est l'entité externe qui agit sur le système. Dans notre application, nous distinguons un seul utilisateur qui est : L'administrateur.
Un administrateur doit avoir la possibilité à travers cette application d'effectuer un ensemble de règles sur les paquets, de géré ces port et de bloquer une connexion avec une machines distante en boquant son adresse IP.

2.1.1. Filtrage de paquets :

Le filtrage de paquets et le premier module de notre application, il permet d'ajouter et de supprimer des règles de filtrage, il permet aussi de visualiser les règles de filtrage déjà définis.

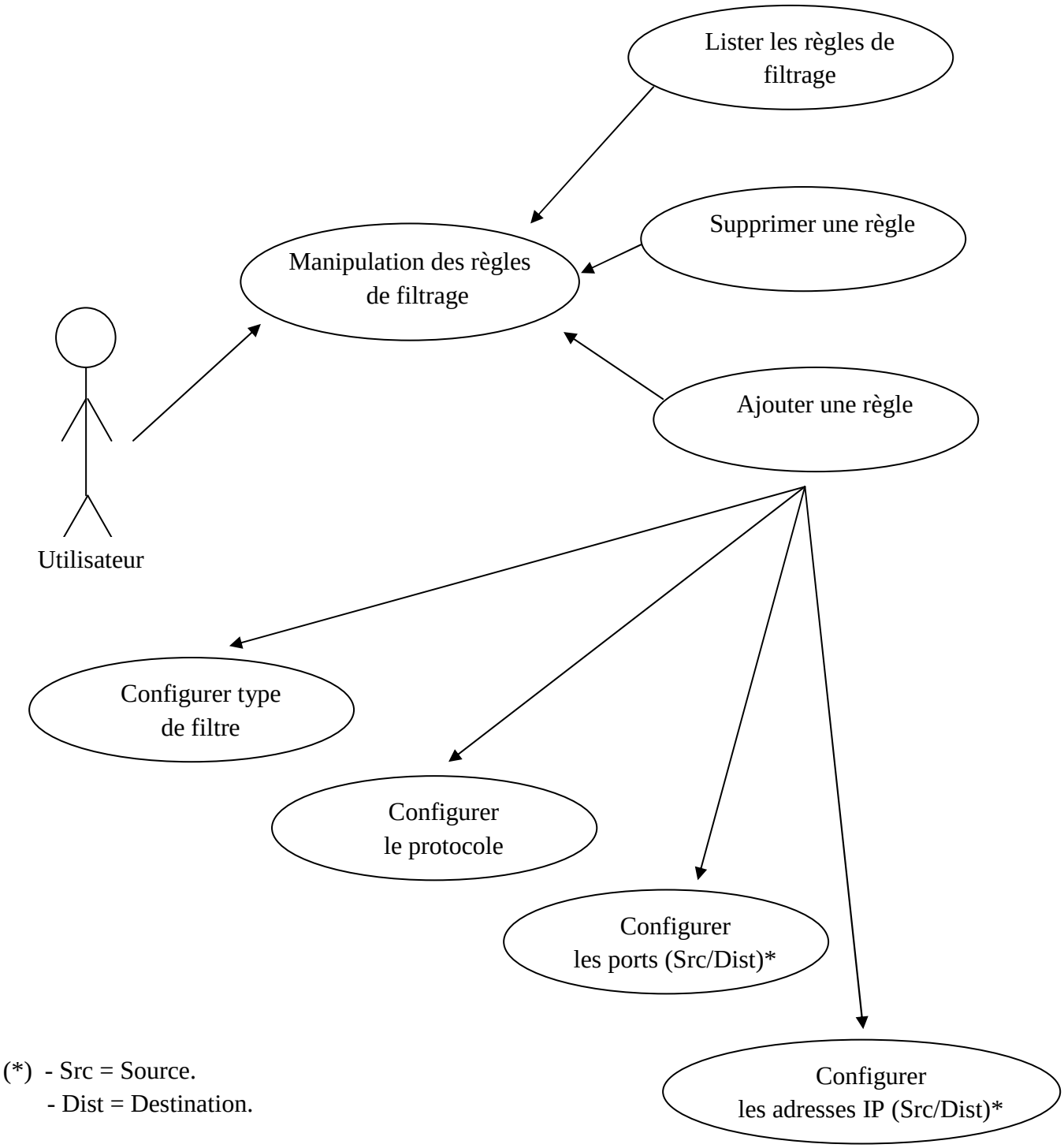

Figure 7. Cas d'utilisation « Filtrage de paquets »

2.1.2. Blocage de ports :

Ce module comporte trois (3) fonctionnalités qui sont :

- ✓ Bloquer un seul port.
- ✓ Bloquer le Ping.
- ✓ Bloquer une plage d'adresses IP.

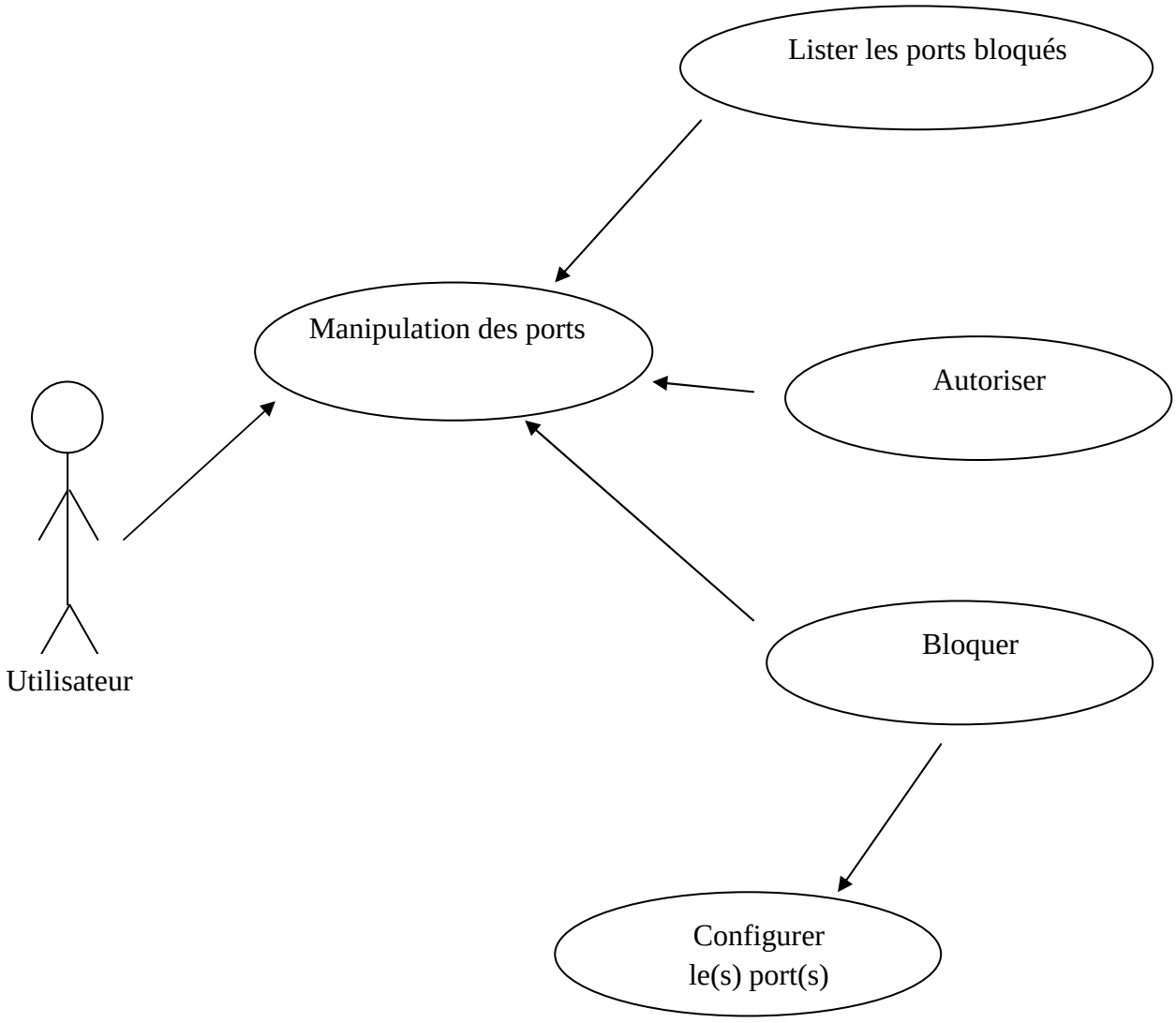

Figure 8. Cas d'utilisation « Blocage de ports »

2.1.3. Blocage d'adresses IP :

Ce module comporte deux (2) fonctionnalités qui sont :

- ✓ Bloquer une seul adresse IP.
- ✓ Bloquer une plage d'adresses IP.

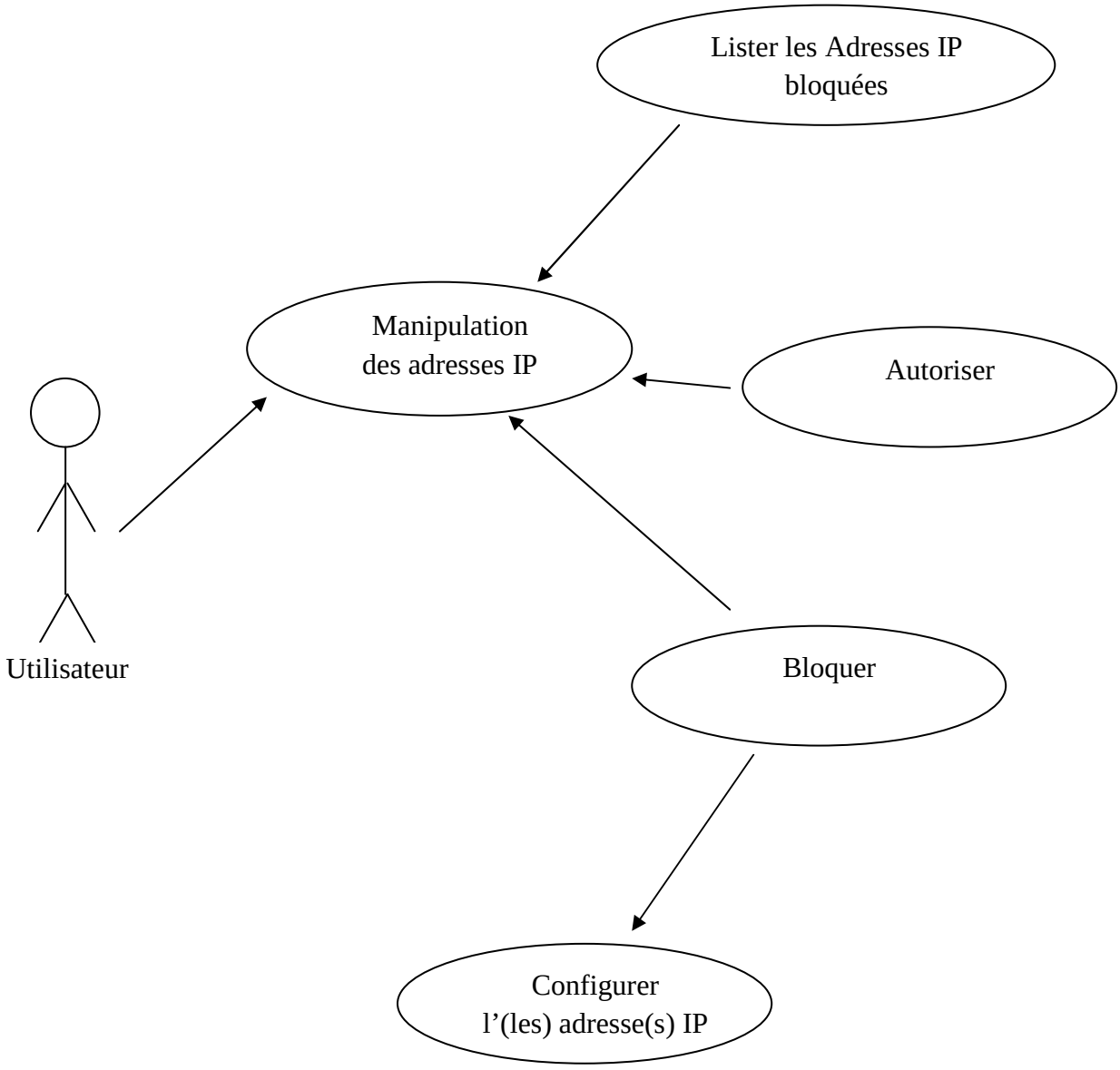

Figure 9. Cas d'utilisation « Blocage d'adresses IP »

2.1.4. Scan de port :

Ce module permet de faire un balayage des ports et d'indiquer leurs états.

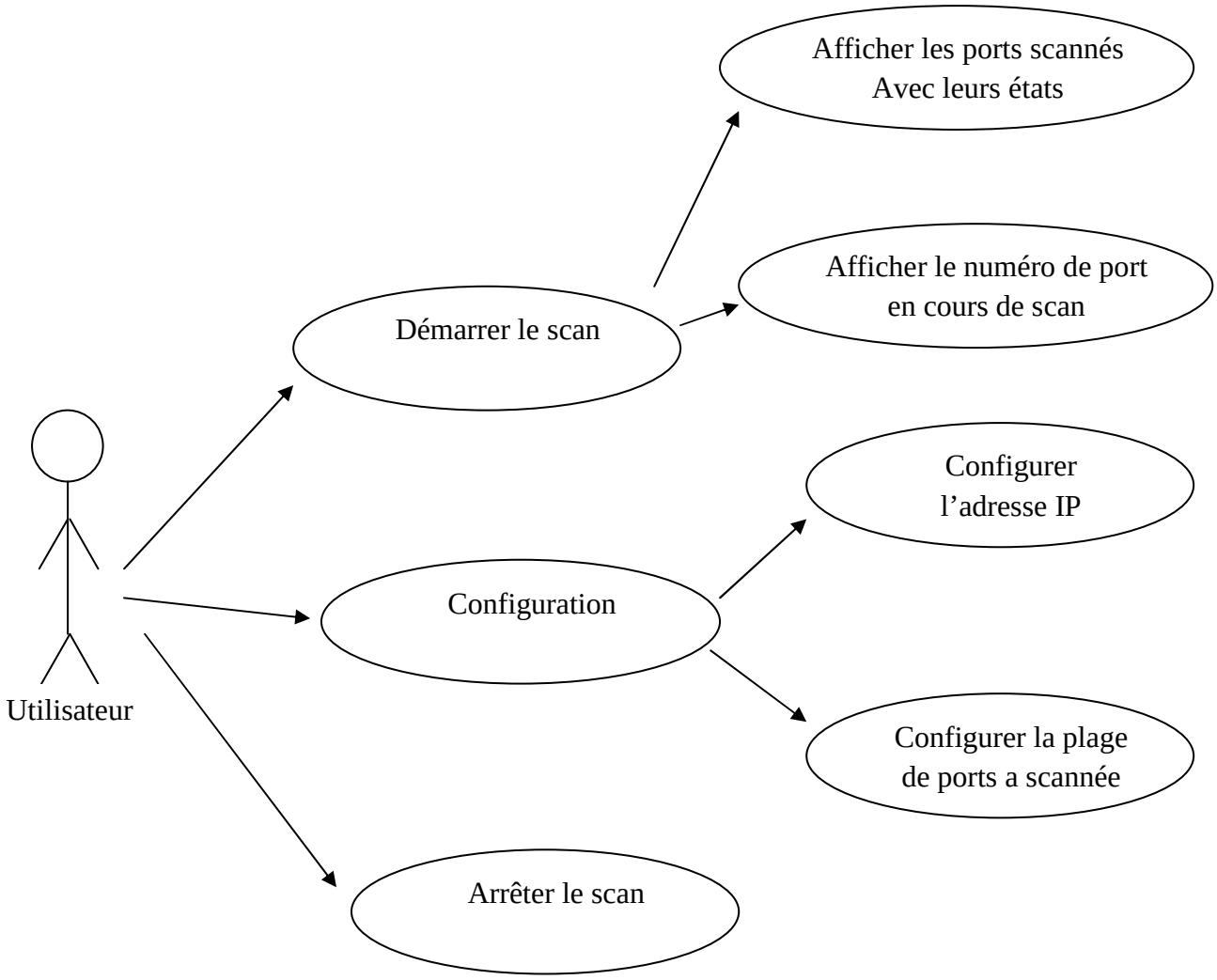

Figure 10. Cas d'utilisation « Scan de ports »

2.2. Conception :

- **Différence entre l'étape « définition et analyse des besoins » et l'étape « conception » :**

L'étape « définition et analyse des besoins » englobe la conception. Ceci étant dit, la première étape est définie comme l'expression de toutes les caractéristiques de l'objet à développer selon une vue externe (comportements, propriétés, contraintes, etc.) et que la seconde sera définie comme la description de l'objet à développer selon une vue interne (structures et comportements des composants) [9].

La phase de conception permet de décrire de manière non ambiguë, le fonctionnement du futur du système, afin d'en faciliter la réalisation.

On doit déterminer un diagramme de classe ainsi qu'un diagramme de séquence pour chacun des modules de notre application.

