DEVELOPMENT OF A DISTRIBUTED FIREWALL ADMINISTRATION TOOL

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Yunus ERDOĞAN

November 2008
İZMİR

We approve the thesis of **Yunus ERDOĞAN**

Assist. Prof. Dr. Tuğkan TUĞLULAR
Supervisor

Assist. Prof. Dr. Gökhan DALKILIÇ
Committee Member

Assist. Prof. Dr. Tolga AYAV
Committee Member

16 December 2008

Prof. Dr. Sıtkı AYTAÇ
Head of the Computer Engineering
of Department

Prof. Dr. Hasan BÖKE
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Assist. Prof. Dr. Tuğkan TUĞLULAR, for his guidance, patience and encouragement. He provided me with many helpful suggestions and important advice during this thesis work.

I would also like to thank Özgür Kaya, helping me when writing this thesis using MATLAB.

Finally, I would like to thank my parents, Ayşe and Erkan Baykal Erdoğan, my sister, Deniz Erdoğan, and my wife Selcen Erdoğan, for their consideration and understanding.

ABSTRACT

DEVELOPMENT OF A DISTRIBUTED FIREWALL ADMINISTRATION TOOL

Today firewalls not only guard internal computer networks but also individual personal computers against malicious and unauthorized accesses from outside. The purpose of this study is to create architecture and its corresponding application to manage distributed firewalls running on Microsoft Windows platform. Distributed Firewall Administration is about creating a management center for a network composed of the firewalls running on Microsoft Windows platform. Main important part of this work is to determine distributed firewall network topology with breadth-first search and depth-first search algorithms.

The Microsoft Windows Firewall API makes it possible to programmatically manage the features of firewalls running on windows platform by allowing applications to create, enable and disable firewall exceptions. This study used the Windows Firewall API to manage the features of it. This API is only reachable using C/C++ low level programming languages.

Distributed Firewall Administration Tool (DFAT) can add, modify or delete rules on the end-user firewall rule set, these rules stored on the database. This tool works on a distributed environment, there is a parent child relationship between firewalls. Parent firewalls have right to manage its child firewall's rule set. Firewalls introduce themselves to each other with broadcast method.

ÖZET

DAĞITIK GÜVENLİK DUVARLARI İÇİN BİR YÖNETİM ARACI GELİŞTİRİLMESİ

Güvenlik duvarları günümüzde sadece yerel bilgisayar ağları için koruma değil aynı zamanda kişisel bilgisayarlar için de dışarıdan bilgisayarlara izinsiz girişleri engellemektedir. Bu çalışmanın amacı Microsoft Windows platformunda çalışan dağıtık güvenlik duvarlarını yönetebilecek uygun bir mimari oluşturmak ve bu mimariye uygun bir uygulama geliştirmektir. Dağıtık güvenlik duvarı yönetim aracı yerel ağlar için Microsoft Windows platformunda çalışan güvenlik duvarlarını yöneten bir yönetim merkezi olarak düşünülmüştür. Bu çalışmanın en önemli noktası dağıtık güvenlik duvarları ağ yapısını derinlik öncelikli arama ve yayılma öncelikli arama algoritmaları ile oluşturmaktır.

Windows güvenlik duvarı yazılım programlama ara yüzü kullanılarak Microsoft Windows platformunda çalışan güvenlik duvarlarına ait özellikler yönetilebilir. Hazırlanan yazılım ile güvenlik duvarı kural dışı listesi oluşturulabilir, liste aktif yada pasif hale getirilebilir.Yazılım programlama ara yüzüne sadece C/C++ gibi alt düzey diller kullanılarak ulaşılabilmektedir.

Dağıtık güvenlik duvarı yönetim aracı, güvenlik duvarı kurallarını yönetir anılan kurallar veritabanında saklanmaktadır. Bu araç dağıtık ortamda çalışmaktadır, güvenlik duvarları arasında anne çocuk ilişkisi bulunmaktadır. Anne güvenlik duvarları çocuklarının kurallarına müdahale etme hakkına sahiptir. Güvenlik duvarları ağ içerisinde birbirlerini yayın yapma metodu ile tanımaya çalışırlar.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The Internet is a highly effective tool for communicating and gathering information and for cooperation between distant locations. It can be considered as a kind of global meeting place where people from all parts of the world can come together. Uploading, downloading, and sharing document files are now everyday activities. Unfortunately, when people perform these daily tasks, they expose their computers to serious security risks.

Network security is very important to avoid security risks. There are several techniques and tools that are used to provide network security. Network security aims are confidentiality, integrity and availability of data. Firewalls are one of the well-known tools that are very helpful in network security to provide above mentioned security tools. Basically, a firewall separates the private network from the external and non-secure environment. A firewall that is located between the private network and Internet enforces a predefined network security policy by controlling network traffic passing over it. A network security policy is composed of a set of rules that defines what types of connections are allowed between internal and external networks. Additionally, a security policy defines the action that should be performed when a violation of policy is detected.

To be sure that security policy is managed correctly; the whole traffic between internal and external networks must travel through the firewall. In other words, firewall should be at the entry point of the private network. Although the traditional firewalls have very critical roles in the network security, they have some drawbacks. There are several researches on the firewall technologies to overcome these drawbacks. One of the recent approaches in this field is the concept of distributed firewall. In this scheme, security policy is still centrally defined, but enforcement is left up to the individual endpoints.

The main concern of this study is to design a distributed firewall architecture and to develop a Distributed Firewall Administration Tool for Windows platforms. There are some previous works about this topic but in this thesis some new methods are implemented. Firewalls are discovering distributed firewall topology by using Breadth-First Search (BFS) and Depth First Search (DFS) algorithms. Moreover, personal firewall provides switching between domains according to the selected domain suitable rules loaded the firewall. Furthermore, parent firewalls can interfere to their child firewall's rule set.

This thesis introduces a Distributed Firewall Administration Tool for Windows environment to keep track of some certain events performed on policy rules. The thesis is composed of five chapters. Chapter 2 gives some background information about Internet firewalls, distributed firewalls, and windows firewalls. In addition, Windows Filter Hook Driver creation, initialization and implementation are described in this chapter. Chapter 3 explains the details of suggested distributed firewall architecture on which Distributed Firewall Administration Tool will run. Moreover, database design and software design also explained in this section. Chapter 4 describes experiments which are performed using Distributed Firewall Administration Toll and gives an evaluation of experiment results. Finally, Chapter 5 gives the conclusion of this thesis work.

CHAPTER 2

BACKGROUND

In this chapter, packet filtering firewalls and their disadvantages are explained. In addition, the idea of distributed firewalls is examined along with.

2.1. Packet Filtering Firewalls

Firewalls are network elements that have an important role for network security. Basically, a firewall is a secure and trusted machine that separates the private network and public network. The purpose of a firewall is to keep "bad" things outside a protected environment. Firewalls implement a security policy. Although the policy might be prepared to prevent any access from outside (while still allowing traffic to pass from the inside to the outside), it might be prepared to permit access only from certain places, from certain users, or for certain activities.

A firewall filters both inbound and outbound traffic (see Figure 2.1). Firewalls can filter packets based on their source and destination address and port numbers. This kind of filtering name is address filtering in addition they can also filter specific types of network traffic. This is known as protocol filtering; in this type of filtering the decision to forward or reject traffic depends upon the protocol used. Firewall can also filter traffic by packet attribute or state.

Packet filtering firewalls are the basic level of the firewalls. As the name suggest, this firewalls check for each and every IP packet individually, either coming in or going out of private network. According to the selected policies (called Rule set or Access Control Lists) it determines whether to accept a packet or reject it. Packet filters use one or more of the following pieces of information to make their decision on whether or not to forward the packet: source address; destination address; options in the network header; transport-level protocol (i.e., TCP, UDP, ICMP, etc.); flags in the transport header; options in the transport header; source port or equivalent if the protocol has such a construct; destination port or equivalent if the protocol has such a construct; the interface on which the packet was received or will be sent; and whether the packet is inbound or outbound (Wack, et al. 2002).

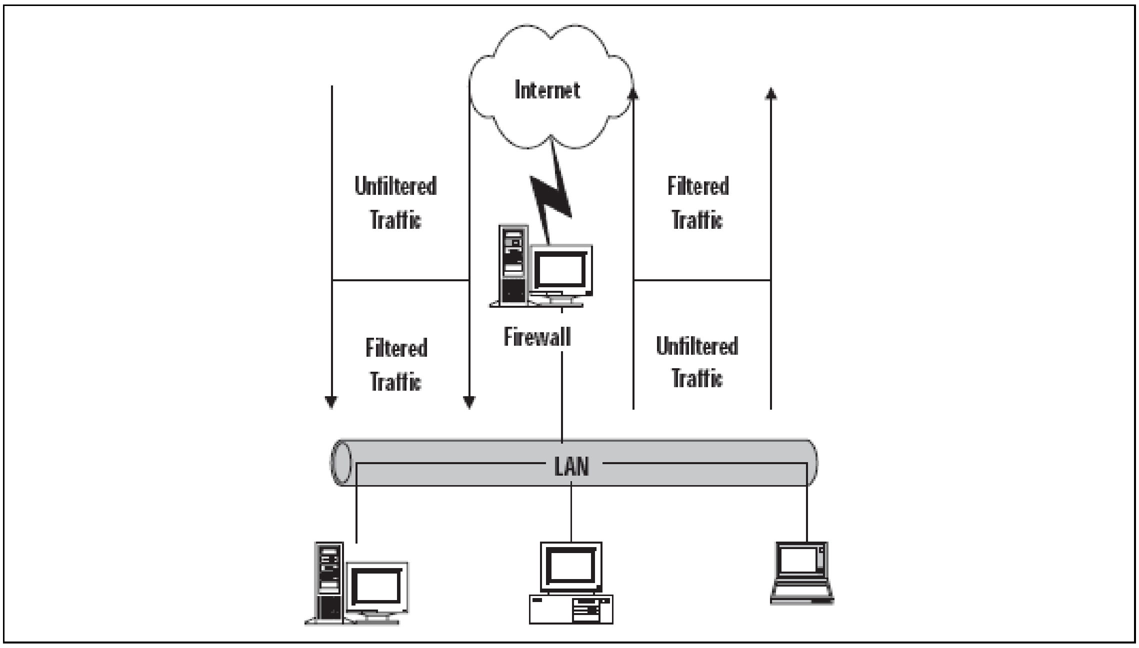

Figure 2.1. Firewalls filter both inbound and outbound Internet traffic
(Source: Chapman and Zwicky 1995)

A firewall does not protect whole network from every type of attack since it does not block all traffic but it will limit risk of significantly. Moreover, firewalls cannot provide protection against viruses or malicious code. Since most firewalls do not inspect the payload or content of the packet, they are not aware of any threat that may be contained inside (Kurose and Ross 2003).

The success of any firewall solution's implementation is directly related to the existence of a well-thought-out and consistently-implemented security policy. A network security policy defines the network’s expectations of proper network use and the procedures to prevent and respond to security incidents. Without a network security policy, a proper security framework cannot be established (Chapman and Zwicky 1995).

Drawbacks of Firewalls

Although they provide several benefits for network security, traditional firewalls, or packet filtering firewalls have some drawbacks as stated in (Bellovin 1999, Li 2000, Ioannidis et al. 2000, Gray 2003). These drawbacks can be summarized as follows.

They can be bottleneck to throughput, since firewalls must analyse every packet on network communication, they often decrease network performance. Moreover conventional firewalls are central management of the network security also they are central point for attack and if an intruder breaks through the firewall they may have unlimited access to the corporate network (Li 2000).

Conventional firewalls restricted access to desirable services. They block services that user wants to access for example telnet, ftp etc. Moreover they rely on topology restrictions; sometimes network topology may not fit the firewall design this kind of problem can be solved using insecure services across major gateways (Gray 2003).

The biggest disadvantage of firewalls is that they do not provide protection from insider attacks. Conventional firewalls assume that all insiders are trusted since most corporate computer crime is perpetrated by internal users, a firewall offers little protection against this threat (Gray 2003).

Another disadvantage of conventional firewalls is that they do not protect against back door attacks. For example, if unrestricted modem access is permitted into a site attacker could jump around the firewall or attacker routing the network traffic through backdoor intermediary computers such as proxy servers. Firewalls cannot stop these problems, since they must allow the common or diverted traffic (Gray 2003).

Firewalls cannot protect the network against data-driven attacks these attacks can be summarized as follows:

- Viruses: Firewalls does not offer virus protection
- Executable Content: Java applets, ActiveX Controls, JavaScript, VBScript
- Tunneling
- End to End Encryption

2.2. Distributed Firewalls

As a result of dramatic increase in network complexity and development of new technologies like wireless networks and VPNs, it is not easy to maintain a fixed network topology anymore. Additionally, there are increasing user demands like mobility, security, performance and reliability. As a result of these and the disadvantages mentioned above, conventional firewalls have started to become inadequate.

In order to remove such kind of problems, Bellovin and Ioannidis, et al. introduced the concept of distributed firewall. The distributed firewall design is based on the idea of enforcing the policy rules at the endpoints rather than a single entry point to network. The security policies are still defined centrally. The aim with this approach is to retain the advantages of firewalls while resolving the disadvantages mentioned above (Bellovin 1999,Ioannidis, et al. 2000).

In the architecture described by Bellovin, a firewall is placed at each host in the network and managed centrally, i.e. defined centrally and enforced in a distributed way. This way the domain of the firewall is no longer defined by the topology of the network and also the policies, being defined at a central location in terms of end nodes, don't suffer an impact form the change in topology.

Distributed firewall system is not a single system that acts as a gateway between two networks (see Figure 2.2). By removing this single choke point, the network bottleneck is eliminated. In distributed firewall technology, an access control mechanism is distributed to each host which requires protection. The unique access control for each of the network systems allows different levels of security to be implemented on computer system in the same network (Grant, et al. 2001).

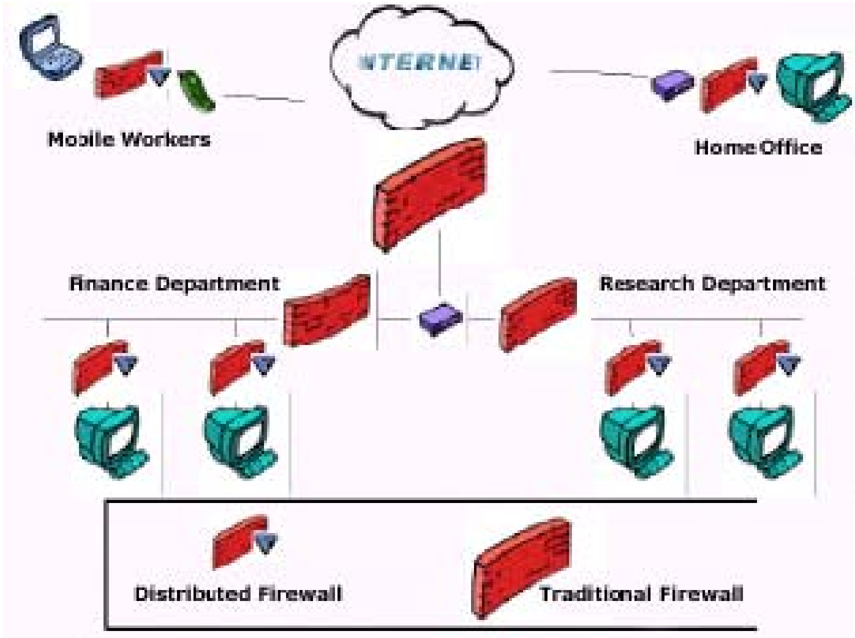

Figure 2.2. Distributed Firewall architecture.
(Source: Li 2000)

There are three components for distributed firewall environment. These components are policy language, policy distribution scheme and certificates. Policy language defines which inbound and outbound connections are allowed or rejected. Basically it is equivalent to packet filtering rules. Policy language should also support credentials, for delegation of rights and authentication purposes (Ioannidis, et al. 2000). While traditional firewalls usually use the IP address as an identifier, distributed firewalls use cryptographic certificates as identifier since distributed firewalls are independent of topology. Certificates enable making decisions without knowledge of the physical location of the host. Public-key cryptography mechanisms are most often applied in contemporary implementations (Ioannidis, et al. 2000).

The policy language defines which inbound and outbound connections on any component of the network policy domain are allowed. Policy distribution scheme is used to enable policy control from central point. Distribution scheme defines type of the security policy delegation to the members of the network (Stepanek 2001). Policies are distributed according to one of the following distribution scheme:

- Policies as well as credentials can be pushed to every single end point in the policy domain.

- Policies and credentials can be pulled from a trusted repository during initialization.
- Policies are pulled during initialization of the policy verifier whereas credentials for authentication mechanisms remain on a trusted repository and are requested whenever communication traffic is reaching a node from a yet unknown host (Stepanek 2001).

In distributed firewall architecture, preferred network level encryption mechanism for TCP/IP is IPSEC. IPSEC supports network-level peer authentication, data origin authentication, data integrity, and data confidentiality (encryption), and replay protection.

Advantages and Disadvantages of Distributed Firewalls

Distributed firewalls have some advantages and disadvantages. The advantages of distributed firewalls can be stated as follows: First of all, there is no longer a single chokepoint this is major benefits of both a performance and availability. Generally throughput is limited with the firewall speed in conventional firewall but in distributed firewall throughput is no longer limited by the speed of the firewall (Ioannidis, et al. 2000).

Another advantage of the distributed firewall is requiring less CPU usage than conventional firewalls for equal amount of traffic. Because all of the policies for all nodes splitting the time to process traffic over the network's node. Conventional firewalls cannot prevent inside attacks. With the distributed firewall architectures, it is possible to estimate the source of unfriendly traffic coming from Internet or local network (Ioannidis, et al. 2000).

Filtering of certain protocols such as FTP listening a specific port this operation that is generally not permitted through a traditional firewall but in distributed firewall, each node knows when it is listening for a particular data. Despite of some advantages of distributed firewalls which are described above also they have some disadvantages. If firewall command center is compromised, due to attack or mistake by the administrator, this situation is high risky for security of the entire network (Li 2000).

Topological independence is one of the main advantages of distributed firewalls. Since network security no longer depends on network topology, it provides more flexibility in defining the security perimeter. Security perimeter can easily be extended to cover remote hosts and networks whenever required (Ioannidis et al. 2000).

Distributed firewalls provide separate level of security for different kinds of data. For example financial, engineering and medical data need different level of security so using distributed firewall each node would have firewall and they can define different level of security. In addition they can provide separate level of security to the Web, Mail servers, Application servers or individual nodes in the setup. These are meant to provide higher security to the corporate networks (Li 2000).

In traditional firewalls there is an assumption on that insiders are trustable. However, according to the researches most of the attacks are originated by the insiders. With the distributed firewall architectures, the insiders are no longer treated as "unconditionally trusted". Dividing network into parts having different security level is much easier with distributed firewalls (Ioannidis, et al. 2000).

End-to-end encryption is possible without affecting the network security in distributed firewall systems.In conventional firewalls, the use of end-to-end encryption was causing some problems in network security. On the other hand, end-to-end encryption significantly improves the security of the distributed firewall.

On the other hand, there are some drawbacks of distributed firewalls that can be summarized as follows (Li 2000).

Compliance of security policy for insiders is one of the major issues of distributed firewalls. This problem especially occurs when each ending host have the right of changing security policy. There can be some techniques to make modifying policies harder but it is not totally impossible to prevent it.

Intrusion detection systems are less effective with distributed firewalls because complete network traffic is not on the single point. There is more than one entry point and each node has log file. It is possible to log suspicious connections on local server but these logs need to be collected and analyzed by security experts in central services.

2.3. Windows Firewall

Windows Firewall is a personal firewall which comes bundled with Microsoft's Windows XP, Windows Server 2003, Windows Vista operating systems. Windows Firewall is a stateful host firewall technology that inspects and filters all network traffic. Windows Firewall searches the state of each network connection and determines whether the unsolicited incoming traffic should be allowed or dropped. Not only does Windows Firewall check incoming information; it takes care of outgoing information as well. A firewall that works with a router program probes every network packet to decide whether to forward it toward its address.

Windows Firewall protects the valuable data stored on computers in following ways (Source: Microsoft TechNet 2008):

• Denying Remote Login.

• Preventing SMTP session hijacking.

• Scanning E-mail bombs that carry viruses and other malware.

• Not allowing suspicious Macros to run on PC.

• Keeping track of Virus, Spams and Trojans.

• Deleting / blocking risk ware.

2.3.1. Windows Firewall Architecture

Windows firewall architecture has several networking components shown in Figure 2.3. These components can be list as follows (Source: Microsoft TechNet 2008):

- Connection Sharing Service
- Network Address Translation (NAT) driver
- IPv4-based TCP/IP driver
- IPv6-based TCP/IP driver
- Windows Sockets driver (Winsock.dll)

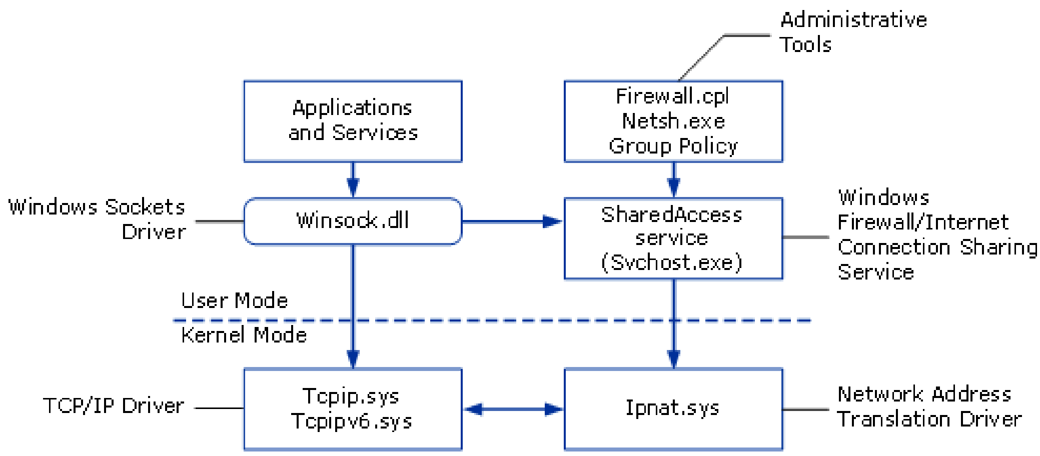

Figure 2.3. Windows Firewall Architecture
(Source: Microsoft TechNet 2008)

The NAT driver provides data storage for Windows firewall. NAT driver stores connection state information which has 5-tuple entries. These are the protocol, the source and destination port numbers, and the source and destination IP addresses. Windows firewall is stateful firewall because of storing data.