- **Diagramme de classe :**

Le **diagramme de classes** est un schéma utilisé afin de présenter les classes et les interfaces des systèmes.
Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensembles par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples [9].

- **Diagramme de séquences :**

Les **diagrammes de séquences** sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique.
On montre ces interactions dans le cadre d'un scénario d'un diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets [9].

2.2.1. Filtrage de paquets :

Dans ce module on a utilisé un Activex nommé (SocketBlock) ainsi qu'une base de données contenant une seule table ou sera stocker l'ensemble des règles définis.

Table des règles
Cle :int AdrDstInf : char AdrDstSup: char PortDstInf : int PortDstSup: int AdrSrcInf : char AdrSrcSup: char PortSrcInf : int PortSrcSup : int Typedefiltrage : char Protocole : char
Ajouter règle (Typedefiltrage, AdrDstInf, AdrDstSup, PortDstInf, PortDstSup, AdrSrcInf, AdrSrcSup, PortSrcInf, PortSrcSup, Protocole) : int Supprimer règle (Cle): int

Figure 11. Diagramme de class « Table des règles »

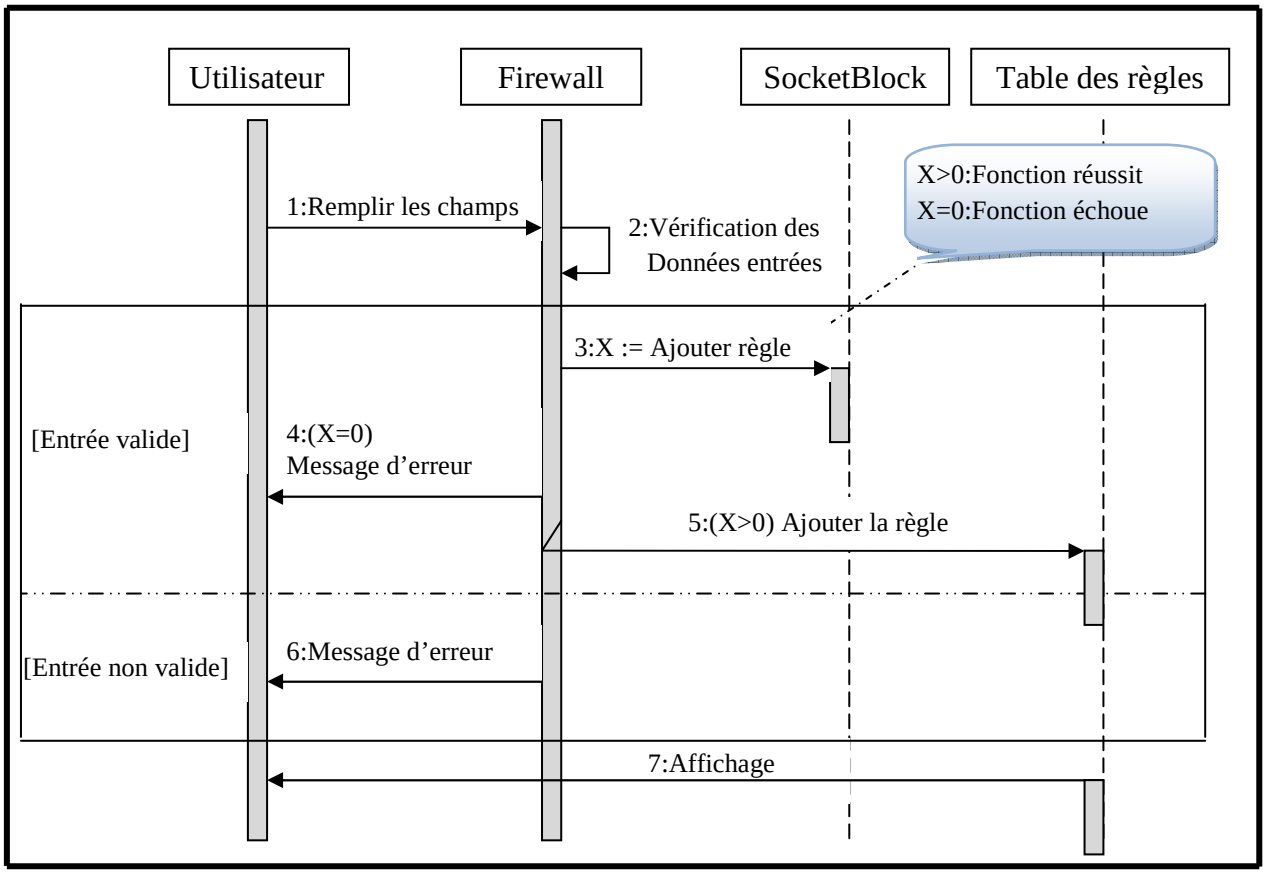

Figure 12. Diagramme de séquences « Ajouter une règle »

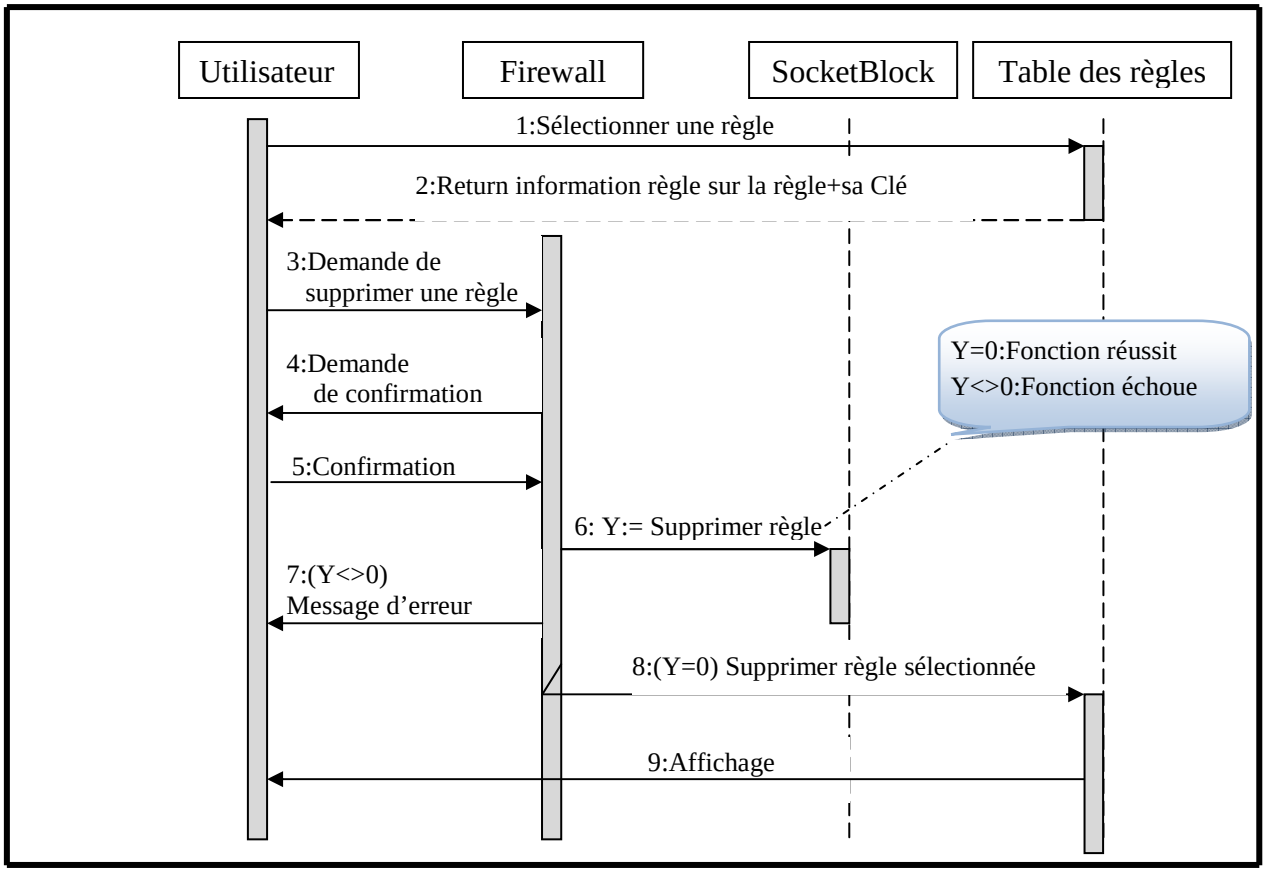

Figure 13. Diagramme de séquences « Supprimer une règle »

2.2.2. Blocage de ports :

Dans ce module en utilise l'Activex SocketBlock ainsi qu'une base de données contenant deux tables, la première contiendra la liste des ports bloqués et la deuxième contiendra la liste des plages de ports bloqués.

Table des ports bloqués
Cle : int Port: int Protocole : char
Bloquer (Port, Protocole) : int Autoriser (Cle): int

Figure 14. Diagramme de class « Table des ports bloqués »

Table des plages de ports bloqués
Cle : int PortInf: int PortSup: int Protocole : char
Bloquer (PortInf, PortSup, Protocole) : int Autoriser (Cle): int

Figure 15. Diagramme de class « Table des plages de ports bloquées »

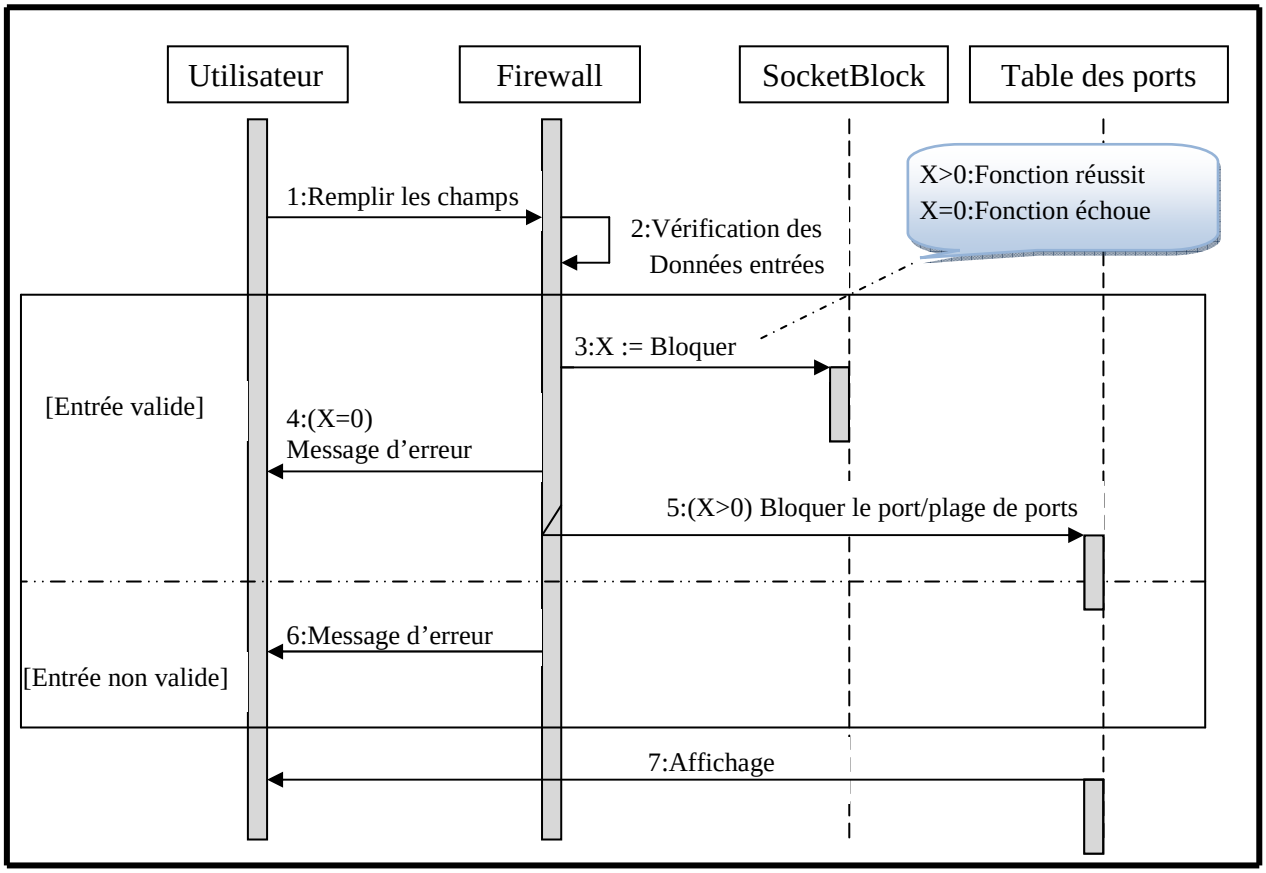

Figure 16. Diagramme de séquence « Bloquer port / plage de ports »

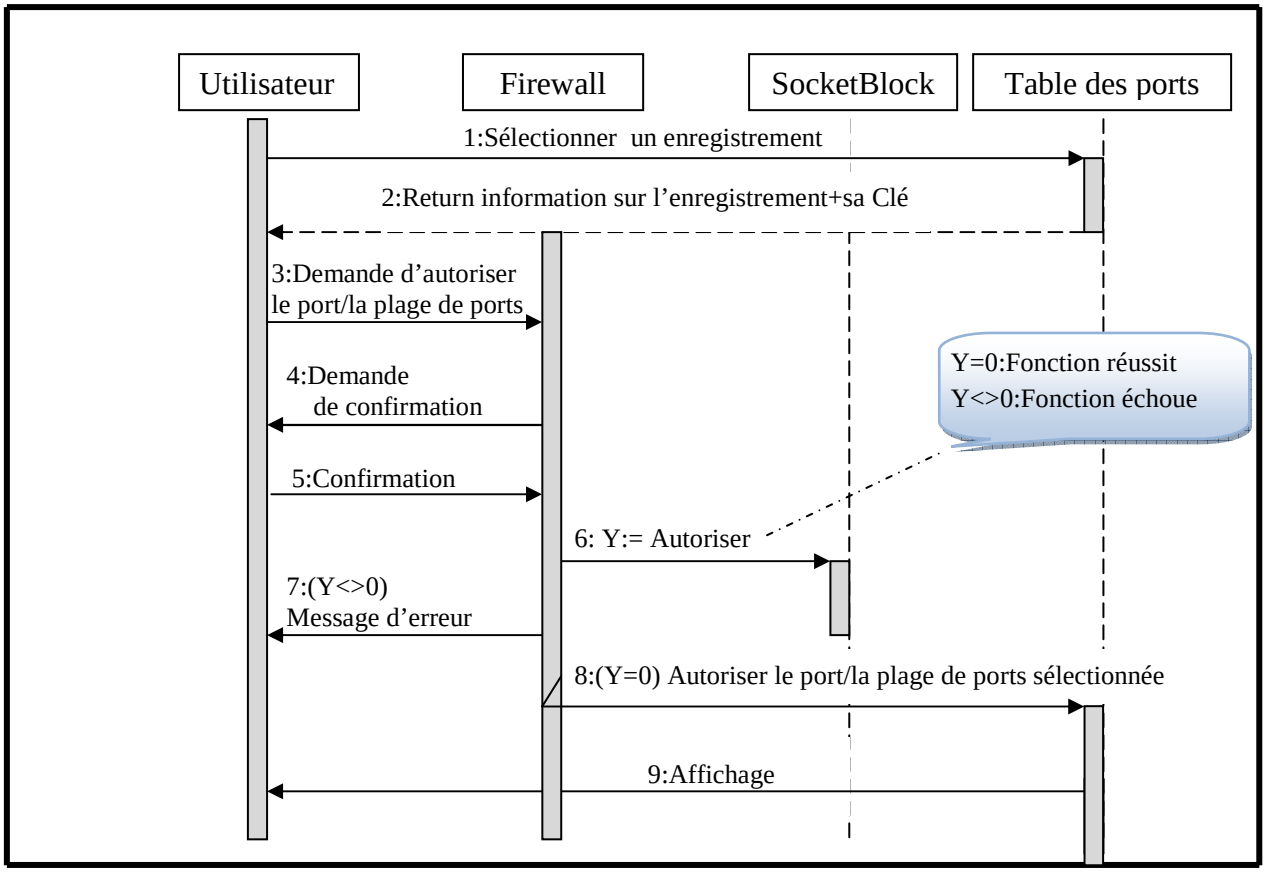

Figure 17. Diagramme de séquence « Autorise port / plage de ports »

2.2.3. Blocage d'adresses IP :

Dans ce module en utilise l'Activex SocketBlock ainsi qu'une base de données contenant deux tables, la première contiendra la liste des Adresses bloquées et la deuxième contiendra la liste des plages d'adresses bloquées.