TCP/IP Driver controls the flow of information between a network adapter and system service. While incoming traffic flows through the TCP/IP driver, the traffic is inspected by the NAT driver. NAT driver compares the incoming traffic entries with the Windows Firewall exception list. If the traffic matches an exception, the NAT driver determines that the traffic is allowed otherwise dropped. Neither NAT and TCP/IP Driver sends a notification to the sender when packets are dropped.

The Winsock driver is responsible for assigning and binding ports to a program or system service. Programs and system services use this driver when they need to listen for incoming traffic.

2.3.2. Windows Firewall Interfaces

There are two interfaces related to Windows Firewall first one is firewall-hook interface another one is Windows Firewall application programming interfaces (APIs). The intent of a filter-hook interface is to manage network packets that were sent and received across a firewall in the context of the TCP/IP protocol (Source: Microsoft TechNet 2008). Windows Firewall Interfaces shown in Figure 2.4.

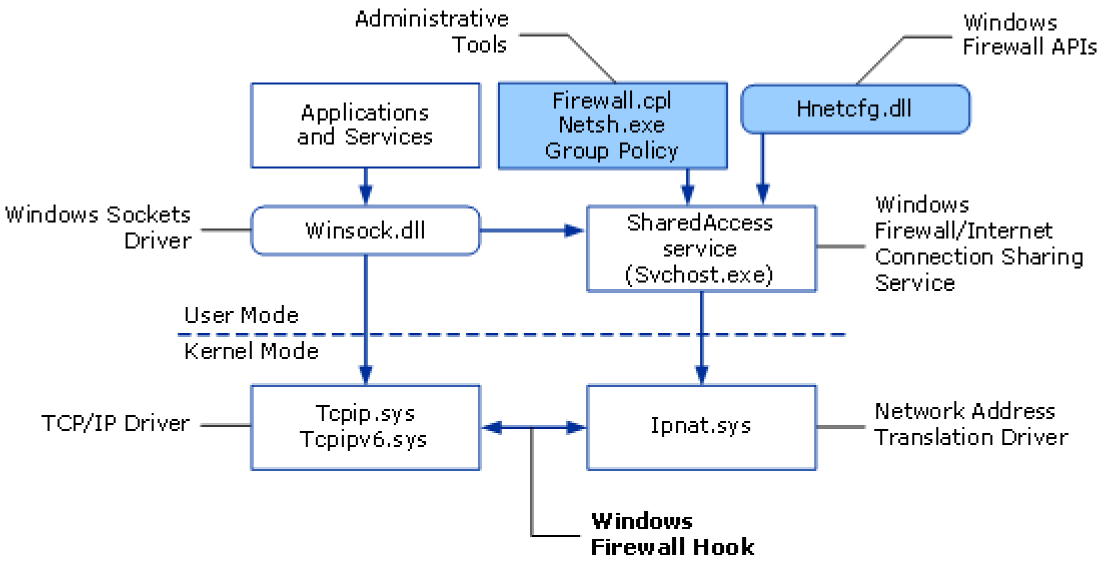

Figure 2.4. Windows Firewall Architecture Interfaces.
(Source: Microsoft TechNet 2008)

The Windows Firewall APIs provide a public interface for programmatically configuring Windows Firewall settings (Source: Microsoft TechNet 2008).

- Enable and disable Windows Firewall.
- Add and remove programs and system services from the exceptions list.
- Add and remove TCP and UDP ports from the exceptions list.
- Enable and disable preconfigured system service exceptions.
- Configure ICMP settings.
- Configure the log file.

There are some personal firewalls available on the commercial market place to provide the security for the computers, such as Comodo Firewall, Zone Alarm etc. These products use filter-hook drivers to custom solutions on Windows environment. In this thesis, distributed firewall administration tool will deal with filter-hook drivers to implement network security policy.

2.4. Windows Filter-Hook Drivers

A filter-hook driver is a kernel-mode driver that is used to filter network packets. It implements a callback function then it registers callback function with the system-supplied IP filter driver. Then IP filter driver uses the filter hook to determine how to process incoming and outgoing packets (Microsoft Developer Network 2008).

To supplement the Packet Filtering application programming interface (API), a filter-hook driver can be created to filter network packets. User-mode applications use this API to create and manage, in the system-supplied IP filter driver, input and output filters that filter packets with specific IP addresses or TCP/UDP port numbers. Since the Packet Filtering API optimizes the system-supplied. IP filter driver to process packets without the overhead that is associated with a filter-hook driver.

2.4.1. Creating a Filter Hook

A filter hook is created by implementing a function of type *PacketFilterExtensionPtr*. The *PacketFilterExtensionPtr* data type points to a filter-hook callback function (Microsoft Developer Network 2008). Aim of the function is to analyze the incoming packets whether to forward, drop or allows the IP Filter driver to further process the packets.

A filter hook performs the following actions on incoming and outgoing packets: Compares specific information with information that the IP filter driver passed to the filter hook to determine how packets should be further processed. After the filter hook inspects packets, returns with one of the following response codes that direct the IP filter driver:

- PF_FORWARD - Returns the forward response immediately to the IP stack. For local packets, IP forwards them up the stack. If the destination for packets is another computer and routing is enabled, IP routes them accordingly.

- PF_DROP - Returns the drop response immediately to the IP stack. IP should drop the packet.
- PF_PASS - Filters packets and return the resulting response to the IP stack. Continue to filter packets as defined by the Packet Filtering API. The filter hook returns this pass response if it determined that it should not process the packet but should allow the IP filter driver to filter the packet.

2.4.2. Initializing and Unloading the Filter-Hook Driver

The filter-hook driver's DriverEntry routine creates and initializes a device object for the driver object. The filter-hook driver's DriverEntry routine can also register the driver's filter hook with the IP filter driver.

The DriverEntry routine must export a routine that unloads the filter-hook driver. This unload routine removes the device that was created in DriverEntry but must not clear the previously registered filter hook.

2.4.3. Registering and Clearing a Filter Hook

A filter-hook driver registers its filter-hook callback function with the IP filter driver to inform the IP filter driver to call the hook callback for every IP packet that is received or transmitted. A filter-hook driver might also clear a previously registered hook callback. To register or clear a hook callback function, the filter-hook driver must first create an IRP using a pointer to the device object for the IP filter driver. The filter-hook driver then submits this IRP to the IP filter driver (Microsoft Developer Network 2008).

The filter-hook driver must clear its filter hook from the same entity with which it registered its filter hook; therefore, the filter-hook driver should store the pointers to the IP filter driver's file and device objects in global variables.

2.4.4. Implementing Filter Hook IOCTL

A filter-hook driver implements a dispatch routine so applications or higher-level drivers can send I/O control (IOCTL) requests down to the filter-hook driver to set up the filter hook. This dispatch routine processes these IOCTLs. The DriverEntry routine must export this dispatch routine (Microsoft Developer Network 2008).

When the IOCTL request retrieved a filter-hook driver's dispatch routine call the system function which is IoGetCurrentIrpStackLocation**.** In this call, the dispatch routine passes a pointer to the IRP that it received. Then dispatch routine determines which IOCTL request was received and processes the request accordingly.

After the current IOCTL request completes, dispatch routine calls the IoCompleteRequest system function and passes the status of the operation. This status is returned to the application or higher-level driver that made the request (Microsoft Developer Network 2008).

The following are typical IOCTLs that a filter-hook driver's device-control routine might process:

- Register-Hook IOCTL
- Registers a filter hook
- Clear-Hook IOCTL
- Clears the previously registered filter hook

2.4.5. Implementing Filter Hook Drivers

As with any kernel-mode driver, a filter-hook driver is considered as a trusted component and can crash the system if it executes an illegal instruction or references an invalid region of memory.

The integrity of data being sent over the network might be suspicious, or faulty hardware or other drivers might submit data in unexpected or improper formats. Therefore, a filter hook must validate the format of such data.

A filter hook must limit the amount of per-packet processing that it performs to reduce the overhead associated with each incoming and outgoing packet. A filter hook should be optimized to minimize the average time it spends processing packets.

A filter-hook driver is a kernel-mode driver that is used to filter network packets. Filter-hook drivers extend the functionality of the system-supplied Internet Protocol (IP) filter driver. A filter-hook driver can only be installed on the Microsoft Windows 2000 operating system and later versions (Microsoft Developer Network 2008).

Only a single filter-hook driver can be installed on the operating system and used by the IP filter driver. A filter-hook driver can only register itself with the IP filter driver if the pointer to the extension hook for the IP filter driver is set to NULL. After the filter-hook driver is registered, the IP filter driver assigns the file object for the filter-hook driver to the extension hook for the IP filter driver, thereby ensuring that it can only accept and use a single filter-hook driver (Microsoft Developer Network 2008).

When Network Packet is received in computer IP Driver passes the packet to the filter (callback) function and waits the return value if function returns “Allow Packet” packet will be allowed in contrast function returns “Drop Packet” IP Driver drops the packet. This situaton shown in Figure 2.5.

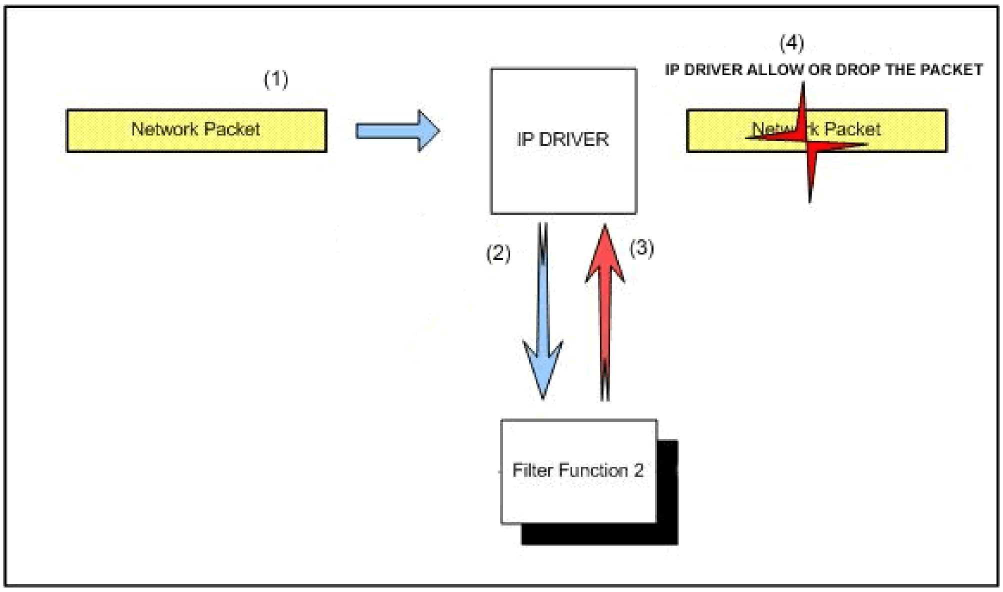

Figure 2.5. IPFilter Driver Working Diagram
(Source: CodeProject 2008)

CHAPTER 3

PROPOSED APPROACH

In this chapter, the distributed firewall administration architecture is explained along with the database design that each node in the architecture needs to maintain. Moreover, software design of client application and server application is described using UML notation.

3.1. Distributed Firewall Administration Architecture

Distributed Firewall Administration Architecture based on hierarchically organized distributed firewall system. As illustrated in Figure 3.1, it is a tree structure. The domain statement has a domain firewall which is standing on the domain entrance and protects the entire domain according to the organizational policy. Similarly, according to the network model there are subnets available and connected to the domain firewall. Each subnet has a subnet firewall which is located on the subnet entrance. Purpose of the subnet firewall is same as the domain firewall.

Every subnet may have different number of personal firewall; this personal firewall can control their network traffic. In addition subnet firewall may have child firewall which type can be subnet firewall. According to the description, there are three types of firewall available in the network domain model. Organization of the firewalls are based on the tree structure, domain firewall is the root node of the tree structure. Subnet firewalls are nodes between domain firewall which is root and personal firewalls which are leaves. Subnet firewalls are middle or intermediate nodes of the domain. Finally, personal firewalls, which are running inside the subnet, are leaf nodes located at the bottom of the tree.

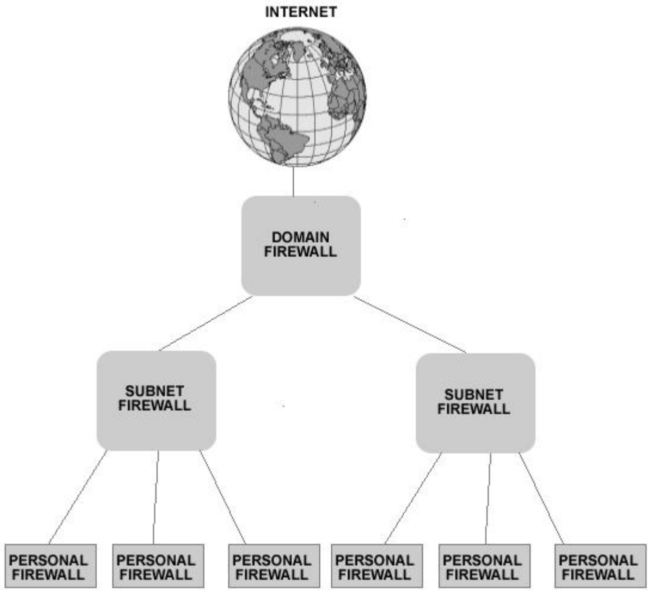

Figure 3.1. Distributed Firewall Administration Architecture.

At the beginning domain firewall does not know any information about distributed firewall architecture. Distributed firewall architecture defined by using two different algorithms such as Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms.

Communication scheme between these firewall nodes in the system as follows personal firewall nodes has to maintain local rule base to store rules. They are responsible to enforce the local policy. When personal firewall performs any operations such as insert, delete policy rule they have to propagate to their Subnet firewall. Subnet firewalls can communicate to all of the nodes inside that subnet but they cannot communicate to another subnet firewall at the same level. Similarly, a domain firewall can communicate to any other nodes in that domain. The communication between a domain firewall and leaf firewall is possible with the help of the subnet firewalls. Communication request of the domain firewall is received by the leaf level firewall via the subnet firewall.

3.2. Database Design

Each firewall node has to maintain a local rule base to store rules that will be used in the distributed firewall administration tool. Security policy rules are kept on the

Microsoft SQL(Structured Query Language) Server 2000. According to the architecture which is described in Section 3.1 firewall type can be Subnet or Personal firewall. Basic principles same for all types of firewall which is shown in Figure 3.2. In the following part each type of firewall database design will be examine.

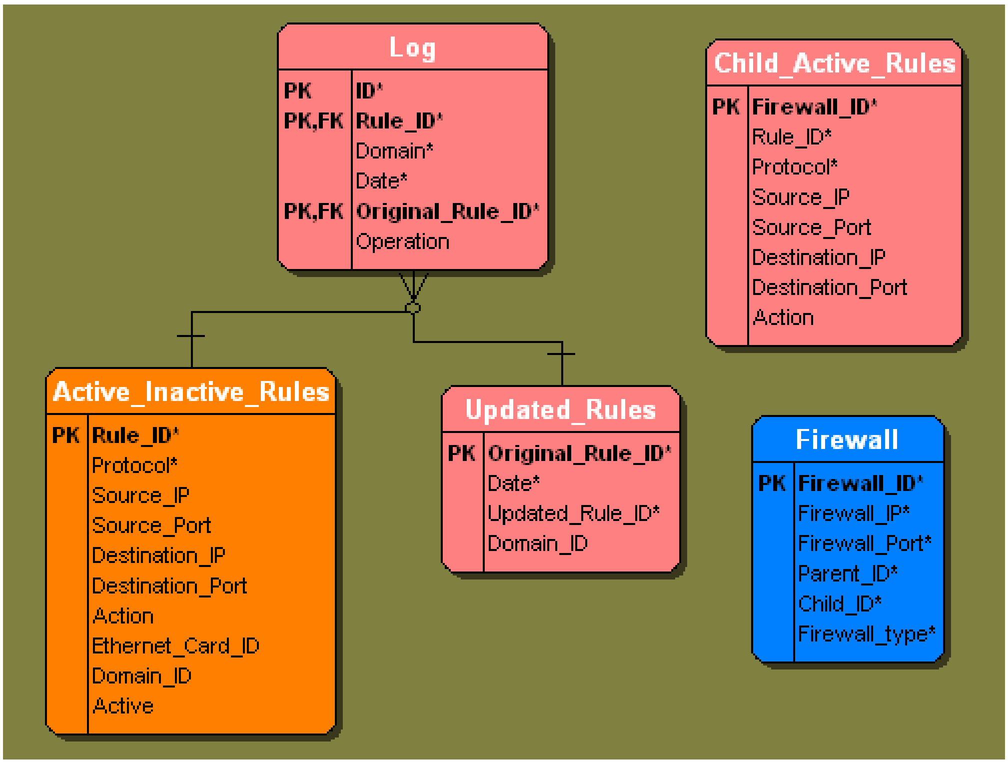

Figure 3.2. E-R Diagram of Database Tables.

3.2.1. Subnet Firewall Design

Subnet firewalls have their own local policy rules to provide the security. They interested in to keep track of the following operations that are performed on firewall policy rules: creating a new rule, reading, updating and deleting an existing rule inside the rule set. Moreover, subnet firewalls may have child firewall node, so they have some right over the their child firewall rule set such as getting, deleting and inserting rule to the rule set. Also, there are three different domains available and each domain has different rule set. In order to meet this requirement, the following tables will be stored in each local database of subnet firewall.

Table 3.1. Subnet Firewall Design

Table Name	Explanation
Active_Inactive_Rules	This table contains the policy rules that are active or inactive.
Updated_Rules	This table is to keep track of updated rules.
Log	This table lists the actions that have occurred. Example of actions are create, read, update, delete operations and relevant information like who made the action, when, etc.
Child_Active_Rules	Aim of the table is keeping the child firewall active rules.

The detailed explanation of these tables is given below.

3.2.1.1. Active_Inactive_Rules Table

Basically, this table contains the necessary information of the firewall policy rule (Table 3.2). This rule can be actively applied by the firewall or can not be valid any more. The database table contains the following fields, which are used to describe a firewall policy rule.

Table 3.2. Active_Inactive_Rules Table

Field Name	Explanation
Rule ID	A unique rule ID to identify policy rule within the overall system.
Protocol	The network protocol of the traffic to which this rule will be applied.
Source IP Address	The IP Address of the source that has produced the network traffic.
Source Port Number	Port number of the source that has created the network traffic.
Destination IP Address	IP Address of the target machine to which source is trying to connect.

(cont. on the next page)

Table 3.2. (cont.). Active_Inactive_Rules Table

Destination Port Number	Port number on the target machine
Action	The action such as accept, deny that will be performed by the firewall when a matching network packet is found for this rule.
Ethernet Card ID	The identity (ID) of the physical interface through which the network traffic passes.
Domain ID	This field is keeps the Domain ID of the rule so user can define different rules for each domain according to the security needs.
Active	This field is describing the rule is active or inactive rule.

3.2.1.2. Updated_Rules Table

It may become necessary to update policy times in time as a result of changing requirements, etc. This table helps to implement this feature by using the information stored in the following fields (Table 3.3).

Table 3.3. Updated Rules Table

Field Name	Explanation
Original Rule ID	The ID of the rule before the update operation
Date	The date when the rule is updated
Updated Rule ID	Once the update operation is completed, the new rule is inserted into the Active Rules table with this new rule id.
Domain ID	The ID of the Domain which updated rules belongs to.

3.2.1.3. Log Table

Aim of the table is kind of logging in the system (Table 3.4). It basically stores the data to represent the Create, Read, Update and Delete operations performed on the rule set. The fields in the table are as follows:

Table 3.4. Log Table

Field Name	Explanation
Domain ID	The Domain ID inside which the specified operation is performed
Operation	The type of the operation (Create,Read,Delete,Update) performed on the ruleset
Date	Time of the operation
Rule ID	The ID of the rule on which the operation is performed.

3.2.1.4. Child_Active_Rules Table

Table is special for only subnet firewalls which have child nodes firewall in the system. Using this table subnet firewall can easily reach the child firewalls active rules (Table 3.5).

Table 3.5. Child_Active_Rules Table

Field Name	Explanation
Firewall ID	Unique ID of the firewall in the system.
Rule ID	A unique rule ID to identify each policy rule within the
Protocol	The network protocol of the traffic to which this rule will be
Source IP Address	The IP Address of the source that has produced the network traffic.
Source Port Number	Port number of the source that has created the network traffic.
Destination IP Address	IP Address of the target machine to which source is trying to connect
Destination Port	Port number on the target machine.
Action	The action, such as accept, deny; that will be performed by the firewall when a matching network packet is found for this rule.

3.2.2. Domain Firewall Design

Distributed firewalls same as the subnet firewalls have their own local policy rules to provide the security. They are responsible of following operations that are performed on firewall policy rules: creating a new rule, reading, updating and deleting an existing rule inside the rule set. Also there are three different domains available and each domain has different rule set. In addition to this In order to meet this requirement, the following tables will be stored in each local database of distributed firewall (Table 3.6).

Table 3.6. Domain Firewall Design

Table Name	Explanation
Active_Inactive_Rules	This table contains the policy rules that are active or inactive.
Updated_Rules	This table is to keep track of updated rules.
Firewall	This table keeps the each firewall information.
Child_Active_Rules	This table keeps active rules of child firewalls.

3.2.2.1. Firewall Table

This is the important table for the distributed firewall administration tool (Table 3.7). This table keeps the each firewall information such as firewall ID, IP number, firewall type. The fields in the table are as follows:

Table 3.7. Firewall Table

Field Name	Explanation
Firewall ID	Unique ID of the firewall in the system
Firewall IP	IP Address of the firewall which is defining in the System
Firewall Port	Port Number of the firewall which is listening.

(cont. on the next page)

Table 3.7. (cont.) Firewall Table

Parent ID	Parent ID of the firewall in the system if firewall does not have parent firewall in the system this field becomes default value which is "NULL".
Child ID	Child ID of the firewall in the system if firewall do not have any child this field becomes default value which is "NULL"
Firewall Type	This field identifies the firewall types in the system. Firewall types can be domain, subnet or personal firewall.

3.3. Scenarios Related to Database Tables

3.3.1. Create a New Rule

Subnet and distributed firewalls both of them can add new rule to the same database if subnet firewall parent of distributed firewall.