Table des adresses bloquées
Cle : int Adr : char Protocole : char
Bloquer (Adr, Protocole) : int Autoriser (Cle): int

Figure 18. Diagramme de class « Table des adresses bloquées »

Table des plages d'adresses bloquées
Cle : int AdrInf: int AdrSup: int Protocole : char
Bloquer (AdrInf, AdrSup, Protocole) : int Autoriser (Cle): int

Figure 19. Diagramme de class « Table des plages de ports bloqués »

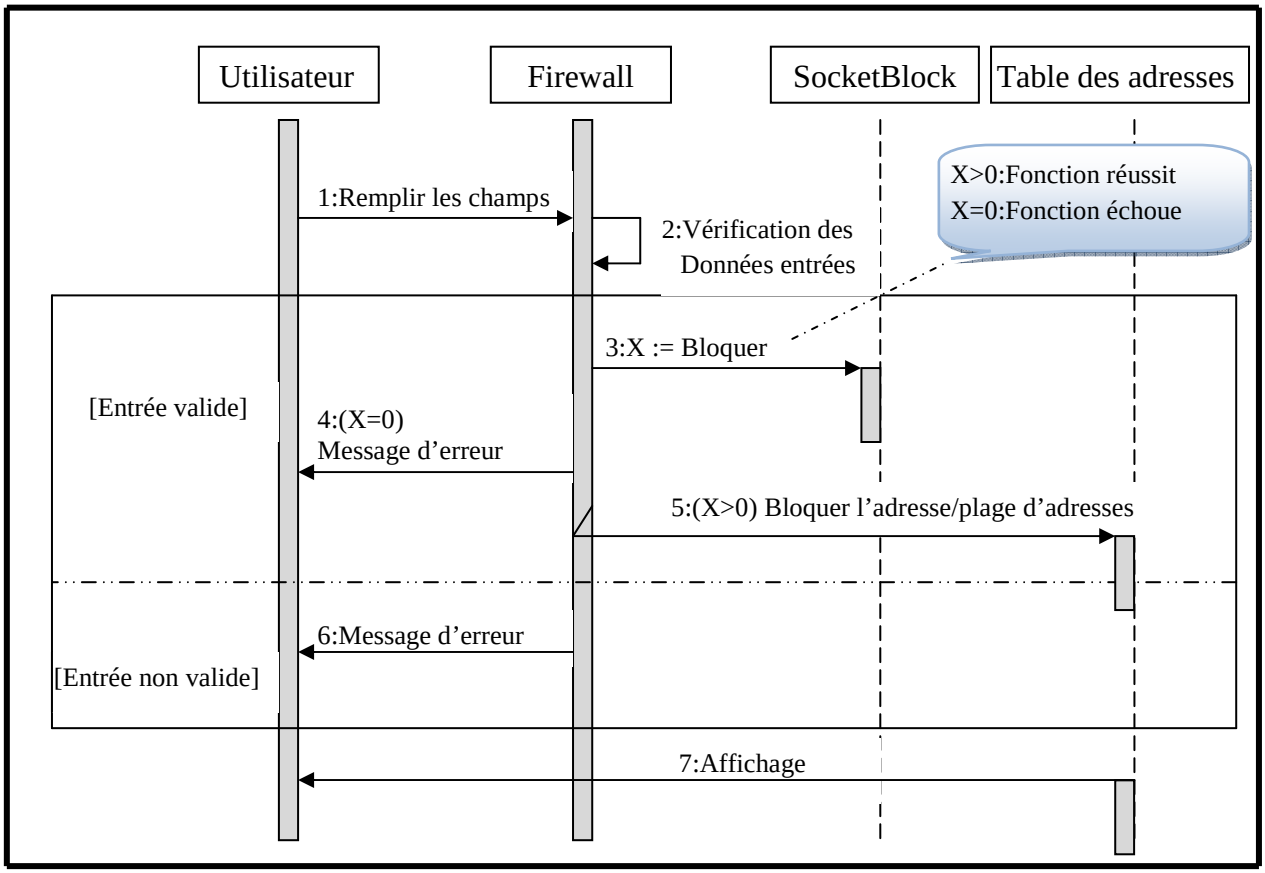

Figure 20. Diagramme de séquence « Bloquer adresse / plage d'adresses IP »

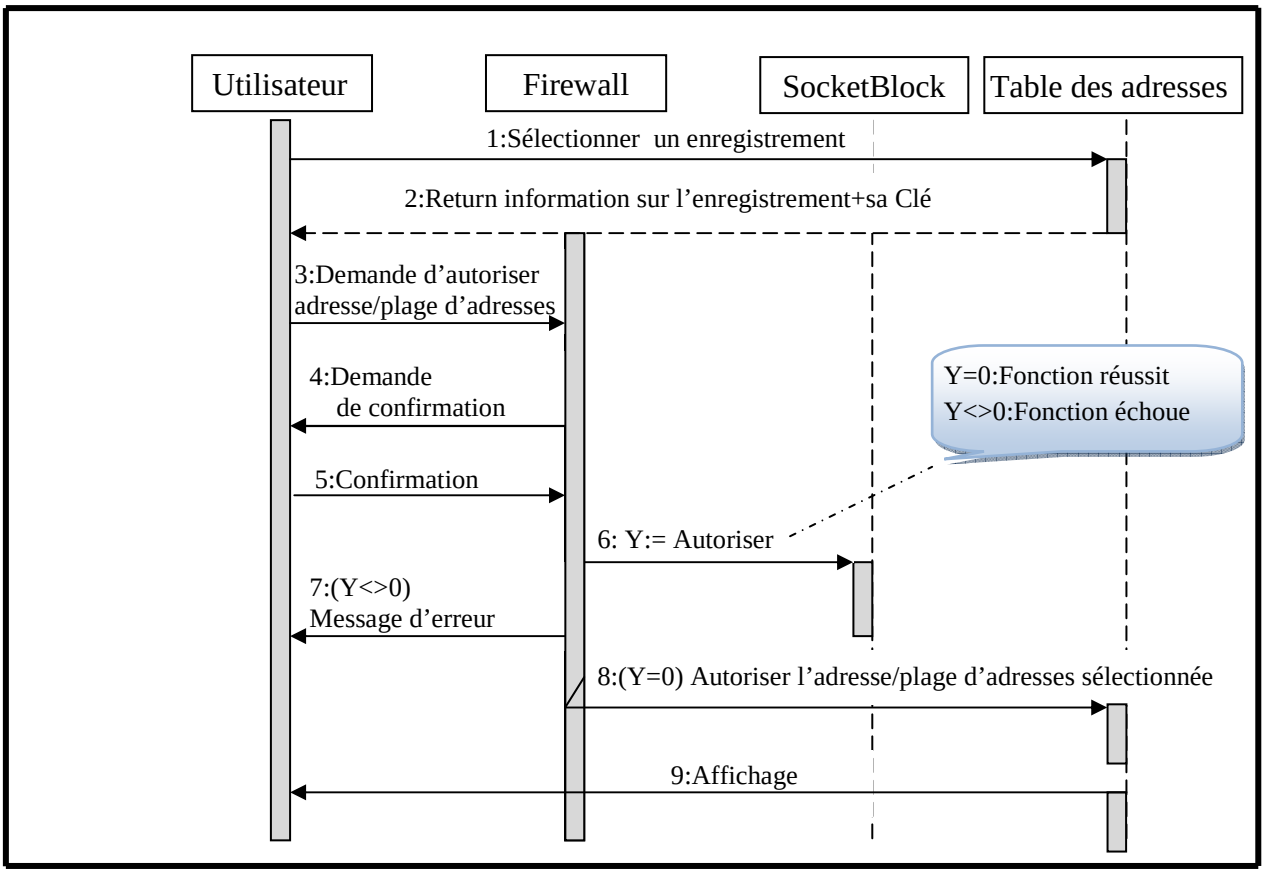

Figure 21. Diagramme de séquence « Autoriser adresse / plage d'adresses IP »

2.2.4. Scan de port :

Dans ce module on va créer plusieurs socket, les connecté au port et analyser le résultat. S'il y une réponse c'est que le port est ouvert, sinon le port est fermer.

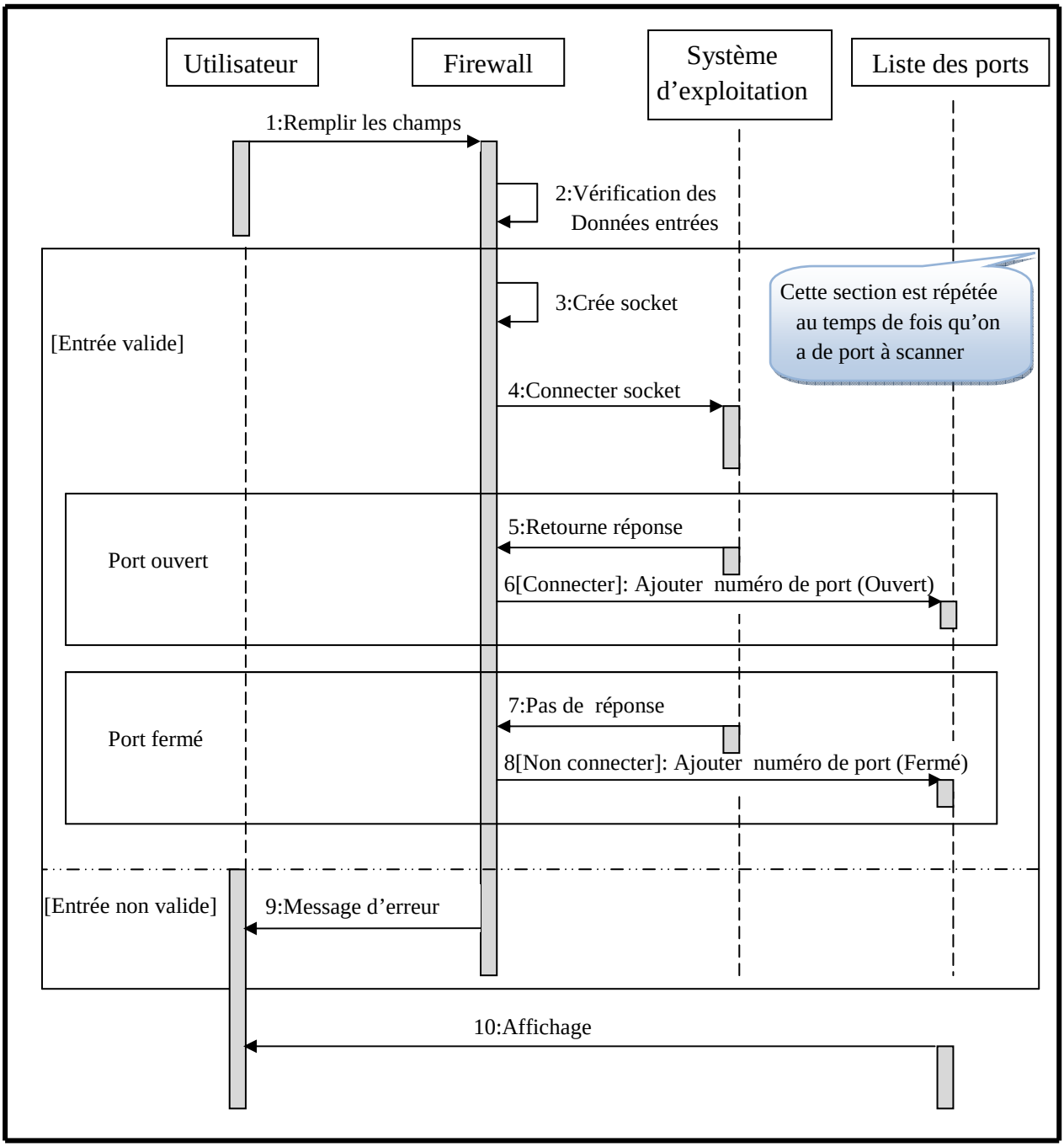

Figure 22. Diagramme de séquence « Scan de ports »

2.3. Implémentation :

2.3.1 Outils de développement :

- **Choix de Delphi :**

Delphi désigne à la fois un environnement de développement intégré (EDI) et un langage de programmation orienté objet.

L'EDI Delphi est un EDI fonctionnant sous Windows créé en 1995 et édité par Borland. À l'époque, créer des programmes graphiques sous Windows se faisait en grande majorité en utilisant soit la chaîne de compilation Visual C++, soit le RAD Visual Basic. Le premier outil étant excessivement complexe et le second assez peu structuré, Delphi apparut alors comme une alternative viable pour beaucoup de développeurs qui souhaitaient créer des programmes standards pour Windows.

Comme il s'agit d'un outil RAD (Rapid Application Développent), Delphi est très simple à prendre en main et il est tout à fait aisé de créer toutes sortes d'applications.

L'environnement de développement s'appuie sur un éditeur d'interface graphique associé à un éditeur de code source. Il doit son succès à sa facilité d'utilisation pour développer des applications graphiques et/ou liées aux bases de données. On l'a souvent comparé à Visual Basic de Microsoft pour cette facilité de développement.

L'interface de développement permet l'ajout de composants tiers (graphiques ou non) via un système de composants. La modularité est obtenue à la conception mais peut aussi être exploitée à l'exécution via un système de chargement dynamique de paquets d'exécution, Borland ayant étendu le concept de bibliothèques partagées et le format Windows DLL en introduisant un modèle propriétaire permettant d'enregistrer dynamiquement et d'exporter des classes entre modules. Le même système sera repris par Microsoft sous Visual Basic avec le format VBX, puis ensuite à l'échelle du système avec les composants COM et ActiveX.

L'Activex « Socketblock » que nous avons utilisés pour développé les fonctionnalités de notre firewall est compatible avec l'environnement Delphi.

2.3.2. Modules développées :

- Pour réaliser les trois premier module (Filtrage de paquets, Blocage de ports, Blocage d'adresses IP) on a choisit d'utiliser l'Activex « SocketBlock ».

Définition :

SocketBlock est un ActiveX permettant de filtrer les paquets échangés sur un réseau sans devoir écrire des codes bas niveau, en établissant des règles de filtrage qui accordent ou refuse l'accès au paquets en se basant sur leurs entête. Quelque soit le type d'application réseau, toutes les applications sont sensibles aux attaques, que ce soit par collègues curieux sur un intranet, ou des pirates sur Internet. SocketBlock aide à sécuriser les applications ainsi que la machine hôte en bloquant les paquets entrants et sortants, grâce aux filtres que le développeur met en place.

Les instructions et les méthodes du SocketBlock :

1. Activate :

Cette instruction (méthode) permet d'activer le composant. Cette méthode doit être appelée avant d'utiliser une autre méthode ou autres propriétés.

Syntaxe : SockBlock.Activate

Type de retour de données : Long.

Remarque : Cette méthode peut échouer avec l'un des « SBK_GENERAL_ERRORS » pour la non vérification de la licence (licence incorrecte ou manquante), l'échec de la version d'exploitation de la plateforme (pas Windows 2000 et plus), paramètre non valide (si le composant a déjà été activé), « SBK_ERROR_INSTANCE_ALREADY _RUNNING » si un composant est déjà en cours d'exécution, ou « SBK_ERROR_NO_ADMIN_PRIVILEGE » si l'utilisateur actuel ne dispose pas de privilèges d'administration.

2. AddFilter :

Cette instruction permet d'ajouter un filtre de paquets sur une base de critères (règles), et peut être appelée avant ou après la méthode « Start ».

Syntaxe : SockBlock.AddFilter (SBK_FILTER_TYPE : le type de filtre, String/BSTR : l'adresse destination inférieure, String/BSTR : l'adresse destination supérieure, Long : le port destination inférieur, Long : le port destination supérieur, String/BSTR : l'adresse source inférieure, String/BSTR : l'adresse source supérieure, Long : le port source inférieur, Long : le port source supérieur, SBK_IP_PROTOCOLS : le type de protocole).

Type de protocole : Pour filtrer les paquets sur tous les autres protocoles à part TCP (SBK_IPPROTO_TCP) et UDP (SBK_IPPROTO_UDP), un filtre peut être ajouté en utilisant l'instruction « SBK_IPPROTO_IP ». Cela signifie que le filtre à appliquer à tous les autres protocoles en dehors TCP et UDP est plus efficace que l'ajout de chaqu'un des autres protocoles à son filtre.

Remarque : Cette méthode est utilisée pour mettre en place les paquets qui seront bloqués et ceux qui seront autoriser. Il est essentiel de comprendre deux points importants sur le fonctionnement du filtrage ; Le premier point c'est que les critères de filtre dans un enregistrement sont évalués à l'aide de la logique : le paquet doit satisfaire tous les critères d'une règle pour que cette dernière prenne effet. Le second point c'est que les filtres sont évalués dans l'ordre avec lequel ils sont ajoutés (s'il n'y a pas de règles ajoutées, le filtrage de paquets permet le passage de tous les paquets).

Type de retour de données : Long.

3. DeleteFilter :

Cette méthode permet de supprimer un enregistrement de filtre ou une règle particulière de filtrage.

Syntaxe : SockBlock.DeleteFilter (Clé du filtre) : le filtre traité (Long).

Type de retour de données : Long.

Remarques : Dans cette méthode, une Clé de filtre est ajoutée lorsque le « AddFilter » est appelé. Une clé de filtre ne peut jamais être inférieur à un. Cette méthode peut échouer si la clé de filtre n'est pas valide, ou s'il y a une erreur de communication avec le pilote. Le succès de cette méthode retourne zéro.

4. Start :

Cette instruction permet le démarrage des services de filtrage de paquets. Les paquets ne sont pas filtrés jusqu'à ce que cette méthode soit appelée.

Syntaxe : SockBlock.Start

Type de retour de données : Long.

Remarques : Cette méthode ne devrait pas être appelée si le filtrage de paquets a déjà commencé, sauf si il a été interrompu en utilisant la méthode « Stop ». Cette méthode peut échouer si l'un des filtrages de paquets est déjà en cours, ou si une erreur se produit alors qu'il tentait de communiquer avec le pilote. Retourne zéro en cas de succès.