Domain Firewall

After necessary fields entered correctly new rule is inserted into Active_Inactive Rules table. After insertion is completed successfully, the Log table will be updated with the corresponding data. This process shown in Figure 3.3

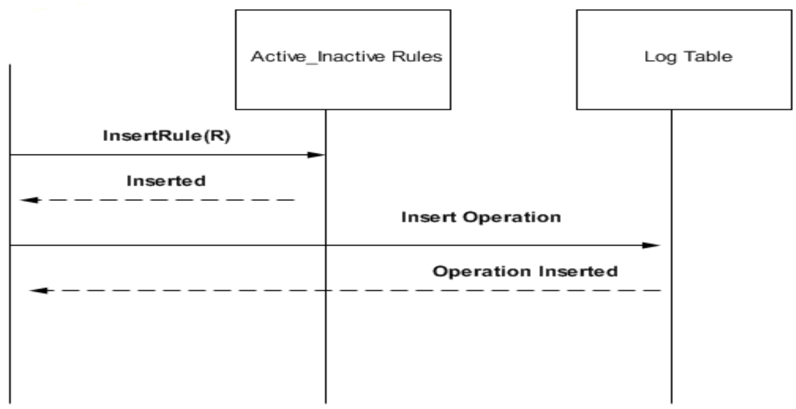

Figure 3.3. Distributed Firewall creating a new rule.

Subnet Firewall

Subnet firewall may request to add new rule to its child firewall node. First of all the new rule inserted into the Active_Inactive Rules table, then Log table will be updated, finally new rule is inserted into the Child Active Rules table also updated. This situation shown in Figure 3.4.

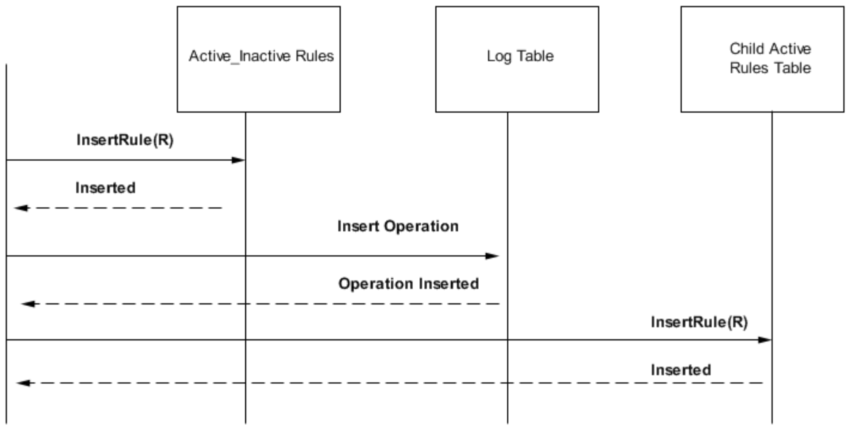

Figure 3.4. Subnet Firewall creating a new rule to it's Child Firewall.

3.3.2. Read a Rule

Reading rule operation can be performed in two different ways. First one is distributed firewall can read local rules, secondly subnet firewall can read its child firewall node rule.

Domain Firewall

As indicated by Figure 3.5, once a request comes to read an active rule, the required data will be extracted from Active_Inactive Rules table. After data has been gathered successfully, Log table will be updated with the corresponding data.

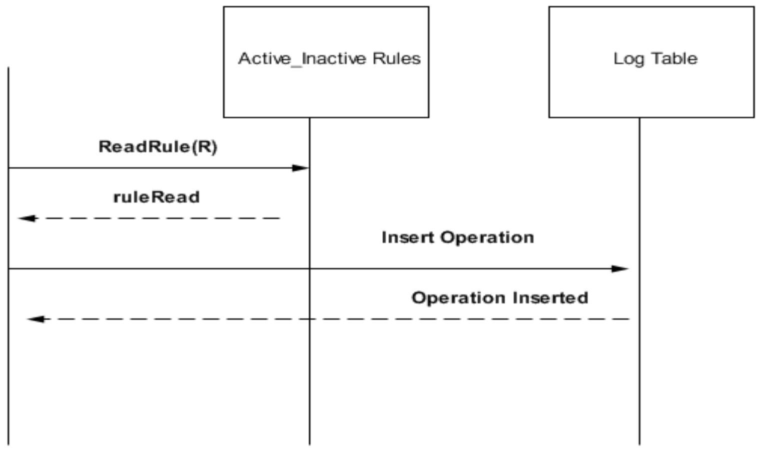

Figure 3.5. Domain Firewall get existing rule.

3.3.3. Delete a Rule

Deleting rule operation also can be performed in two different ways. First one is distributed firewall can delete its local rules. Secondly subnet firewall can delete its child firewall node rule.

Domain Firewall

While deleting existing rule the system, first of all active rule is updated to the deactivate rule on the Active_Inactive Rules table. Secondly, the Log table is updated accordingly. Figure 3.6 summarizes the sequence of operations that are performed when deleting a rule.

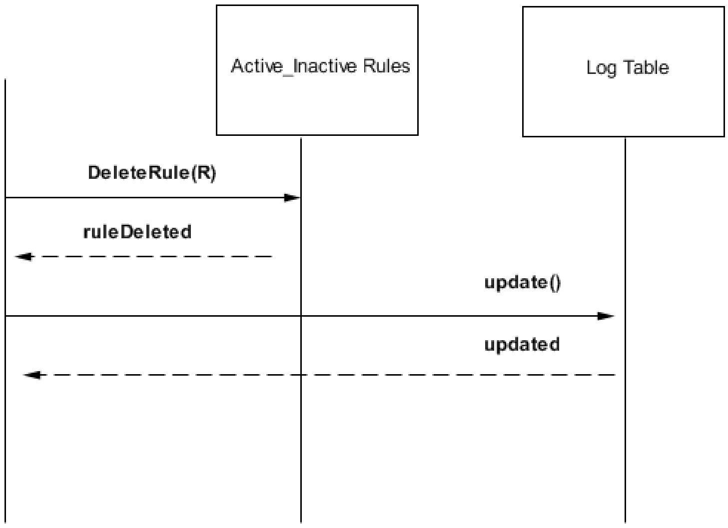

Figure 3.6. Distributed Firewall deleting a rule.

Subnet Firewall

Sometimes subnet firewall may request to its child firewall node to delete some rules. In this issue firstly rule is deleted from Active_Inactive Rules Table then Log Table Updated according to the operation. Lastly rules deleted from the Child Active Rules Table.

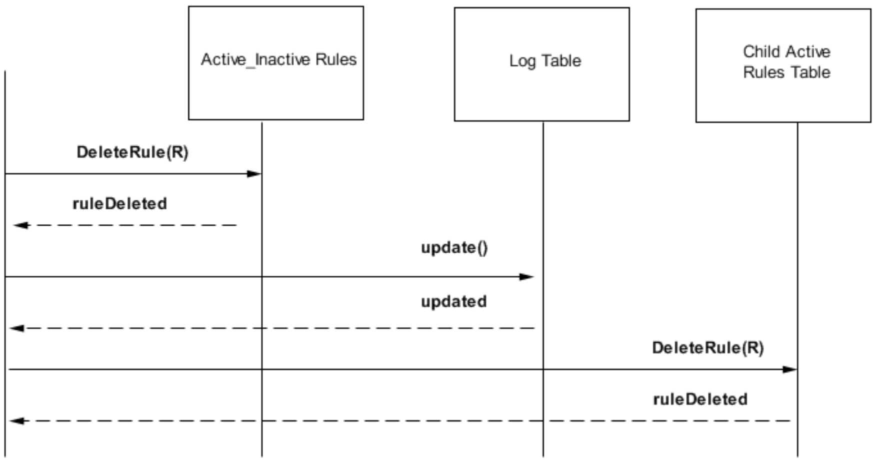

Figure 3.7. Subnet firewall deleting its Child firewall rule.

3.3.4. Update a Rule

According to the security needs user can update existing rules in the system. Update operation work principle is as follows, existing rule which is assume that A is updated into the new rule B. First of all A is updated to inactive rule from Active_InactiveRules table. Then, new rule B is added to Active_InactiveRules table. Then UpdatedRules table is updated with the updated rules. Finally Log table is updated with the operation. This sequence of operations illustrated in Figure 3.8.

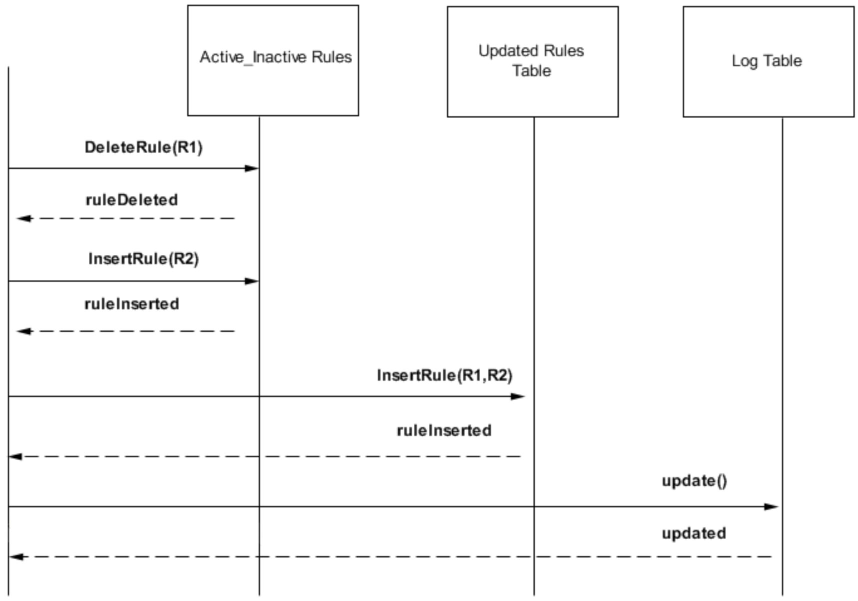

Figure 3.8. Tables that are used when updating a rule.

3.4. Software Design

3.4.1. Existing Software

Distributed Firewall Administration Tool is an improvement to the existing software which is the NetDefender program. NetDefender is a Free Firewall and works only on windows 2000 and above versions of windows. The graphical user interface of NetDefender software shown in Figure 3.9.

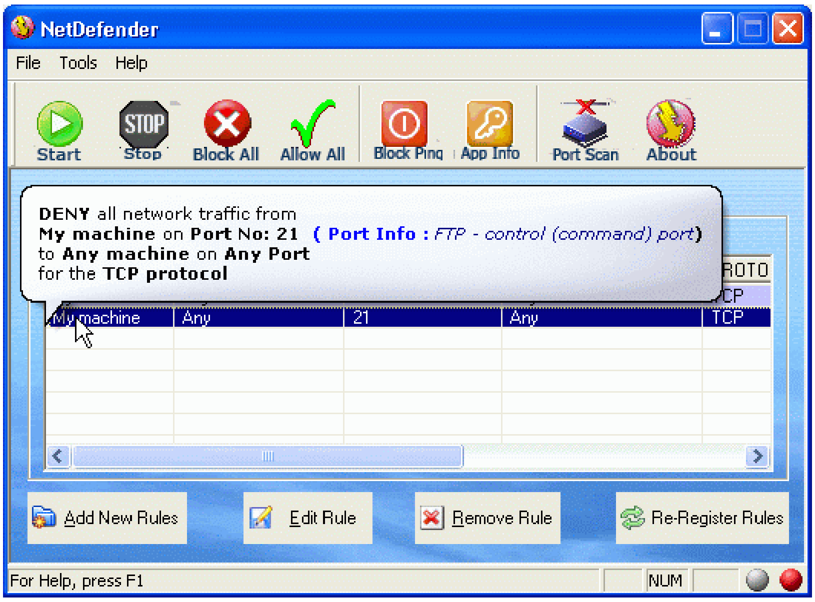

Figure 3.9. NetDefender Main Menu.

NetDefender has a very easy to use interface. NetDefender Firewall is completely written in VC++ 7.1 (Visual Studio 2003) using MFC, Windows API, Filter Hook Driver (Provided with Windows 2000). To run the NetDefender computer must have mfc71u.dll, msvcp71.dll and msvcr71.dll. NetDefender is packet filtering firewall and provides to user to add custom rules to this firewall as per their security requirements. When the user wants to add new rule clicks the Add New Rules button to call New Rule form and define rule based on source and destination IP, source and destination port number, protocol (TCP,UDP,ICMP) and action (ALLOW,DENY) features. Existing rules are listed on the main list NetDefender provides to modify, delete the existing rules on the rule set. If user wants to modify the existing rule, first of all clicks the rule and then click the Edit Rule button Modification form appears and rule features are filled after necessary modification done rule saved to IPFilter Driver. Simultaneously saved rule is also storing on the file. Port scanner is another feature of the software this feature provided to scan the system for open ports. NetDefender provides the list of applications that are currently trying to connect to outside network.

3.4.2. Modified Software

The proposed application is the Distributed Firewall Administration Tool (DFAT), which works on a distributed firewall environment. The details of the distributed firewall architecture on which the application will run are explained in proposed approach section. As stated in that section, the firewall system contains several nodes that are distributed into different domains and subnets organized in a hierarchical manner. Each node in the system is responsible for applying the subset of overall policy rules that are defined for the corresponding domain / subnet. As a result of changes in the security policy or detection of the problems in the existing rule set, it may be required to make some changes on the existing rule set. The major changes that can be performed on the rule set can be listed as follows:

- Creating a new firewall policy rule (see Figure 3.12) .

- Reading a policy rule from the existing rule set.

- Updating an existing rule.

- Deleting a rule from the existing rule set

Basically, Distributed Firewall Administration Tool (DFAT) improved the some features of NetDefender software. First of all, rule set is stored on the SQL Server 2000 Database and Domain Selection feature is added. Using Domain Selection feature, users can create new policy rules in three different domains. These domains are Department, Home and Internet Cafe. According to the security needs user can define different policy rules for different domains. This situation illustrated in Figure 3.10

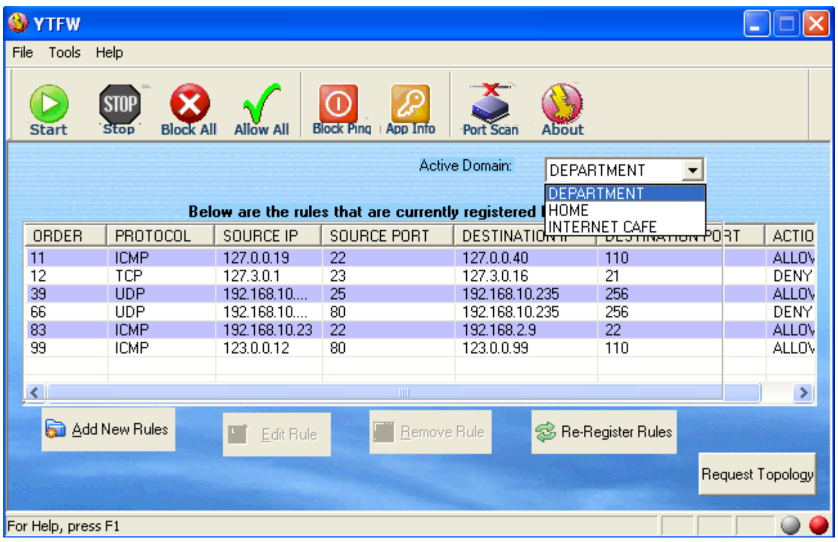

Figure 3.10. Domain Selection Screen.

DFAT uses broadcasting methods to find other computers on the distributed architecture. All firewall types (child, domain, and subnet) use this method to find other computers in the network. If firewall type is subnet firewall according to the proposed architecture firewall has one or more child firewalls. Subnet firewall can perform new rule, delete rule and get rule list function on child firewall rule set on Child Menu (shown in Figure 3.11).

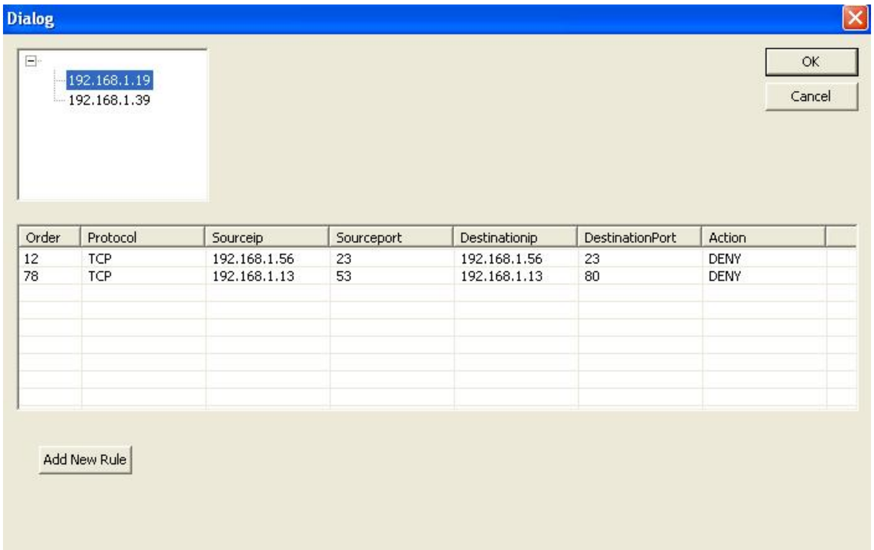

Figure 3.11. Child Menu Screen.

Child Menu provides parent firewalls with the ability to define new rule, delete existing rule and get list of existing rules. After broadcasting operation completed, parent firewalls have a list of their child lists. On the left part of the screen child firewalls IP address are listed. When user double clicks one of the IP address parent firewall send a request to get existing rule list to child if child firewall accept this operation list are loaded on the screen. Child firewall screen shown in Figure 3.11. According to the security needs, parent firewall may want to add new policy rule to child firewall. In this issue, user clicks the Add new rule button on the Child Menu screen, new rule form appears after rule created new rule sent to the child firewall shown in Figure 3.12.

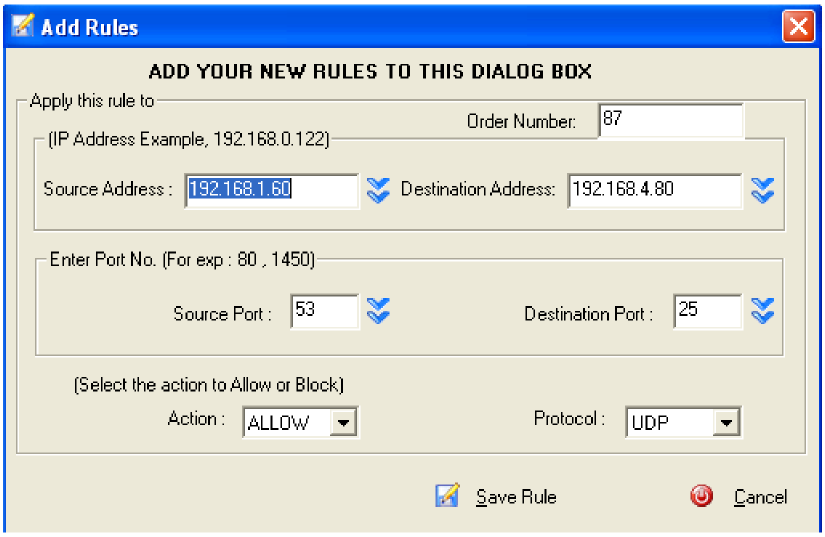

Figure 3.12. Add New Rule Form.

Domain firewall has features to request topology information from other firewalls. When responses are retrieved, domain firewall fills the firewall table.

Table 3.8. Comparison Of Software

Feature	**NetDefender**	**DF**
Rules Stored on Database	**X**	√
Domain Selection	**X**	√
Dynamically find each other by broadcasting	**X**	√
Add Rule to Child Firewalls Ruleset	**X**	√
Delete Rule from Child Firewalls	**X**	√
Using Algorithms to Build Network Topology	**X**	√
Identify the Ethernet Car ID	**X**	√
Firewall Type	**X**	√

√ : does support **X: does not support**

Functional Requirements

Distributed firewall administration tool is decided to have some functionalities, which are mainly related with the firewall policy rules. Details of the functions will be explained in the following subsection.

3.4.3. New Rule Function

According to the proposed approach, which is described in Section 3.1, each node is a firewall and all of them has local database to perform policy rule operations. Purpose of the function is to add new policy rule to IPFilterdriver and Active_Inactive Rules table. New Rule function is shown in Table 3.9.

Table 3.9. "New Rule" Function

Inputs	Order ID, Protocol, Source IP, Source Port, Destination IP, Destination Port, Action
Operations	After all necessary inputs filled correctly New Rule Functions, first of all add new rule to IPFilter Driver using Firewall Hook Driver. Next if operation is succeeding rule will be added to the Active_Inactive Rules Table. This added operation also added to the Log table to inform the parent firewall of the node.
Outputs	Two different types of Output return from this function. Firstly if new rule added properly without any error this function returns "SUCCESS" but if any error occurred during the operation types of error returns.

3.4.4. Delete Rule Function

Sometimes some of the existing policy rules can not be necessary for the security. Aim of the function is deleting existing rule from security policy (Table 3.10).

Table 3.10. “Delete Rule” Function

Inputs	Order ID of the policy rule
Operations	When the user has selected policy rule first of all rule is deleting from Active_Inactive Rules table. Actually deleting operation just updating Active field of the rule then operation is inserting the Log table.
Outputs	Outputs of the this function is if rule deleted successfully without any error this function returns “SUCCESS” but if any error occurred during the operation “ERROR” returns.

3.4.5. Update Rule Function

User sometimes wants to change the policy rules according to the security needs. This function provides ability to update existing rule (Table 3.11).

Table 3.11. “Update Rule” Function

Inputs	Order ID, Protocol, Source IP, Soucer Port, Destination IP, Destination Port, Action
Operations	First of all user selects the policy rule which will modify. According to the Order number rules deleted from Active_Inactive Rules Table. Then user updates some fields of the policy rule and new rule inserted in to the Active_Inactive Rules Table and old Order number, new Order number number and date of operation are inserted in to the Log Table.
Outputs	When the system performs all operation correctly function returns “Successful” but during the operation if any error occurs it returns the specific error message.

3.4.6. Domain Selection Function

Distributed firewall administration tool provides functionality to users for selecting a domain. There are three different domains available which are Department, Home and Internet cafe (Table 3.12).

Table 3.12. “Domain Selection” Function

Inputs	Domain ID: ID number of the Domain
Operations	User selects the one of the domain from the list then selected domain policy rules taken from Active_Inactive Rules table then loaded to the IPFilterdriver.
Outputs	Output of the function is selected domain rules loaded to the IPFilterdriver and data list of the program.