5. Stop :

Cette instruction permet d'arrêter les services de filtrage. Aucun des paquets n'est filtré après que cette méthode soit appelée.

Syntaxe : SockBlock.Stop

Type de retour de données : Long.

Remarques : Cette méthode ne devrait pas être appelée si le filtrage de paquets a déjà été arrêté, à moins qu'il ait été lancé à nouveau en utilisant la méthode « Start ». Cette méthode peut échouer si une erreur se produit alors qu'il tentait de communiquer avec le pilote. Retourne zéro en cas de succès.

6. IsAdmin :

Cette méthode sert à vérifier si l'utilisateur connecté au moment de l'exécution possède les privilèges d'administrateur ou pas.

Syntax : SockBlock.IsAdmin

Type de retour de données : VARIANT_BOOL / Booléen

7. LocalHostIPAliases Property :

Cette instruction renvoie un tableau contenant toutes les adresses IP de l'hôte.

Syntax : SockLite.LocalHostIPAliases.

Type de retour de données : Variant (Contient un tableau de Strings / BSTR).

- ✓ **Algorithme d'analyse (Filtrage de paquets) :**

Si il n'existe aucun filtre

Alors le paquet est autorisé à passé ;

Sinon Analyser l'entête du paquet ;

Pour chaque règle de filtrage

Faire

Si le filtre correspond aux caractéristiques du paquet

Alors

Si le filtre est bloquant

Alors le paquet est rejeté ;

Sinon le paquet est autorisé

Fsi

Fsi

Fait

Si le paquet ne correspond à aucun filtre

Alors le paquet est autorisé à passer ;

Fsi

Fsi

- ✓ **Algorithme d'analyse (Blocage de port(s) et adresse(s) IP) :**

Si il n'existe aucun Blocage

Alors le paquet est autorisé à passé ;

Sinon Analyser l'entête du paquet ;

Pour chaque blocage

Faire

Si le blocage correspond aux caractéristiques du paquet

Alors

le paquet est rejeté ;

Sinon le paquet est autorisé

Fsi

Fait

Si le paquet ne correspond à aucun filtre

Alors le paquet est autorisé à passer ;

Fsi

Fsi

- Pour réaliser le module (Scan de ports) on a utilisé un composant de Delphi « TClientSocket », car ce composant nous permet de :

- Gérer er relier les codes aux événements : le succès ou l'échec de la connexion d'un socket (Onconnect et Onerror) ;

- Lancer plusieurs demandes de connexion en parallèle.

Le principe de ce module est de connecter les sockets au port et d'analyser le résultat. S'il y une réponse c'est que le port est ouvert, sinon le port est fermer, le principe de ce module est représenté par l'algorithme suivant :

✓ Algorithme d'analyse (Scan de port) :

```
Unit PortScan;

Interface

Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ScktComp;

Type
  TmainForm = Class(Tform)
    LblIPAddress: Tlabel;
    IPAddressE: Tedit;
    lblScanRange: Tlabel;
    MinPortE: Tedit;
    lblPorttoport: Tlabel;
    MaxPortE: Tedit;
    StatusL: Tlabel;
    ActivityLB: TlistBox;
    StartBtn: Tbutton;
    Wssocket: TclientSocket;
    StopBtn: Tbutton;
    OpenOnlyCB: TcheckBox;
    Procedure StartBtnClick(Sender: Tobject);
    Procedure WssocketConnect(Sender: Tobject; Socket: TcustomWinSocket);
    Procedure WssocketError(Sender: Tobject; Socket: TcustomWinSocket;
      ErrorEvent: TerrorEvent; Var ErrorCode: Integer);
    Procedure StopBtnClick(Sender: Tobject);
    Procedure FormCreate(Sender: Tobject);
  Private
    { Private declarations }
    PortX, MaxPort:Integer;
    IsRunning:Boolean;
    Procedure SetStuffOnOff(Const St:Boolean);
  Public
    { Public declarations }
  End;

Var
  MainForm: TmainForm;
Implementation

{$R *.dfm}

Procedure TmainForm.SetStuffOnOff(Const St:Boolean); // cette procedure permet de lancer la
Begin                                               // connexion des socket
  IsRunning:=St;
  StopBtn.Enabled:=St;
```

```
  StartBtn.Enabled:=Not St;
  If Not (St) Then
  Begin
    ActivityLB.Items.Add('Done Scanning ' + IPAddressE.text);
    StatusL.Caption:='Status:'
  End
End;

Procedure TmainForm.StartBtnClick(Sender: Tobject); // bouton permetant le demarrage du scan
Begin
  ActivityLB.Items.Clear;
  PortX := StrToInt(MinPortE.text);
  MaxPort := StrToInt(MaxPortE.text);

  wsSocket.Address := IPAddressE.text;
  wsSocket.Port := PortX;
  wsSocket.Active := True;

  SetStuffOnOff(True);
  ActivityLB.Items.Add('Beginning scan: ' + IPAddressE.text)
End;

Procedure TmainForm.WssocketConnect(Sender: Tobject;
  Socket: TcustomWinSocket);
Begin

  ActivityLB.Items.Add('PORT: ' + inttostr(PortX) + '; OPEN!'); // si il y connexion du socket
                                                               // donc le port est ouvert
                                                               //affichage port+etat

wsSocket.Active := False;                                      //fermer le socket
  PortX := PortX + 1;                                          // passage au ports suivant
  wsSocket.Port := PortX;                                      //Connecter le port au socket
  StatusL.Caption:='Scanning port:['+IntToStr(PortX)+']';

  If (IsRunning) Then
    If (PortX > MaxPort) Then
      SetStuffOnOff(False)
    Else
      wsSocket.Active := True //test the new port
End;

Procedure TmainForm.WssocketError(Sender: Tobject; Socket: TcustomWinSocket;
  ErrorEvent: TerrorEvent; Var ErrorCode: Integer);
Begin
                                                               //si il n'y a pas de connexion
  ErrorCode:=0;                                                // donc le port est ouvert
  If Not (OpenOnlyCB.Checked) Then
  ActivityLB.Items.Add('Port: ' + inttostr(PortX) + '; Closed.');  //affichage port+etat


  wsSocket.Active := False;                                     //fermer le socket
  PortX := PortX + 1;                                           // passage au ports suivant
  wsSocket.Port := PortX;                                       //Connecter le port au socket
  StatusL.Caption:='Scanning port:['+IntToStr(PortX)+']';

  If (IsRunning) Then
    If (PortX > MaxPort) Then
      SetStuffOnOff(False)
    Else
      wsSocket.Active := True //test the new port
End;

Procedure TmainForm.StopBtnClick(Sender: Tobject);          //bouton permetant l'arret du scan
Begin
  SetStuffOnOff(False);
  wssocket.Active := False;
  ActivityLB.Items.Add('Stoped scan; port ' + inttostr(PortX) + '!')
End;

Procedure TmainForm.FormCreate(Sender: Tobject);
Begin
  IsRunning := False
End;
End.
```

2.3.3. Interface utilisateur

2.3.3.a. Menu Principal :

Figure 23. Capture d'écran « Menu principal »

C'est la première fenêtre qui s'affiche par default en exécutant l'application, elle contient les liens vers tous les modules développés ainsi que vers un fichier d'aide et une fenêtre qui affiche différentes informations sur cette application.

Le menu principal affiche aussi des informations sur l'état du firewall et divers autres informations sur le système.

2.3.3.b. Filtrage de paquets :

Figure 24. Capture d'écran « Filtrage de paquets »

Ce module est celui du filtrage de paquets, il permet de définir des règles de filtrage en suivant le processus ci-dessous :

1- Remplissage des champs.
2- Choix du type filtrage « Blocage ou Autorisation ».
3- Choix du protocole sur lequel on veut effectuer le filtrage.

Après avoir bien configuré une règle on doit appuyer sur le bouton « Ajouter règle » pour que la règle soit effective.

Les règles crées sont regroupées dans un tableau, pouvant être ainsi visualisées.

Pour supprimer une règle on doit d'abord la sélectionner sur le tableau et ensuite appuyer sur le bouton « Supprimer règle ».

2.3.3.c. Blocage de ports :

Ce module est devisé en trois parties (Blocage d'un seul port, Blocage du Ping et Blocage d'une plage de ports).

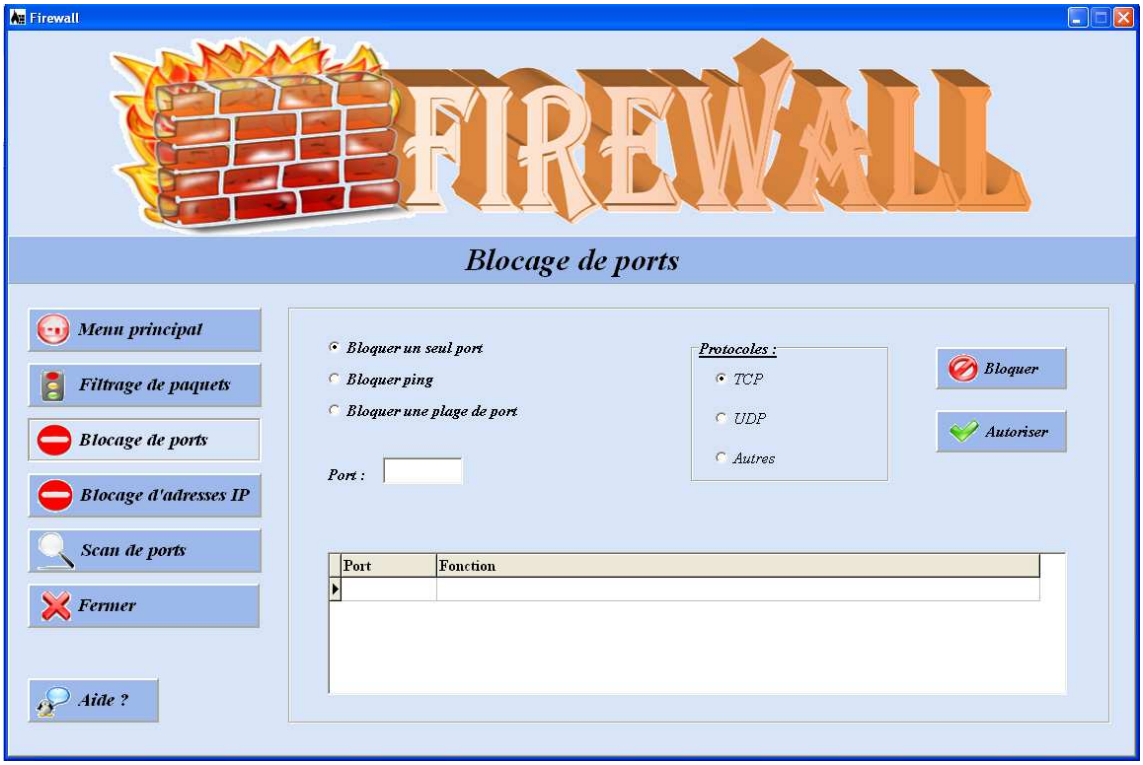

Figure 25. Capture d'écran « Blocage d'un seul port »

Figure 26. Capture d'écran « Blocage du Ping »

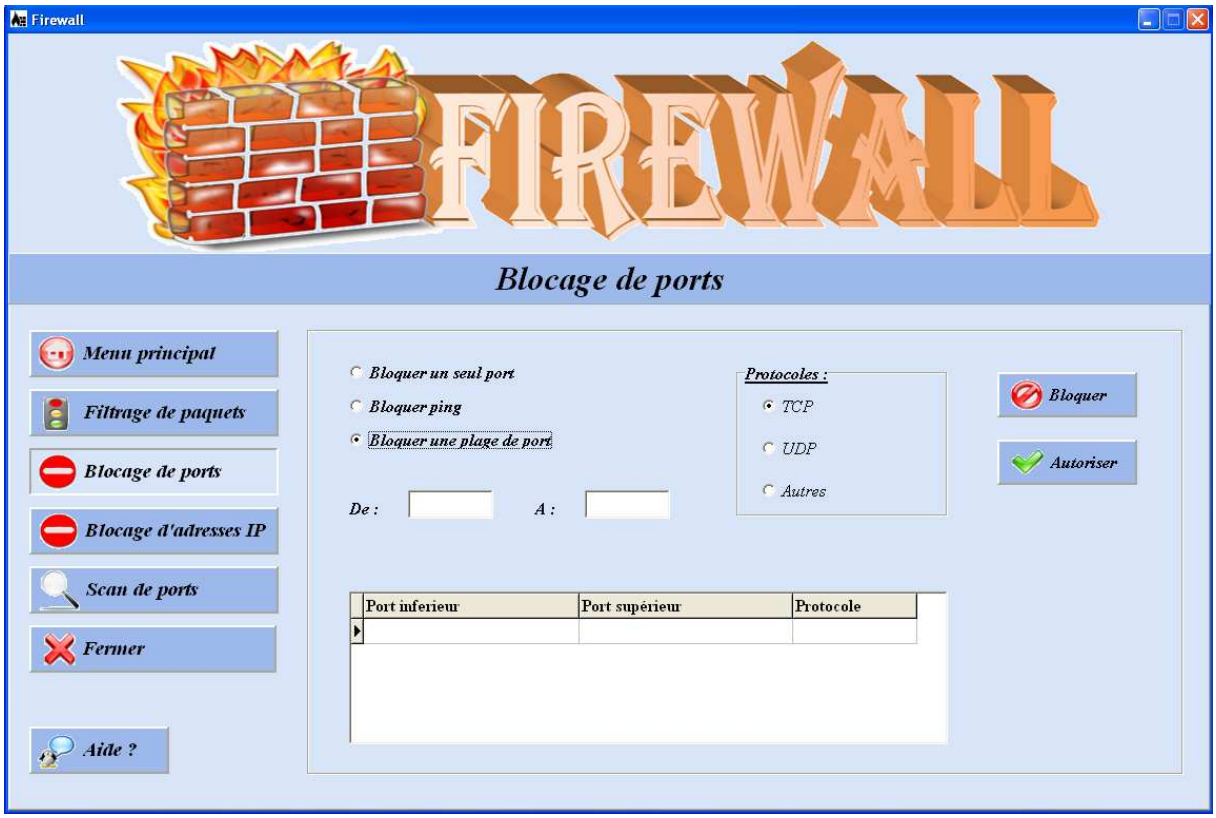

Figure 27. Capture d'écran « Blocage d'une plage de ports »

Ce module permet de bloquer des ports, le blocage se fait en suivant le processus ci-dessous :

1- Choix du type de blocage.
2- Remplissage des champs.
3- Choix du protocole sur lequel on veut effectuer le blocage.

Après avoir bien remplis les champs on doit appuyer sur le bouton « Bloquer » pour que le blocage soit effectif.

Les ports bloqués sont regroupés dans un tableau, pouvant être ainsi visualisés.

Pour autoriser un port ou une plage de ports on doit d'abord sélectionner l'enregistrement sur le tableau et ensuite appuyer sur le bouton « Autoriser ».

2.3.3.d. Blocage d'adresses IP :

Ce module est devisé en deux parties (Blocage d'une seule adresse IP et Blocage d'une plage d'adresses IP).

Figure 28. Capture d'écran « Blocage d'une seul adresses IP »

Figure 29. Capture d'écran « Blocage d'une plage d'adresses IP »

Ce module permet de bloquer l'adresse IP d'une machine distante, le blocage se fait en suivant le processus ci-dessous :

1- Choix du type de blocage.
2- Remplissage des champs.
3- Choix du protocole sur lequel on veut effectuer le filtrage.

Après avoir bien remplis les champs on doit appuyer sur le bouton « Bloquer » pour que le blocage soit effectif.

Les adresses bloquées sont regroupées dans un tableau, pouvant être ainsi visualisées.

Pour autoriser une adresse ou une plage d'adresses on doit d'abord sélectionner l'enregistrement sur le tableau et ensuite appuyer sur le bouton « Autoriser ».

2.3.3.e. Scan de port :

Figure 30. Capture d'écran « Scan de ports »

Ce module permet de faire un balayage des ports et nous indiquer l'état de chaque port scanné, pour se faire il faut préciser un intervalle de ports à scanner puis appuyer sur le bouton « Démarrer ». On peut aussi afficher que les ports ouvert en cochant l'option « Scanner que les ports ouverts ».

2.4. Intégration et tests globaux :

Cette phase correspond au regroupement progressif de tous les modules de façon à garantir la vérification et la validation progressive du logiciel, jusqu'à pouvoir le faire fonctionner dans son environnement réel.

2.5. Installation :

Cette phase correspond a la mise en fonctionnement du logiciel.

2.6. Maintenance :

Elle a pour objectif d'assurer le fonctionnement correct.

3. Conclusion :

Dans ce chapitre nous avons étudié et analysé d'abord les besoins et les objectifs du notre système, ce qui nous a mené à tracer les grandes lignes à suivre pour concevoir et implémenter notre application.

Suivant ces indications nous avons pu concevoir une application qui répond aux besoins exposés au début de notre travail.

Nous sommes parvenu a réalisé une application permettant de faire :

- ✓ Un filtrage de paquets.
- ✓ Un blocage de port (ou plage de ports).
- ✓ Un blocage d'adresses IP (ou une plage d'adresses IP).
- ✓ Un scan de port.

Le chapitre suivant sera consacré aux phases des tests globaux et installation.

Chapitre 4
Tests par simulation d'attaque

1. Introduction :

On a vu dans le chapitre précédant la conception et la réalisation de notre firewall, et comme tout autre logiciel, on doit valider ses fonctionnalités en établissant une série de tests, pour cela on a choisit d'utiliser un cheval de Troie, car ce malware est très répondu dans le monde des hackers.

Il existe plusieurs chevaux de Troie, parmi eux on cite : Propat, Back orifice, Subseven, Netbus…., dans notre cas on a choisit de tester notre firewall avec Back orifice car ce dernier offre beaucoup de fonctionnalités configurable a l'aide des plugins.

2. Back orifice (Bo2K) :

Back orifice est un outil d'administration à distance disponible sur l'environnement de Microsoft. Il s'agit d'une application Client/Serveur qui permet au logiciel client de surveiller, d'administrer et d'effectuer à distance un certain nombre d'action.

Il est créé et distribué par un groupe de hackers, **Cult of the Dead Cow (CDC),** en août 1998 dans le but d'améliorer les capacités d'administration a distance des systèmes Windows et de mettre en évidence les trous de sécurité existants. L'auteur principal de *Back Orifice* est « Sir Dystic »; et celui de *BO2K* est « DilDog ».

3. Installation de BO2K :

3.1. Le serveur :

Il suffit d'exécuter le fichier « BO2K.EXE ».Lors de sa première exécution, le « BO2K.EXE » procède aux deux opérations suivantes :

- **Auto-renommage de « BO2K.EXE » en « EXE » :** Le nom du fichier se réduit à un espace, suivi de l'extension habituelle « exe » des exécutables !, ce qui le rend invisible dans le gestionnaire de tâches. En effet, une commande DOS telle que « dir*.exe » peut afficher le nom court de ce fichier qui est alors « exe~1 » (sans extension).
- **Modification de la clef suivante de la base de registres :** HKEY_LOCAL_MACHINE\SOFTXARE\Microsoft\Windows\CurrentVersion\runServices. La valeur par défaut de cette clef est vide en temps normal, affecter la chaîne « .exe » si le serveur « BO2K » est lancé.Les fonctionnalités du serveur « Back Orifice » peuvent être enrichies par l'adjonction de plugins téléchargeables ou programmables en utilisant « BO2KCFG.EXE », la **figure 31** montre que BO2KCFG est composé de trois zones principales :

- La première où l'on spécifie le fichier du serveur qu'on va configurer ;
- La deuxième où on définie les extensions qu'on va utiliser plus tard et qui seront rajoutées à l'exécutable ;
- La dernière zone concerne les paramètres de chaque fonctionnalité, y compris les extensions qu'on vient d'ajouter. Ici on peut voir que l'option de connexion TCP a été choisie et que le port à utiliser est le **54320**.

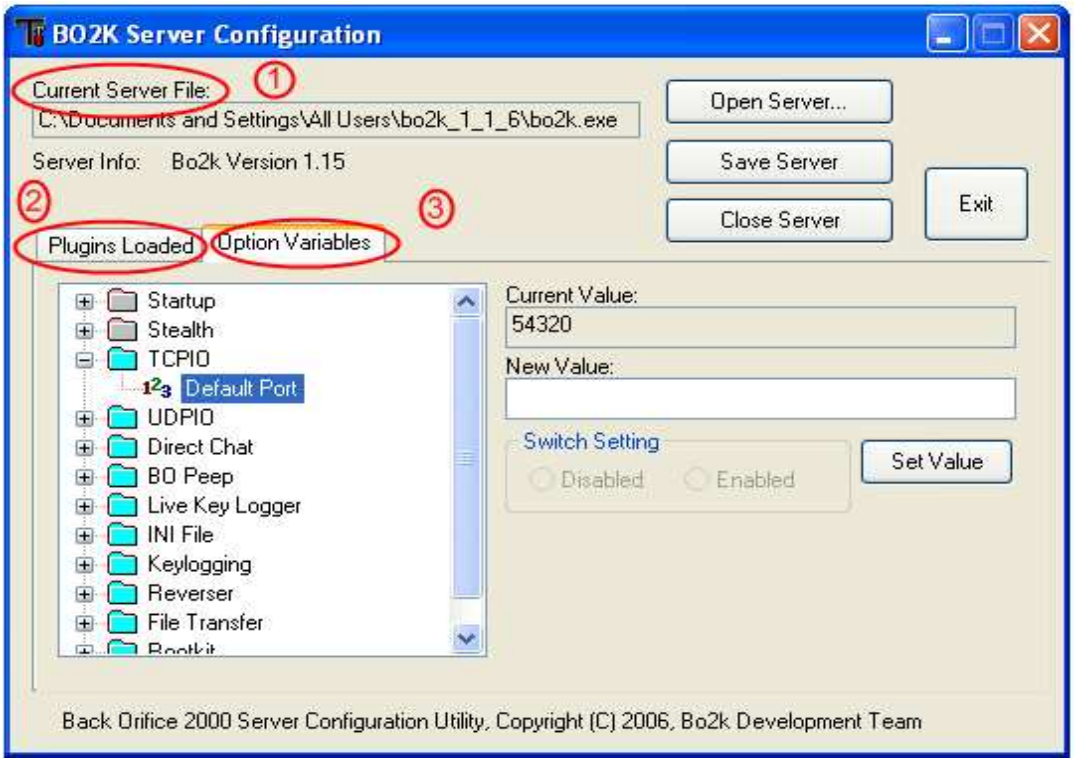

Figure 31. Capture d'écran « BO2KCGF »

Pour le moment, il existe plusieurs plugins, par exemple :

- Bopeep.dll : permet d'avoir accès en streaming (vidéo en temps réel) à l'écran hôte ;
- Botool.dll : permet de télécharger, ajouter, modifier ou supprimer des données ;
- Serpent.dll : permet de crypter de façon sécurisée la communication entre l'ordinateur hôte et l'ordinateur client.

3.2. Le client :

Il suffit d'exécuter « BO2KGUI.EXE » et de configurer le client avec les mêmes paramètres que le serveur, l'interface du « BO2KGUI.EXE » (**figure 32**) est composée de deux zones importantes : dans la première zone on trouve la liste des plugins chargés, qui sont représentés par des nœuds, chaque nœud contient plusieurs commandes, et on utilise le bouton « send command » pour envoyer la commande sélectionnée, et le résultat sera affiché dans la deuxième zone.

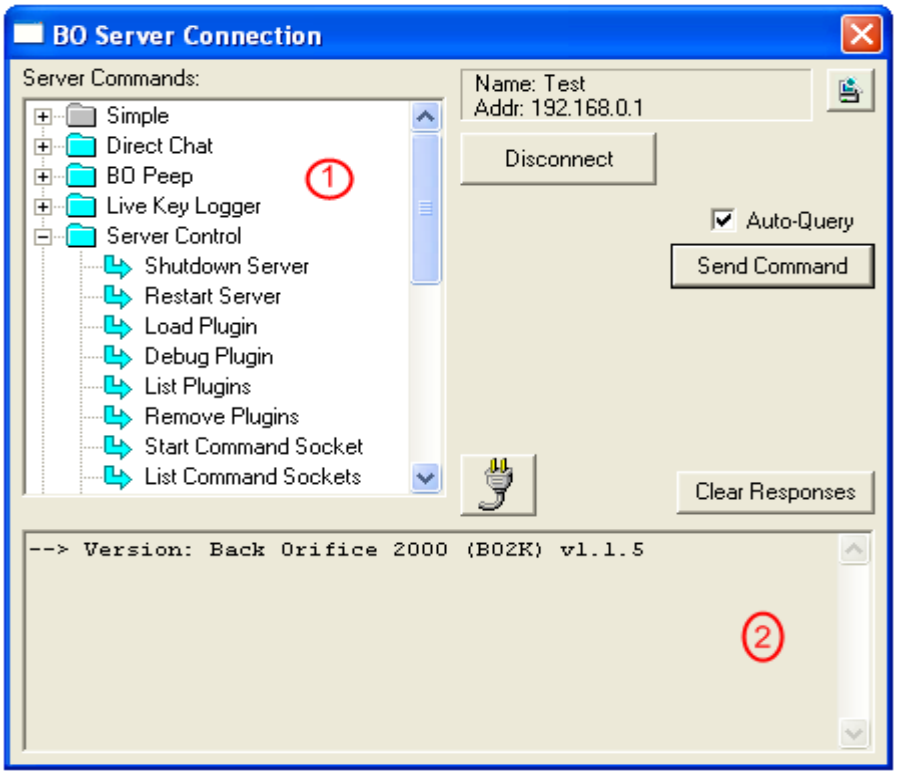

Figure 32. Capture d'écran « BO2KGUI »

Les fonctionnalités de « BO2K » disponibles (de base) comprennent :

- Commandes d'administration : reboot(redémarrer), lockup (arrêter), gestion de processus, registres… ;
- Récupération d'informations système ;
- Enregistrement de frappes clavier ;
- Récupération d'informations : fichier, mots de passe,… ;
- Gestion de fichiers : copie, effacement,… ;
- Supporte http pour navigation dans le système de fichiers (chargements possibles) ;
- Redirection de connections TCP/IP ;

- Affichage message à l'écran ;
- Mise à jour à distance, ainsi que installation/désinstallation ;
- Connections serveurs multiples et connections de plusieurs clients possibles ;
- Supports multimédia, capture audio/vidéo, lecteur audio.

4. Tests :

4.1. Plates-formes des tests effectués :

Plate-forme cliente :

PC sous Windows XP Professionnel SP2 (Pentium IV 3.00 GHz, 1 Go de RAM).

Plate-forme serveur :

PC sous Windows XP Professionnel SP2 (Pentium IV 3.00 GHz, 1 Go de RAM).

4.2. Plan de test :

Dans notre test on a procédé comme suit :

1. Effectuer un Ping ;
2. Scanner les ports de la machine victime ;
3. Exécution du serveur (BO2K.EXE) ;
4. Effectuer un deuxième scan de la machine victime ;
5. Réalisation des attaques ;
6. Exécuter le Firewall réalisé ;
7. Essai d'établissement d'une connexion avec le serveur « BO2K.EXE ».

4.2.1. Ping :

On a effectué un Ping pour vérifier l'existence de la machine victime sur le réseau.

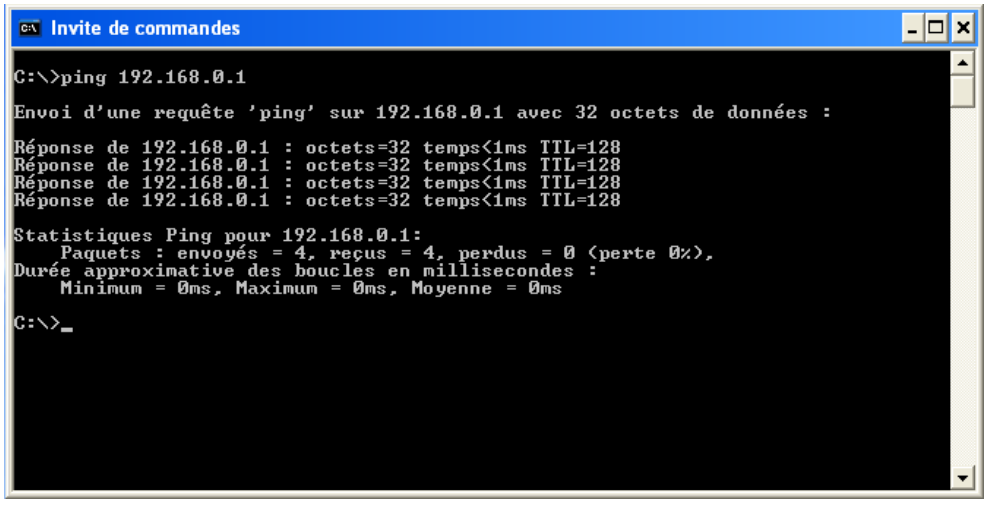

Figure 33. Vérification de l'existence de la machine victime

4.2.2. Scan ports :

Dans cette étape on va effectuer un balayage de ports ouverts sur la machine victime, avant l'exécution du serveur « BO2K.EXE », pour réaliser cette étape on a utilisé notre scanneur de ports (**figure 34**).

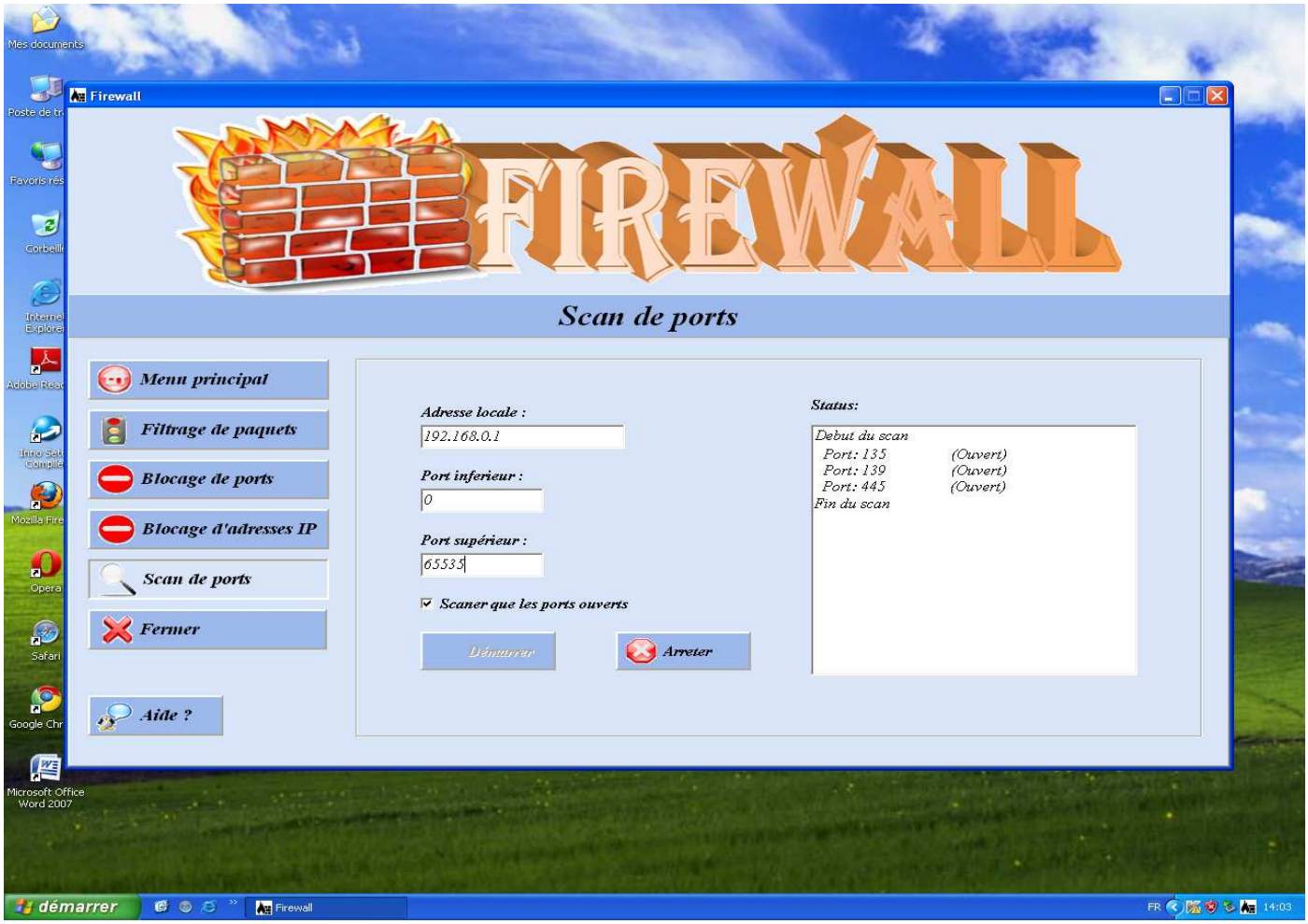

Figure 34. Ports ouverts sur la machine victime avant l'exécution de BO2K.EXE

4.2.3. Exécution du serveur :

Dans cette étape on va exécuter le « BO2K.EXE » sur la machine victime.

4.2.4. Scan de ports

Dans cette étape on va effectuer un deuxième balayage de ports ouverts sur la machine victime, d'après la (**figure 35**) on remarque l'ouverture du port « **54320** » qui était fermé

avant l'exécution du serveur, donc « BO2K.EXE » est en cours d'exécution sur la machine victime.