3.4.7. New Rule to Child Function

According to the proposed architecture each firewall can be parent firewall, child firewall, or both. Each parent firewall in the architecture can interfere to its child policy rules. Parent firewall can insert new rules to its child rule set but child firewall have right to accept or deny this process. This function shown in Table 3.13.

Table 3.13. “New Rule To Child” Function

Inputs	Firewall IP, New Rule features as described in section 3.4.1
Operations	Child firewall listens the specific port when the new rule retrieved child firewall has right to accept or deny this operation. If child firewall accepts the rule, rule is loaded the IPFilterDriver and then sent to the Parent Firewall. If child firewall denies send to deny packet to parent firewall.
Outputs	If child firewall accepts new rule loaded to IPFilterDriver and stored to child firewall database and new rule sent to the parent firewall.

3.4.8. Delete Rule from Child Function

Each parent firewall in the architecture can interfere to its child policy rules. Parent firewall can delete rules from its child rule set but child firewall have right to accept or deny this process (Table 3.14).

Table 3.14. "Delete Rules From Child" Function

Inputs	Firewall IP, Order ID
Operations	Child firewall listens the specific port when the Order ID retrieved rule features appear on the screen child firewall has right to accept or deny this operation. If child firewall accepts, rule is deleted the IPFilterDriver and database. Then Operation complete packet sends to parent firewall. If child firewall denies sent to deny packet to parent firewall.
Outputs	If child firewall accepts, rule deleted from Ip IPFilterDriver and database. If deny, deny packet send to parent firewall.

3.4.9. Get Active Rule from Child Function

To be able to protect the consistency among the overall security policy, parent firewall may get all existing rules of its child firewall policy rules (Table 3.15).

Table 3.15. "Get Rule List From List" Function

Inputs	Firewall IP
Operations	Child firewall listens the specific port when the request retrieved child firewall sends all existing rules to parent firewall.
Outputs	Child firewall send existing rules to parent firewall.

3.4.10. Request Topology Function

Distributed firewall administration tool works on a distributed environment. At the beginning domain firewall hasn't got any information about distributed firewall architecture. Domain firewall sends a request packet to its children which are subnet firewall. This request is retrieved and sent to subnet firewall's child firewalls. This recursive function works until all child firewall, send firewall related information to the domain firewall. Domain firewall implements Breadth-first search and Depth-first search algorithms to determine the distributed firewall architecture.

Table 3.16. "Request Topology" Function

Inputs	Input of the function is TCP packet, which contains Firewall ID, firewall type and Firewall IP.
Operations	Firewall is listening the specific port when the request retrieved firewall looks its type if firewall type is subnet firewall it send request to its child and listen to child's response. When the response arrived firewall append the ruleset its own information and send to the parent.
Outputs	Domain firewall process two algorithms to define distributed architecture, this algorithms input are node relations.

3.4.11. Broadcasting Function

Aim of this function is to inform all firewalls about environment. At the beginning neither parent nor child do have information about each other. Because of that they broadcast and introduce themselves. Broadcasting function shown in Table 3.17.

Table 3.17. "Broadcasting" Function

Inputs	Input of function is TCP packet which contains IP Address and type of firewall.

(cont. on the next page)

Table 3.17. (cont.) “Broadcasting” Function

Operations	Every firewall listens specific port if any request retrieved firewall insert packet data to the database.
Outputs	Packet data insert into the firewall table and data list of the program.

3.5. Algorithms

Distributed Firewall administration tool discovers the distributed firewall architecture using two algorithms which are Breadth-first search (BFS) and Depth-first search (DFS).

3.5.1. Breadth-First Search Algorithm

Breadth-first search (BFS) is a graph search algorithm that begins at the root node and explores all the neighboring nodes. BFS is an uninformed search method that aims to expand and examine all nodes of a graph or combinations of sequence by systematically searching through every solution..

```
Define randomBFS(Graph g, float prob, Node startNode, Node
                        destinationNode)
        Node n:=startNode
        Queue q:={}
        Stack s:={}
        Enqueue(q,startNode)
        Visited[startNode]:=true
         While(n not equals destinationNode)
            if(queue empty)
                tempNode:=pop(s)
                Enqueue(tempNode)
                 For each node (g)
                     Visited[node]:=false
                 n:=Dequeue(q)
                 if(n not equals destinationNode)
                    push(s,n)
                    for each neighbor of n
                       if(visited[neigbor]=false)
                           if(prob<rand())
                               visited[neigbor]:=true
                                Enqueue(q.neigbor)
```

Figure 3.13. Breadth-First Search Algorithm
(Source: Sahni 1998)

In other words, it searches the entire graph or sequence without considering the goal until it finds it. Algorithm is illustrated in Figure 3.13

Time complexity of Breath First Search algorithm can be described as follows. There is a search tree with branching factor n and the maximum path length is d. The root of the search tree generates n nodes at the first level, each of which generates n more nodes, for a total of n^2 at the second level. Each of these generates n more nodes, yielding n^3 nodes at the third level, and so on. Then the maximum number of nodes expanded before finding a solution is $1 + n + n^2 + n^3 +...+ n^d$. Therefore, in the worst case the complexity bound would be $O(n^d)$. Because it must examine every node in the tree. Space complexity of the algorithm is also $O(n^d)$ because it must store the whole frontier in memory (Gutin and Jensen 2001).

Breadth-first search algorithm is implemented in order to define the distributed firewall architecture. Algorithm is only performed by domain firewall. All possible paths between subdomains in the network must be determined before the execution of the algorithm. Breadth-first search algorithm takes a starting node which is domain firewall node as parameter. Our domain firewall is at level 0. In the first stage, it visit all subnet firewalls at level 1. In the second stage, it visits all subnet firewalls or child firewalls at second level. These new subnet firewalls or child firewalls, which are adjacent to level 1 subnet firewalls, and so on. This situation illustrated in Figure 3.14.

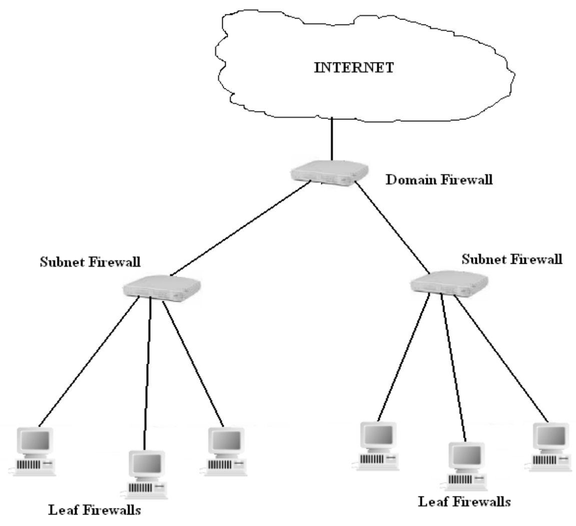

Figure 3.14. Example of Distributed Firewall Environment.

3.5.2. Dept-First Search Algorithm

Depth-first search (DFS) is an uninformed search algorithm that progresses by starting the first child node of the search tree that appears and thus traversing until a node that has no children. Algorithm is illustrated in Figure 3.15.

```
dfs( graph G)
{
    list L=empty
    tree T=empty
    choose a strating vertex x
    search(x)
    while(L is not empty)
    {
        remove edge (v,w) from beginning of L
        if w not yet visited
        {
            add (v,w) to T
            search(w)
        }
    }
}

search(vertex v)
{
    visit v
    for each edge (v,w)
    add edge (v,w) to the beginning of L

}
```

Figure 3.15. Depth-First Search Algorithm
(Source: Sahni 1998)

The time complexity for a depth first search is $O(n^d)$, where n is the branching factor and d is the maximum depth. The time complexity is the same with respect to the breath first search. The drawback to the depth first search is that it can get stuck going down the wrong path and never recover, with respect to time, from an unlucky choice at one of the nodes near the top of the tree. Space complexity of the algorithm is O(nd), for a depth first search in memory, it only needs to store a single path from the root to a leaf node, along with the remaining unexpanded sibling nodes for each node on the path (Gutin and Jensen 2001).

Depth-first search algorithm provides to define distributed firewall architecture to distributed firewall administration tool. After all the possible paths defined in the network algorithm starts from root node which is domain firewall node and thus going deeper and deeper until it finds a node that has no children. Then the search backtracks, returning to the most recent node it hadn't finished exploring.

Breath-first search algorithm always finds the shortest path to goal for example root node wants to interfere any leaf firewall and if there are several solutions breadth-first search provide shortest path to destination leaf firewall. The major weakness of Depth-first search algorithm is that it will fail to terminate if there is an infinite path "to the left of" the path to the first solution so DFS is not complete. Furthermore, DFS is not suitable for large search tree or infinite maximum depth.

Alternatively there are well-known graph search algorithms available and they can be implemented in the project. First one is Dijkstra's algorithm solves the problem of finding the shortest path from a point in a graph (the *source*) to a destination. Secondly Bellman-Ford algorithm, this algorithm used by Routers to maintain the distance tables, which tell the distances and shortest path to sending packets to each node in the network (Gutin and Jensen2001).

CHAPTER 4

EXPERIMENTS AND EVALUATION

On top of the proposed distributed architecture, two different algorithms, namely Breadth-First Search (BFS) and Depth-First Search (DFS) are implemented for topology discovery and rule collection. These two algorithms are applied to different experimental topologies. Basically, the domain firewall (root node) requests distributed architecture information using one of the algorithms from subnet firewalls which also request this information from their child firewalls and this recursive iteration continues according to the selected algorithm. Breadth-First Search algorithm discovers all neighboring nodes. Starting from those nearest nodes, it finds their undiscovered neighbor nodes and this iterative process proceeds until the last node. Depth-First Search algorithm traverses a single path of the graph that is, until it visits a node with no child, it then returns at the last node on the path that has an unvisited child then traversing starts from that node. In this experiment, the traversing time of the proposed architecture is calculated in terms of selected algorithm. These experimental topologies constructed on the 19 computers which have Windows XP operating systems and Pentium IV-M 1.73 GHz. processor with 1 GByte of RAM. Each computer network card is Marvell Yukon 88E8056 PCI-E 10/100/1000Base-T Ethernet Controller.

According to the proposed architecture each computer should have more than one network interface card (NIC) and should be assign a different IP address to each card so different subnets can be created. However, it is possible to assign multiple IP address to a single NIC in Windows XP. Assigning more than one IP address to a network card on Windows requires the steps below:

- Right-click on My Network Places, choose Properties.
- Right-click on the Local Area Connection, choose Properties.

- Highlight Internet Protocol (TCP/IP), click Properties.
- Set IP address, Subnet Mask and Default Gateway, click Advanced button shown in Figure 4.1.

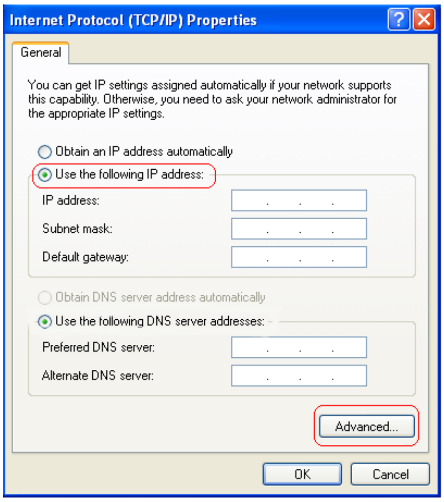

Figure 4.1.Windows XP TCP/IP Menu.