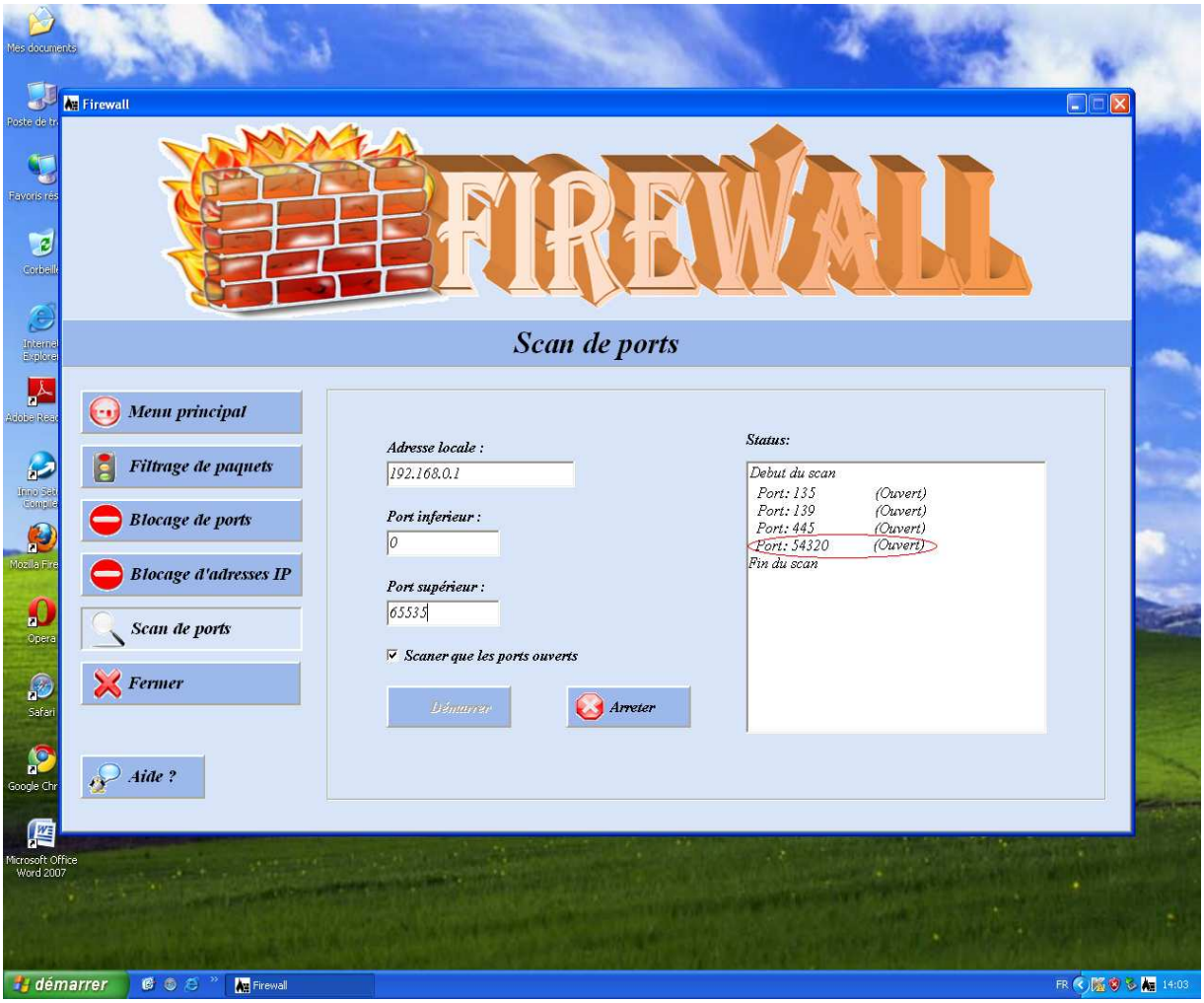

Figure 35. Ports ouvert sur la machine victime après l'exécution de BO2K.EXE

4.2.5. Réalisation des attaques :

Dans cette étape on va en premier temps établir la connexion du client avec son serveur, puis on lancera quelques commandes. Avant et après chaque commande on fera une capture d'écran pour voir l'impact de celle-ci sur la machine victime.

On a utilisé deux fonctionnalités du Back orifice à savoir le Chat et le Hijaking. Dans ce qui suit les captures d'écran des deux machines (victime et pirate) pour les deux fonctionnalités utilisées.

- Envois de message par le pirate a la machine victime, comme c'est illustré par les **figures 36 et 37.**

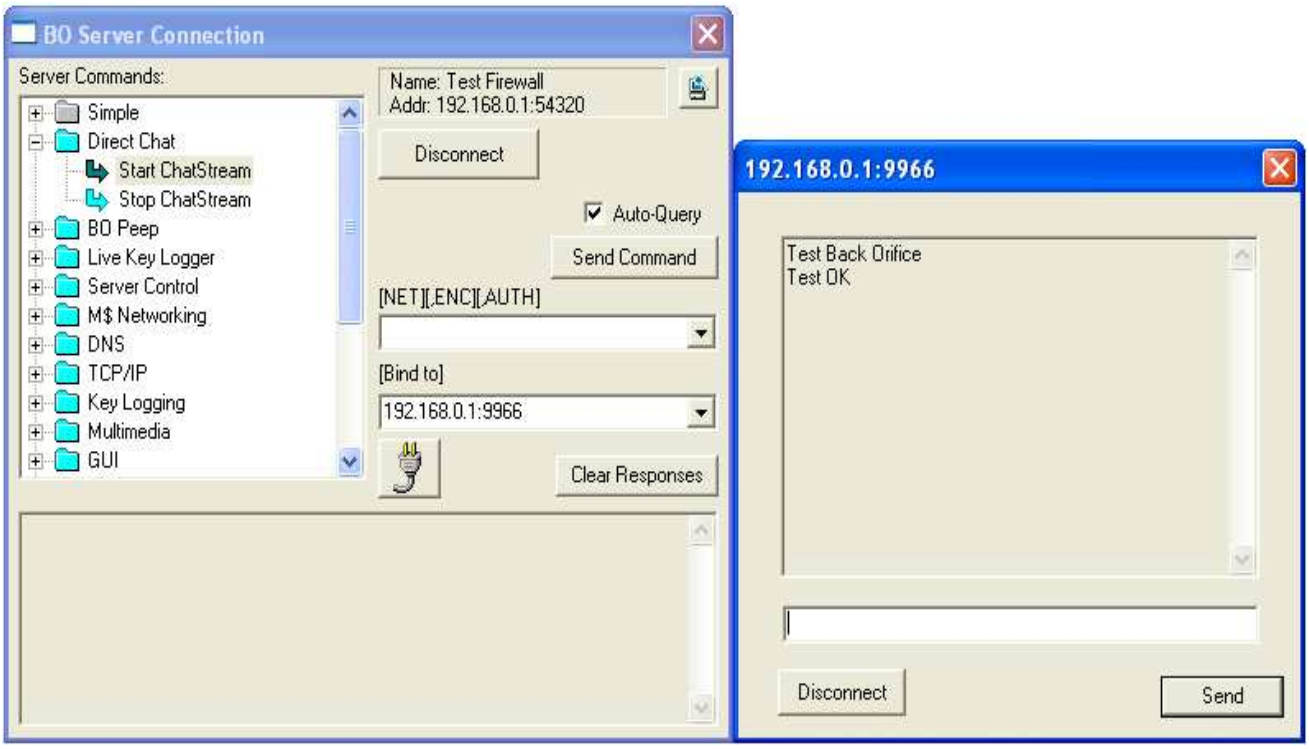

Figure 36. Contact par message (Machine pirate)

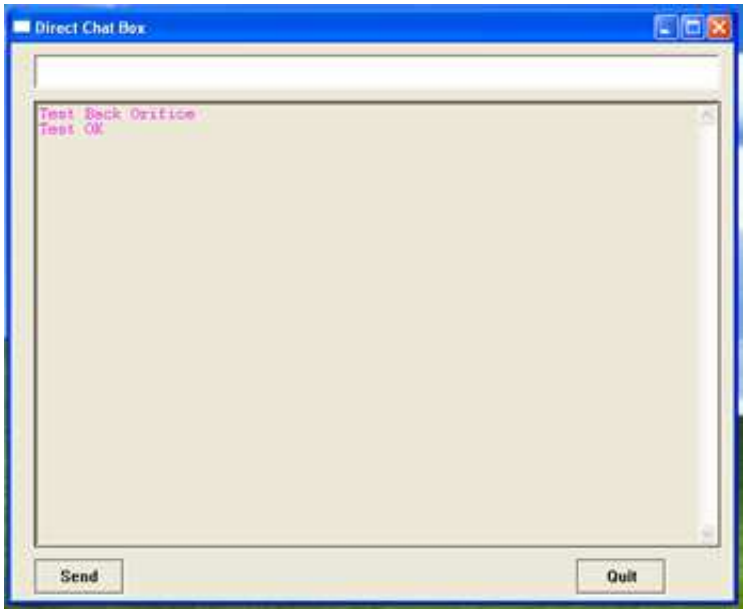

Figure 37. Contact par message (Machine victime)

- Vidéo Hijacking (visualisation d'écran de la machine victime) comme c'est illustré par les **figures 38 et 39**.

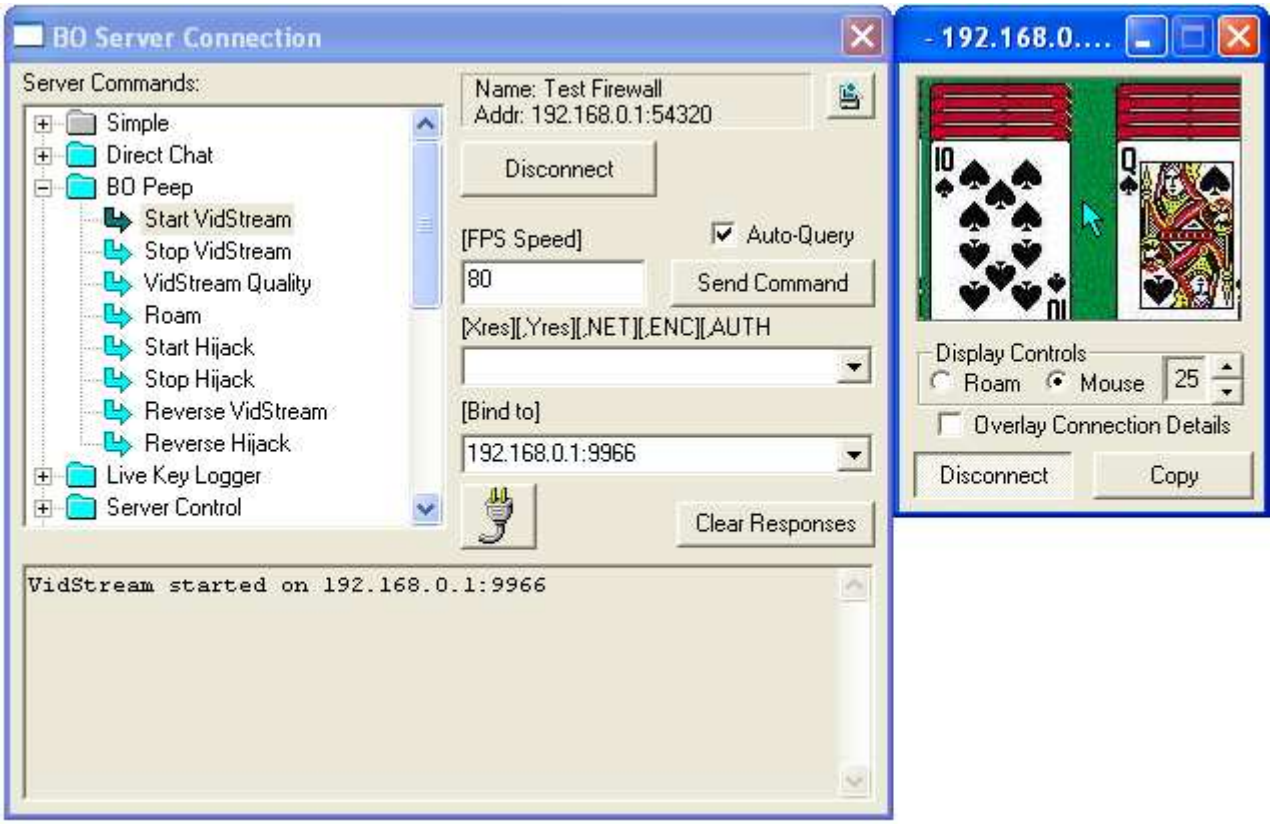

Figure 38. Vidéo hijaking (machine pirate)

Figure 39. Vidéo hijaking (machine victime)

4.2.6. Installation du Firewall :

Dans cette étape on va installer notre firewall, une icône apparaît dans la barre des taches, comme c'est illustré dans la figure suivante :

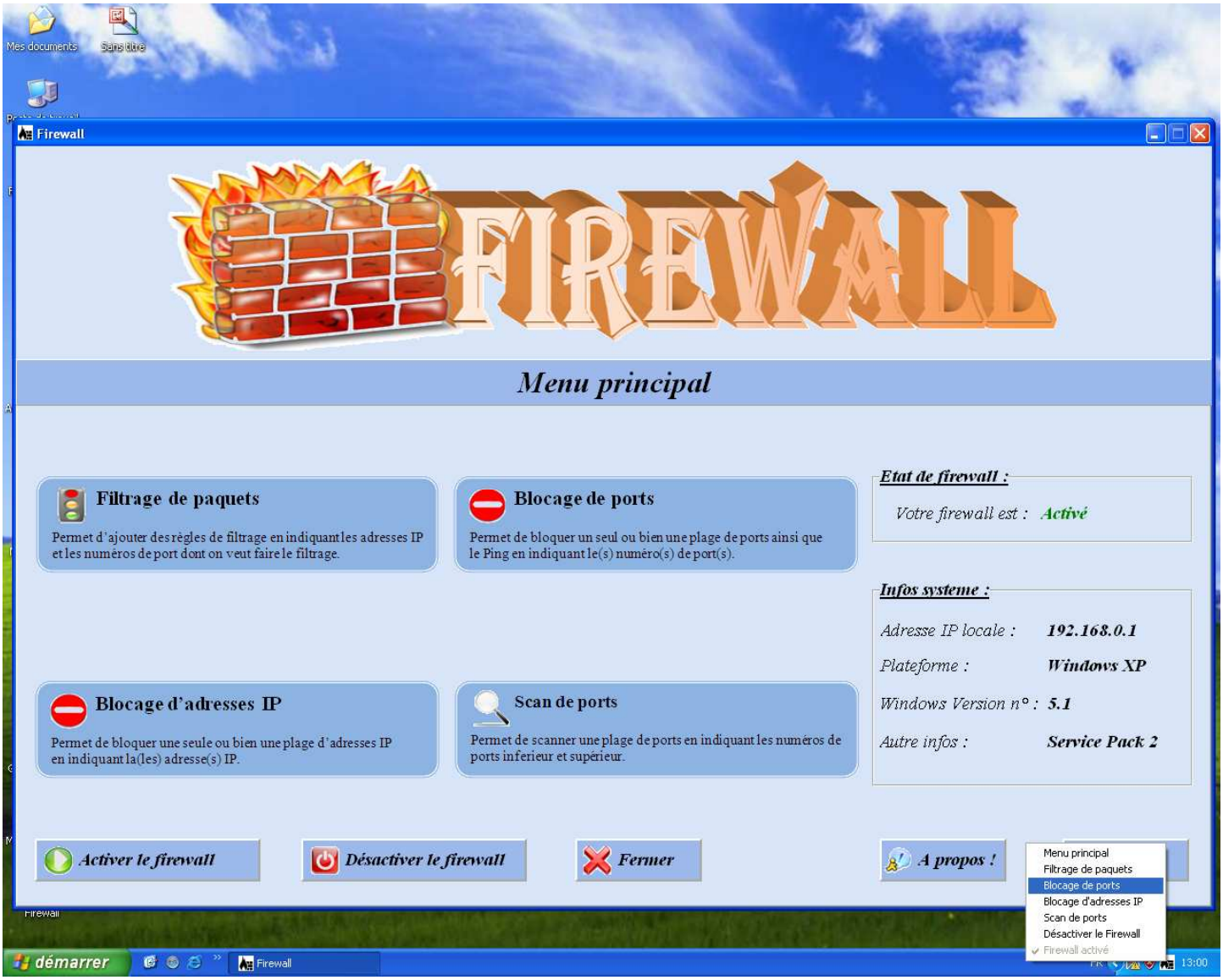

Figure 40. Installation du firewall

On va maintenant utiliser notre Firewall pour fermer le port « **54320** », afin d'empêcher la communication entre le client et le serveur « BO2K » (**figure 41**).

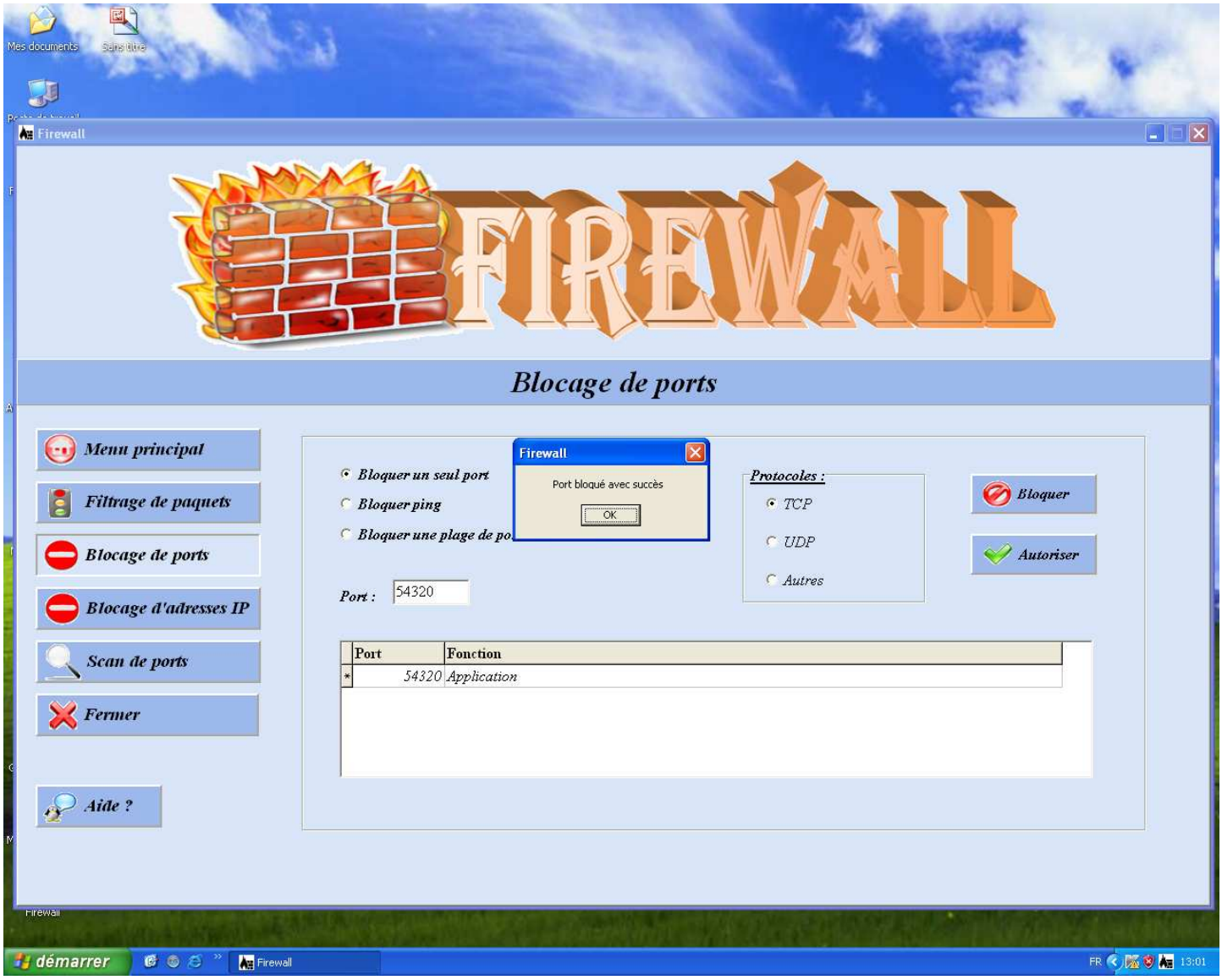

Figure 41. Blocage du port 54320

4.2.7. Etablissement d'une connexion avec le serveur :

Dans cette étape on essai d'établir une connexion avec le serveur, mais puisqu'on a fermé le port « 54320 » ; la connexion a échouée, comme montre la figure suivante :

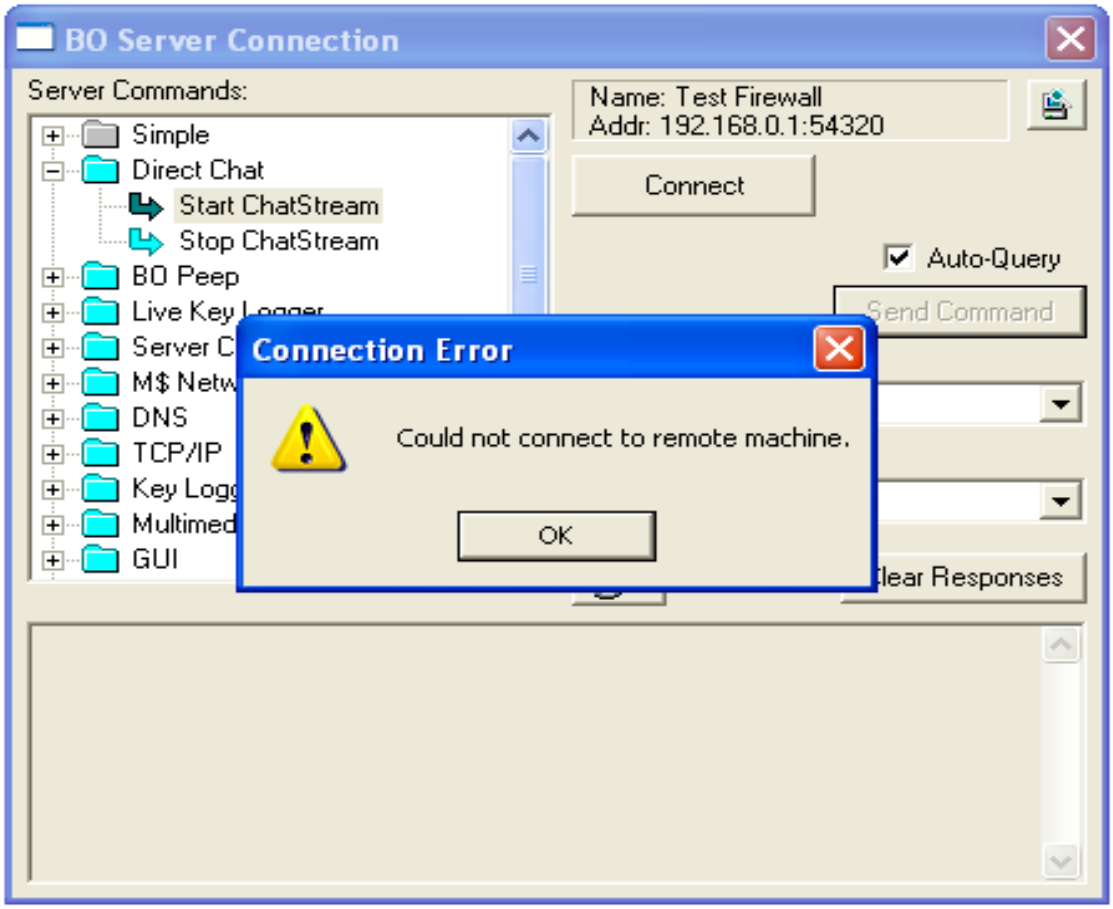

Figure 42. Echec d'établissement de la connexion avec le serveur

On a effectué par la suite un scan port à partir de la machine cliente et on a trouvé que le port « 54320 » est fermé comme montre la figure suivante :

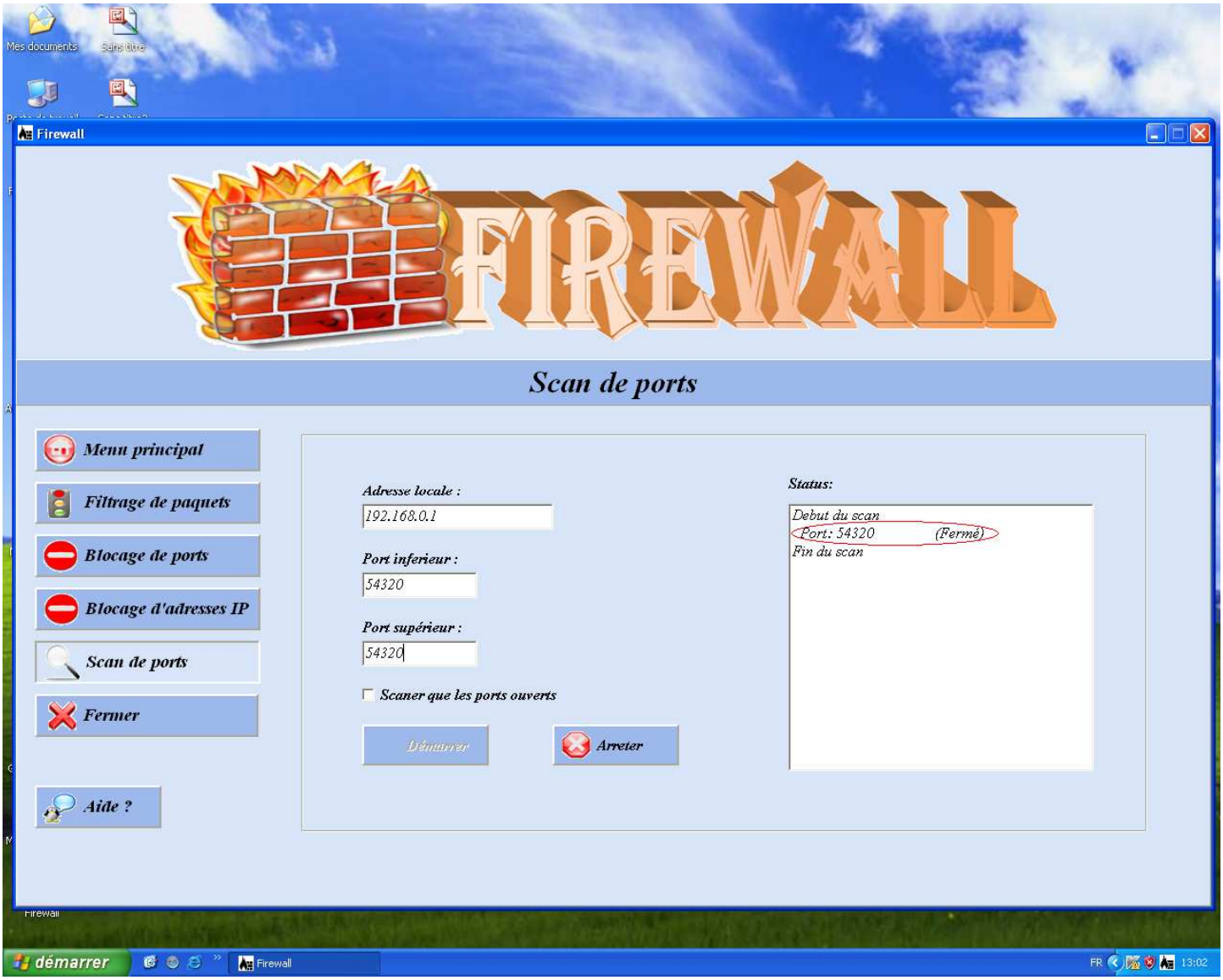

Figure 43. Port 54320 fermé

4. Conclusion

Back orifice est sans doute un logiciel remarquable par ses fonctionnalités, malgré ça la fermeture du port que le serveur utilise a empêché la communication avec son client, donc l'ouverture des ports constitue une faille de sécurité qu'il faut la gérer et surveillé afin de contrôler les communications qui déroulent entre les systèmes, le Firewall est le meilleur outil pour réaliser cet objectif. Donc les Firewalls couvrent une brèche importante de la sécurité informatique.

Conclusion Générale :

Il existe un très grand nombre d'attaques possibles sur les réseaux informatiques. Le firewall est un des éléments mis en place dans le cadre de la politique globale de sécurité définie. En effet, il est très utile (et très efficace si celui-ci est bien configuré) dans le domaine du contrôle des flux.

Les firewalls représentent uniquement un outil de sécurisation, ils ne peuvent en aucun cas rendre un système ultra-sécurisé en couvrant toute ces failles, pour cette raison et afin d'augmenter le degré de sécurité d'un système, il faut installer tous les dispositifs connus comme le antivirus, les firewalls, les IDS, les antispyware, ….. etc. ces derniers constituent les éléments essentiels d'une politique global de sécurité.

L'objectif de notre projet consisté a développé un firewall capable d'effectuer :

- ✓ un filtrage de paquets.
- ✓ un blocage de ports.
- ✓ un blocage d'adresses IP.
- ✓ un scan de ports.

Afin de valider notre application et vérifier son bon fonctionnement, nous avons effectué une série de tests tel que les attaques de type cheval de Troie (Back Orifice, Remote Administrater) mais aussi d'autres attaques de type déni de service (Ping of death). Nous avons réussi à rendre notre machine invisible au hacker en utilisant les différentes fonctionnalités de notre firewall.

L'objectif que nous nous sommes fixer a été atteint et ce modeste travail nous a permis de :

- ✓ Découvrir la sécurité informatique.
- ✓ Comprendre et mettre en œuvre le déroulement du cycle de vie d'un logiciel.
- ✓ Améliorer nos connaissances en programmation.
- ✓ Utiliser de nouveaux outils.

Finalement, cette application pourra être amélioré en lui ajoutant d'autres fonctionnalités tel qu'un détecteur d'intrusion (IDS), un antivirus,…etc.

Bibliographie

[1]. ACISSI, « Sécurité informatique - Ethical Hacking, Apprendre l'attaque pour mieux se défendre », edition Eni, 2011.

[2]. Jean-Francois Pillou, « Tout sur la securité informatique », edition DUNOD, 2005.

[3]. www.commentcamarche.net.

[4]. YKHLEF Hadjer, HAMZAOUI Nesrine, « Etude et développement d'un firewall pour la sécurité d'un réseau informatique », thèse de licence, USDB, 2008/2009.

[5]. Roulot, «Le piratage de A à Z », édition Edigo, 2010.

[6]. Xavier Tannier, « Se protéger sur Internet, Conseils pour la vie en ligne », édition Eyrolles, 2010.

[7]. Davis Chapman, « Firewalls – La sécurité sur Internet », edition O'Reilly, 1997.

[8]. http://www.frameip.com

[9]. http://www.developpez.net

Glossaire

A

ACK : Acquittement

L’acquittement d'une donnée ou d'une information consiste à informer son émetteur de sa bonne réception. On utilise souvent le terme ack pour un acquittement, ce terme correspond à l'équivalent anglais du terme : acknowledgement.

ACL : Access Control List

Access Control List est une liste des adresses et ports autorisés ou interdits par un pare-feu.

ARP : Address resolution protocol

L’Address resolution protocol est un protocole effectuant la traduction d’une adresse de protocole de couche réseau (typiquement une adresse IP) en une adresse MAC.

D

DMZ : Zone démilitarisée

Une zone démilitarisée est un sous-réseau isolé du réseau local par un pare-feu. Ce sous-réseau contient les machines étant susceptibles d'être accédées depuis Internet.

DOS : Attaque par déni de service

Une attaque par déni de service (denial of service attack, d'où l'abréviation DoS) est une attaque ayant pour but de rendre indisponible un service, d'empêcher les utilisateurs légitimes d'un service de l'utiliser.

F

FTP : File Transfer Protocol (protocole de transfert de fichiers)

Le File Transfer Protocol (protocole de transfert de fichiers), ou FTP, est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau, d'alimenter un site web, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur.

H

HTTP : L'HyperText Transfer Protocol (protocole de transfert hypertexte)

L'HyperText Transfer Protocol, est un protocole de communication client-serveur développé pour le World Wide Web. HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée.

I

ICMP : Internet Control Message Protocol

Internet Control Message Protocol est l'un des protocoles fondamentaux constituant la suite de protocoles Internet. Il est utilisé pour véhiculer des messages de contrôle et d'erreur pour cette suite de protocoles, par exemple lorsqu'un service ou un hôte est inaccessible.

IDS : Système de détection d'intrusion

Un système de détection d'intrusion est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions.

IP : Internet Protocol

Internet Protocol (abrégé en IP) est une famille de protocoles de communication de réseau informatique conçus pour et utilisés par Internet.

IPSec : Internet Protocol Security

Internet Protocol Security est un protocole utilisant des algorithmes permettant le transport de données sécurisées sur un réseau IP.

M

MAC : Media Access Control address

Une adresse MAC est un identifiant physique stocké dans une carte réseau ou une interface réseau similaire et utilisé pour attribuer mondialement une adresse unique au niveau de la couche de liaison

R

RTC : Réseau Téléphonique Commuté

Le Réseau Téléphonique Commuté est le réseau du téléphone (fixe et mobile), dans lequel un poste d'abonné est relié à un central téléphonique par une paire de fils.

S

SYN : Demande de SYNchronisation ou établissement de connexion

T

TCP : Transmission Control Protocol (protocole de contrôle de transmissions)

Transmission Control Protocol est un protocole de transport fiable, en mode connecté, les applications transmettent des flux de données sur une connexion réseau, et TCP découpe le flux d'octets en segments.

U

UDP : User Datagram Protocol (protocole de datagramme utilisateur)

User Datagram Protocol est un des principaux protocoles de télécommunication utilisés par Internet. Le rôle de ce protocole est de permettre la transmission de données de manière très simple entre deux entités, chacune étant définie par une adresse IP et un numéro de port.

V

VPN : Virtual Private Network (réseau privé virtuel)

Le réseau privé virtuel, fait référence l’usage du Protocol IPSec afin de crée un canal de communication sécurisé a usage privé, a travers un réseau public non sécurisé.

Annexe

DataWizard's SocketBlock ActiveX Firewall SDK for Windows 2000

Overview

Welcome to SocketBlock!

SocketBlock is a high performance ActiveX COM DLL for Windows 2000 and later that allows developers to get low level firewall functionality without having to write low level code. Developers can incorporate SocketBlock into their existing applications for added security, or create stand alone firewall applications.

Regardless of the type of network application, all applications are susceptible to attacks, whether it be on by curious co-workers on an intranet, or serious hackers on the internet. SocketBlock helps secure your application as well as the host machine by blocking incoming and outgoing packets according to filters that the developer sets up.

The basic steps to using SocketBlock are the same regardless of your environment:

* Declare and instantiate your SocketBlock object variable, as well as a sink variable if needed.
* Call the Activate method to initialize the component.
* Call the Start method to start packet filtering.
* Use the AddFilter and DeleteFilter methods to add/delete filters at will.

Company Information

DataWizard Technologies, Inc.
27 Hurlingham Cres.
North York, Ontario
Canada
M3B 2P9
Phone: (416) 385-9741
Email: sales@datawizard.net
Web: www.datawizard.net

Technical Support

Registered users can contact technical support via email at:

Support: socketblock@datawizard.net

Licensing Requirements

The component is licensed on a CPU basis, and is not Royalty Free. For each developer that will be developing concurrently with SocketBlock you must purchase one license. Each license allows one developer to develop with the component, as well as deploy the component

a dual processor machine (or to two separate machines), you must purchase two licenses. Similarly, if you have two developers developing but are deploying to one CPU, you must still purchase two licenses. Corporate and Site Licenses are available. Please email sales@datawizard.net if you have any questions regarding pricing or licensing. Licenses can be purchased securely online directly from our web site at www.datawizard.net/Buy/buy.htm. Licensing terms and conditions are as per the License Agreement.

License Agreement

SOCKETBLOCK SOFTWARE DEVELOPMENT KIT

END-USER LICENSE AGREEMENT FOR DATAWIZARD TECHNOLOGIES' SOFTWARE

IMPORTANT-READ CAREFULLY: This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and DataWizard Technologies Inc. for the software product identified above , which includes computer software and may include associated media, printed materials and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. GRANT OF LICENSE. This EULA grants you the following limited, non-exclusive rights:

Software Product. You may install and use one (1) copy of the SOFTWARE PRODUCT to design, develop, and test your software application ("Application"), and then you may deploy to 1 machine (computer). This constitutes one (1) license.

Sample Code. You may modify any sample source code located in the SOFTWARE PRODUCT's "samples" directories ("Sample Code") if provided, to design, develop, and test your Application. You may also reproduce and distribute the Sample Code in object code form only along with any modifications you make to the Sample Code, provided that you comply with the Deployment Requirements described below. For purposes of this section, "modifications" shall mean enhancements to the functionality of the Sample Code.

Deployable Code. You may deploy the SOCKBLKX.DLL and SOCKBLKD.SYS files ("Deployable Code") to one machine (computer). You may not otherwise copy or redistribute this code. This SOFTWARE PRODUCT is not royalty free.

Deployment Requirements. You may deploy any Sample Code and/or Deployable Code to one machine (collectively "DEPLOYABLE COMPONENTS") as described above, provided that (a) you deploy the DEPLOYABLE COMPONENTS only in conjunction with, and as a part of, your Application; (b) your Application adds significant and primary functionality to the DEPLOYABLE COMPONENTS; (c) you do not permit redistribution of the DEPLOYABLE COMPONENTS; (e) any deployment of Deployable Code is only in conjunction with your Application and includes each and every file contained therein deployed as a single set. The SOCKBLKX.DLL and SOCKBLKD.SYS files may not be individually reproduced or distributed.; (f) you include a valid copyright notice on your Application; and (g) you agree to indemnify, hold harmless, and defend DataWizard Technologies and it's distributors from and against any claims or lawsuits, including attorneys' fees, that arise or result from the use or deployment of your Application (h) you do not use the same application names, filenames, or binary compilations as those that are deployed with the SOFTWARE PRODUCT (i) any Sample Code or Deployable Code, whether enhanced and/or modified, may only be deployed in compiled form.

DataWizard Technologies Inc. reserves all rights not expressly granted to you.

2. COPYRIGHT. All rights, title, and copyrights in and to the SOFTWARE PRODUCT (including, but not limited to, any names, images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT) and any copies of the SOFTWARE PRODUCT are owned by DataWizard Technologies Inc. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material, except that you may either (a) make one copy of the SOFTWARE PRODUCT solely for backup or archival purposes, or (b) install the SOFTWARE PRODUCT on a single computer, provided you keep

the original solely for backup or archival purposes. You may not copy printed materials (if any) accompanying the SOFTWARE PRODUCT.

3. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.

LIMITATIONS ON REVERSE-ENGINEERING, DECOMPILATION, AND DISSASSEMBLY. You may not reverse- engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

RENTAL. You may not rent, lease or lend the SOFTWARE PRODUCT.

SOFTWARE TRANSFER. You may permanently transfer all of your rights under this EULA, provided you retain no copies, you transfer all of the SOFTWARE PRODUCT (including all component parts, the media and printed materials, any upgrades, this EULA, and the recipient agrees to the terms of this EULA. If the SOFTWARE PRODUCT is an upgrade, any transfer must include all prior versions of the SOFTWARE PRODUCT.

RUN-TIME DEPLOYMENT. You may deploy the run-time modules of the Software to one (1) computer provided that: (a) you deploy the run-time modules only in conjunction with and as a part of your software product; (b) you include valid copyright notices on your software product; (c) you agree to indemnify, hold harmless, and defend DataWizard Technologies Inc. and its suppliers and distributors from and against any claims or lawsuits, including attorneys' fees, that arise or result from the use or deployment of your software product; and (d) you do not embed the run-time modules in a toolkit which allows users to build and use or distribute applications containing the run-time modules; (e) your Application adds significant and primary functionality to the DEPLOYABLE COMPONENTS. The "run-time modules" refers to the SOCKBLKX.DLL and SOCKBLKD.SYS files that are required for execution of your software program. The run-time modules are limited to run-time files and install files.

TERMINATION. Without prejudice to any other rights, DataWizard Technologies Inc. may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

4. EXPORT RESTRICTIONS. You agree that neither you nor your customers intend to or will, directly or indirectly, export or transmit (a) the SOFTWARE PRODUCT or related documentation and technical data, or (b) your Application as described in Section 1 of this EULA (or any part thereof), or process, or service that is the direct product of the SOFTWARE PRODUCT to any country to which such export or transmission is restricted by any applicable government regulation or statute, without the prior written consent, if required, by such governmental entity as may have jurisdiction over such export or transmission.

MISCELLANEOUS. If any provision of this Agreement is found to be unlawful, void or unenforceable, then that provision shall be severed from this Agreement and will not affect the validity and enforceability of any of the remaining provisions.

NO WARRANTIES. To the maximum extent permitted by applicable law, DataWizard Technologies Inc. expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT and any related documentation are provided "as is" without warranty of any kind, either express or implied, including, without limitation, the implied warranties of merchantability or fitness for a particular purpose. The entire risk arising out of use or performance of the SOFTWARE PRODUCT remains with you.

LIMITATION OF LIABILITY. DataWizard Technologies Inc.'s entire liability and your exclusive remedy under this EULA shall not exceed five dollars ($5.00 CDN).

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. To the maximum extent permitted by applicable law, in no event shall DataWizard Technologies Inc. or its suppliers or distributors be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of, or inability to use, this product, even if DataWizard Technologies Inc. has been advised of the possibility of such damages. Because some states/provinces/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

RIGHT OF PUBLICITY. You agree that DataWizard Technologies, Inc. is hereby granted the right to promote SOFTWARE PRODUCT and your use of it in it's online portfolio, it web site, its press kits, its press releases, and any other promotional materials.

SocketBlock is a trademark of DataWizard Technologies Inc.

System Requirements

Any system that will use SocketBlock, both for development and production, must have:

* Microsoft Windows 2000 or greater

* A minimum of Winsock 2 correctly installed and configured

* **Sockblkx.dll** present and registered on the target machine

* Have **Sockblkd.sys** present on the machine in either the applications working directory (where the EXE is), or the System32, or the System32\Drivers directory.

Distribution Requirements

SocketBlock requires the **Sockblkx.dll** to be distributed and registered on the user's system prior to use, as well as the **Sockblkd.sys** driver to be present in either the applications working directory (where the EXE is; cannot be a network drive), or in the System32\Drivers directory (**Sockblkd.sys** does not have to be registered). This is customarily done using a setup program to compile a setup application to do this for you (most languages include a setup utility, including MS VB, MS VC++). It can also be done manually using the **Regsvr32.exe** file, which is customarily found in the System32 directory. **Sockblkx.dll** and **Sockblkd.sys** were built with MS VC++ 6 using ATL 3.0, and as a result, does not require any additional DLLs to be distributed.

SocketBlock

SocketBlock Object

SocketBlock Object

Object Name:	SocketBlock
Description:	SocketBlock Class
File Name:	SOCKBLKX.DLL
Help File:	SOCKBLKX.HLP
GUID:	{37511CED-908C-44AD-8D31-A6E1F4F01FA1}
Properties:	2
Events:	0
Methods:	5

Before you can use the SocketBlock object in your application, you must add the **Sockblkx.dll** file to your project. This may vary depending on the language/environment you are using.

To distribute applications you create with the SocketBlock object, you must install and register the **Sockblkx.dll** on the user's computer, as well as have the **Sockblkd.sys** file present in either the applications working directory (where the EXE is), or in the System32\Drivers directory. Registering a file can be done manually with **RegSvr32.exe**, however it easier to have this done by an automated installation script (Visual Studio includes such a tool---see the Visual Studio documentation for more details).

SocketBlock Properties

IsAdmin

Checks whether or not the currently logged on user has admin privilege.

Syntax

SockBlock.IsAdmin

Remarks

Returns a VARIANT_BOOL/Boolean indicating whether or not the current user has admin privilege. This level of privilege is required to use this component.

Return Data Type

VARIANT_BOOL/Boolean

LocalHostIPAliases

Returns a Variant containing an array holding all host IPs and IP aliases.

Syntax

SockLite.LocalHostIPAliases

Remarks

Returns a variant containing an array with all host IPs for the local host. Use this property to determine which interfaces you can block/allow packets on.

Return Data Type

Variant (Contains an array of Strings/BSTR)

SocketBlock Methods

Activate

Activates the component. This method must be called before any other methods or properties are called.

Syntax

SockBlock.Activate

Remarks

This method can fail with any of the SBK_GENERAL_ERRORS for license check failure (incorrect or missing license), OS platform version failure (not Windows 2000 or greater),

SBK_ERROR_INSTANCE_ALREADY_RUNNING if an instance of the component is already running (only one instance can be running at a time), or SBK_ERROR_NO_ADMIN_PRIVILEGE if the current user does not have Admin privilege. Note that any method called without first calling this method to initialize the component will generate a run-time error. This method should only ever be called once in the life of an application.

Return Data Type

Long

AddFilter

Adds a packet filter based on the criteria in the parameters passed. May be called either before or after the Start method.

Syntax

SockBlock.AddFilter SBK_FILTER_TYPE *FilterType*, String/BSTR *DstAddrLowRange*, String/BSTR *DstAddrHighRange*, Long *DstPortLowRange*, Long *DstPortHighRange*, String/BSTR *SrcAddrLowRange*, String/BSTR *SrcAddrHighRange*, Long *SrcPortLowRange*, Long *SrcPortHighRange*, SBK_IP_PROTOCOLS *Protocol*

Remarks

Use this method to set up which packets will be blocked, and which will be let through. It is crucial to understand two important points about how filtering works. The first is that criteria within one filter record (i.e. the parameters contained in one call to this method) are evaluated using AND logic: the packet must meet all of the criteria in this rule for the rule to take effect (whether it is blocking or allowing). The second is that filters are evaluated in the order they are added in (if no rules have been added, the packet filter lets all packets through). The following illustrates the logic used to accomplish a typical server scenario where we want to block all incoming TCP packets, except on port 110 (we will assume this machine is a POP3 server, and allows users to check their mail from the internet). Assuming the server's IP is **207.46.197.100**:

-Start method is called: all packet are being let through

-AddFilter method is called with the following params:

Parameter	**Value**	**Comments**
FilterType	SBK_BLOCK	Blocks packet that meet this filters criteria
DstAddrLowRange	207.46.197.100	Catch packets with a destination IP that falls in the range of DstAddrLowRange to
DstAddrHighRange	207.46.197.100	DstAddrHighRange.
DstPortLowRange	0	Catch packets with a destination port that falls in the range of
SrcAddrLowRange	0.0.0.0	Catch packets with a source IP that

		falls in the range of SrcAddrLowRange to
SrcAddrHighRange	255.255.255.255	SrcAddrHighRange.
SrcPortLowRange	0	Catch packets with a source port that falls in the range of SrcPortLowRange to
SrcPortHighRange	65535	SrcPortHighRange
Protocol	SBK_IPPROTO_TCP	Packet must of TCP type

At this point, all TCP packets going to the IP **207.46.197.100** from any address on any port are blocked. Next we want to add another filter that will override this rule, if it meets the criteria set in the filters rule, as follows:

Parameter	**Value**	**Comments**
FilterType	SBK_ALLOW	Allow packet that meet this filters criteria
DstAddrLowRange	207.46.197.100	Catch packets with a destination IP that falls in the range of DstAddrLowRange to
DstAddrHighRange	207.46.197.100	DstAddrHighRange.
DstPortLowRange	110	Catch packets with a destination port that falls in the range of DstPortLowRange to
DstPortHighRange	110	DstPortHighRange.
SrcAddrLowRange	0.0.0.0	Catch packets with a source IP that falls in the range of SrcAddrLowRange to
SrcAddrHighRange	255.255.255.255	SrcAddrHighRange.
SrcPortLowRange	0	Catch packets with a source port that falls in the range of SrcPortLowRange to
SrcPortHighRange	65535	SrcPortHighRange

Protocol	SBK_IPPROTO_TCP	Packet must of TCP type

This rule says to allow packets addressed to the IP address **207.46.197.100** from any address, with a (server) local port of 110, and any remote port on any port (0-65535), if the packet is

rule) is then changed to allow it through; if it does not, it will still be blocked as per the previous rule.

Note that rules are only cumulative *per protocol*. This means that if the above rule was added accidentally with the SBK_IPPROTO_UDP instead of SBK_IPPROTO_TCP, it would leave all TCP packets blocked, and perform the redundant task of letting all UDP packets in (because there was no rule blocking them in the first place).

Note that This method can fail with the SBK_ERROR_PARAM_NOT_VALID if any of the parameters are not valid, if the Activate method was not called (the component not initialized), or if the communication attempt with the driver failed. If this method succeeds, it returns a filter handle, which should be saved so that it can be deleted using the DeleteFilter method.

Filtering packets on any protocol other than TCP or UDP does not allow the packets to be filtered by port (many of them do not have a 'port' in the normal sense, so this info is ignored when provided in a filter), and can only be filtered by IP addresses.

To filter packets on all others protocols aside from TCP and UDP, a filter can be added using the SBK_IPPROTO_IP constant. This tells the filter to apply the filter to all other protocols aside from TCP and UDP, and is far more efficient than adding each other protocol as it's own filter. To filter all protocols, a filter would have to be added for each of TCP (SBK_IPPROTO_TCP), UDP (SBK_IPPROTO_UDP), and All Others (SBK_IPPROTO_IP).

Return Data Type

Long

DeleteFilter

Deletes a filter record. Use this method to remove a particular filtering rule.

Syntax

SockBlock.DeleteFilter Long *FilterHandle*

Remarks

Pass this method the Filter Key (Filter Handle) that was returned when the AddFilter was called. Note that a Filter Key can never be less than one. This method can fail if the Filter Key is invalid, or if there is an error communicating with the driver. On success this method returns zero.

Return Data Type

Long

Start

Starts packet filtering services. Packets are not being filtered until this method is called.

Syntax

SockBlock.Start

Remarks

Use this method to start packet filtering services. This method should not be called if packet filtering has already been started (i.e. this method has already been called), unless it has been stopped using the Stop method. This method can fail if either packet filtering is already active (in which case a run-time error will be generated), or if an error occurs while attempting to communicate with the driver. Returns zero on success.

Return Data Type

Long

Stop

Stops packet filtering services. No packets are being filtered after this method is called.

Syntax

SockBlock.Stop

Remarks

Use this method to stop packet filtering services. This method should not be called if packet filtering has already been stopped (i.e. this method has already been called), unless it has been started again using the Start method. This method can fail if either packet filtering is already inactive (in which case a run-time error will be generated), or if an error occurs while attempting to communicate with the driver. Returns zero on success.

Return Data Type

Long

Constants

SBK_GENERAL_ERRORS Constants

Value	Constant
-1	SBK_ERROR_PARAM_NOT_VALID
-2	SBK_ERROR_PLATFORM_VER
-3	SBK_ERROR_MEMALLOC_FAILURE
-4	SBK_ERROR_LICENSE
-5	SBK_ERROR_INSTANCE_ALREADY_RUNNING
-6	SBK_ERROR_NO_ADMIN_PRIVILEGE

SBK_IP_PROTOCOLS Constants

Value	Constant
0	SBK_IPPROTO_IP
1	SBK_IPPROTO_ICMP
2	SBK_IPPROTO_IGMP
3	SBK_IPPROTO_GGP
6	SBK_IPPROTO_TCP
12	SBK_IPPROTO_PUP
17	SBK_IPPROTO_UDP
22	SBK_IPPROTO_IDP
77	SBK_IPPROTO_ND
255	SBK_IPPROTO_RAW

SBK_FILTER_TYPE Constant

Value	Constant
0	SBK_ALLOW
1	SBK_BLOCK