After Advanced button is clicked, Advanced TCP/IP Settings Form appears. Click the Add button and enter a new IP address and Subnet mask. According to the network topology procedure can be repeated. This procedure shown in Figure 4.2.

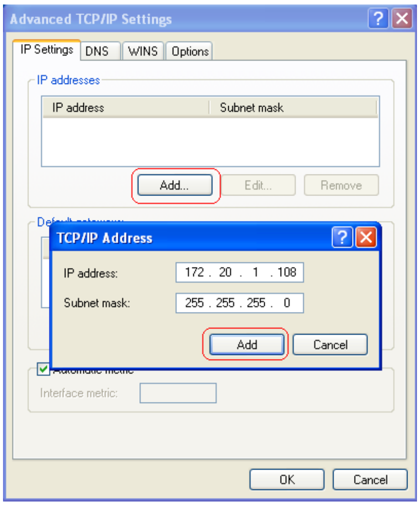

Figure 4.2.Windows XP Advanced TCP/IP Menu.

After multiple IP addresses are assigned to related computers which are subnet firewalls and domain firewalls, different experimental topologies are ready. Basically, the experiment is about calculating the amount of time to construct the distributed firewall architecture according to the selected algorithm. First group consists of networks which have 2 subnet firewalls where each of them has 2, 3, 4, and 5 leaf firewalls. Second group has 3 subnet firewall each of them has 2, 3, 4 and 5 leaf firewalls.

4.1. Experimental Setups For Balanced Topology

In order to distinguish algorithm's execution time two different kind of topology implemented. First of all proposed architecture implemented like a balanced topology, in this topology each subnet firewall have equal number of leaf firewalls.

4.1.1. Experimental Setup I

This experimental setup (Figure 4.3) is composed of 7 firewalls and 2 levels of hierarchy. Each firewall in the each level of hierarchy has two children except the ones in the leaf firewalls. Each firewall in the tree is responsible for executing Distributed Firewall Administration Tool and maintains a local database to store the policy rules that are mentioned in the previous chapter. Domain firewall, which is at the highest level of the hierarchy, sends distributed architecture request has been performed.

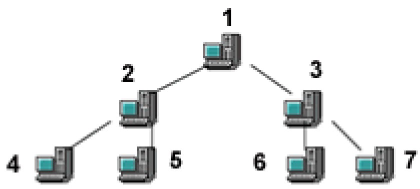

Figure 4.3. Experimental Setup I

The experiment is performed 100 times for each algorithm. In DFS algorithm the average time to complete the request is 240 milliseconds. However in BFS algorithm the average time to complete the request is 216 milliseconds.

4.1.2. Experimental Setup II

The second experimental architecture (Figure 4.4) is composed of 9 firewalls and 2 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 258 milliseconds. However in Breadth-First Search (BFS) algorithm the average time values to complete the request is 223 milliseconds.

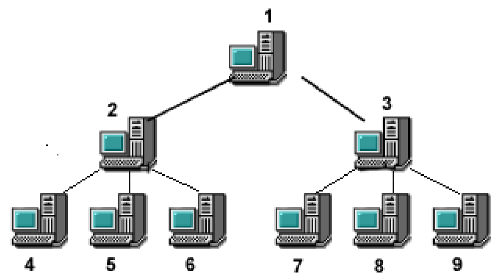

Figure 4.4. Experimental Setup II

4.1.3. Experimental Setup III

The third experimental architecture (Figure 4.5) is composed of 11 firewalls and 2 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 282 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time values to complete the request is 238 milliseconds.

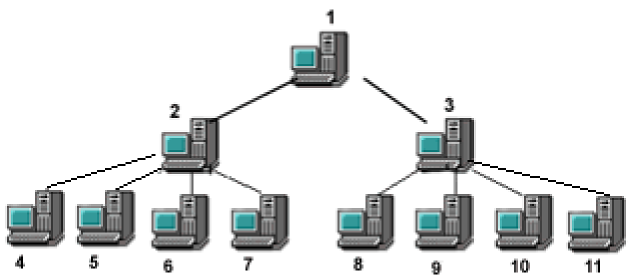

Figure 4.5. Experimental Setup III

4.1.4. Experimental Setup IV

The fourth experimental architecture (Figure 4.6) is composed of 13 firewalls and 2 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 305 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time values to complete the request is 256 milliseconds.

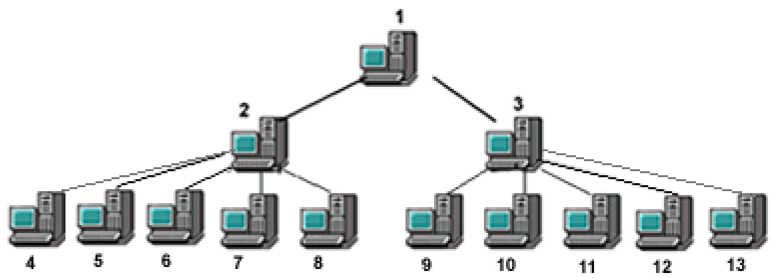

Figure 4.6. Experimental Setup IV

In each case, experiments are performed 100 times and average processing time is measured for both algorithms. The results are shown in Figure 4.7.

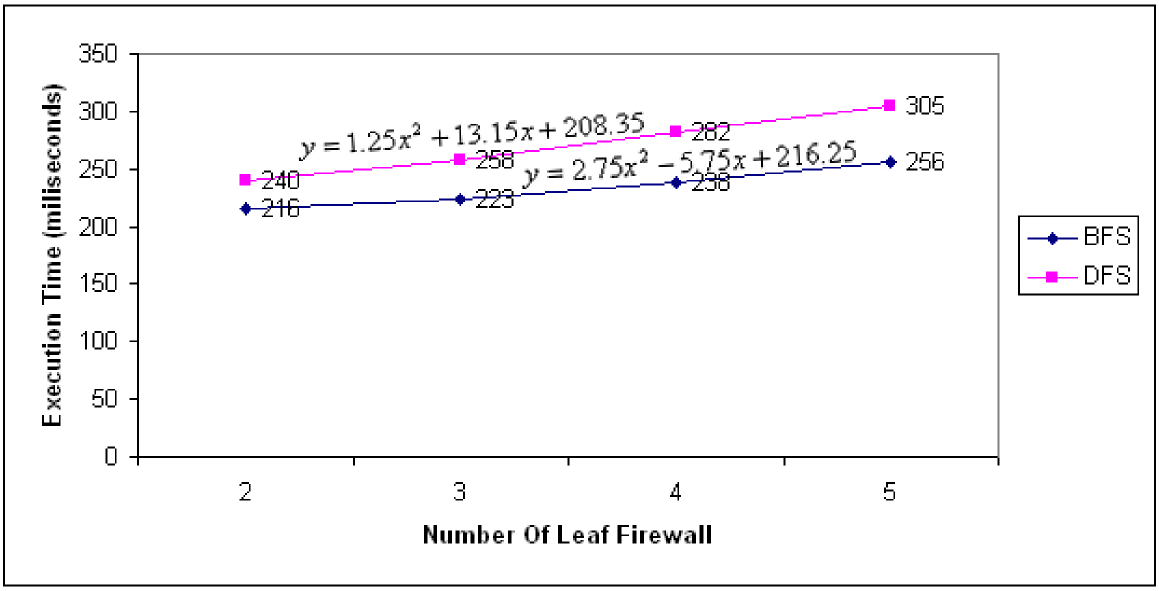

Figure 4.7. Processing Time for 2 Subnets

Equation of the BFS line is $y = 2.75x^2 - 5.75x + 216.25$ however DFS line equation is $y = 1.25x^2 + 13.15x + 208.35$. This result which is shown in Figure 4.7 indicates that Depth-First Search algorithm processing time higher than Breadth-First Search algorithm. Breadth-First Search algorithm work principle is visits all of the node nearest to starting node then begin distributing outer but in Depth-First Search, it is continue to deeper along a given path first, and returns only when no child. In the experiments performed when number of leaf firewall increases DFS algorithm has much execution time than BFS.

4.1.5. Experimental Setup V

The fifth experimental architecture (Figure 4.8) is composed of 10 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 416 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time value to complete the request is 330 milliseconds.

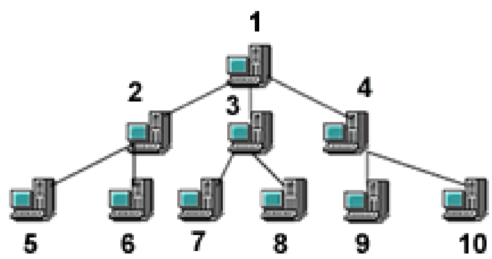

Figure 4.8. Experimental Setup V

4.1.6. Experimental Setup VI

The sixth experimental architecture (Figure 4.9) is composed of 13 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 460 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time values to complete the request is 358 milliseconds.

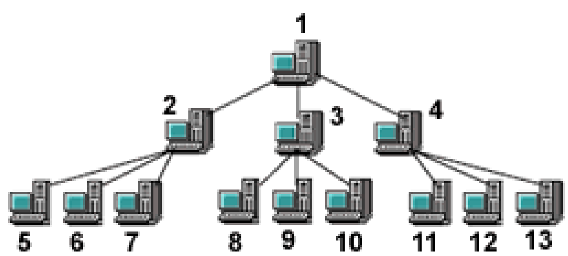

Figure 4.9. Experimental Setup VI

4.1.7. Experimental Setup VII

The seventh experimental architecture (Figure 4.10) is composed of 16 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 468 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time values to complete the request is 380 milliseconds.

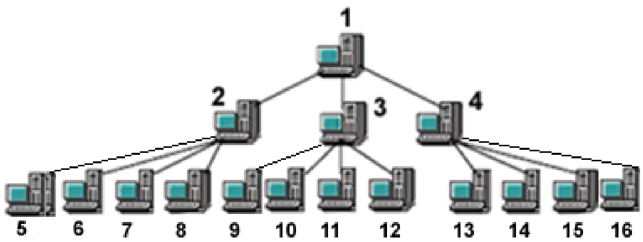

Figure 4.10. Experimental Setup VII

4.1.8. Experimental Setup VIII

The eight experimental architecture (Figure 4.11) is composed of 19 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. In Depth-First Search (DFS) algorithm the average time values to complete the request is 500 milliseconds. However, in Breadth-First Search (BFS) algorithm the average time values to complete the request is 396 milliseconds.

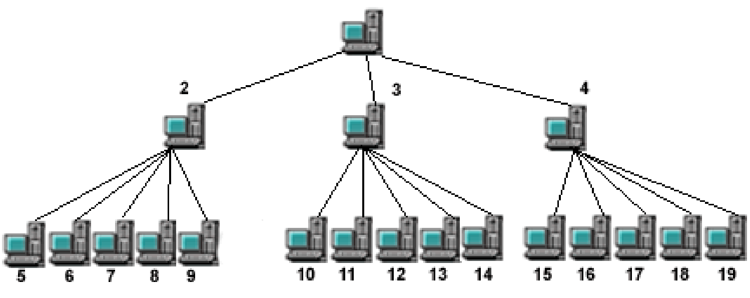

Figure 4.11. Experimental Setup VIII

In each case, experiments are performed 100 times and average processing time is measured for both algorithms. The results are shown in Figure 4.12.

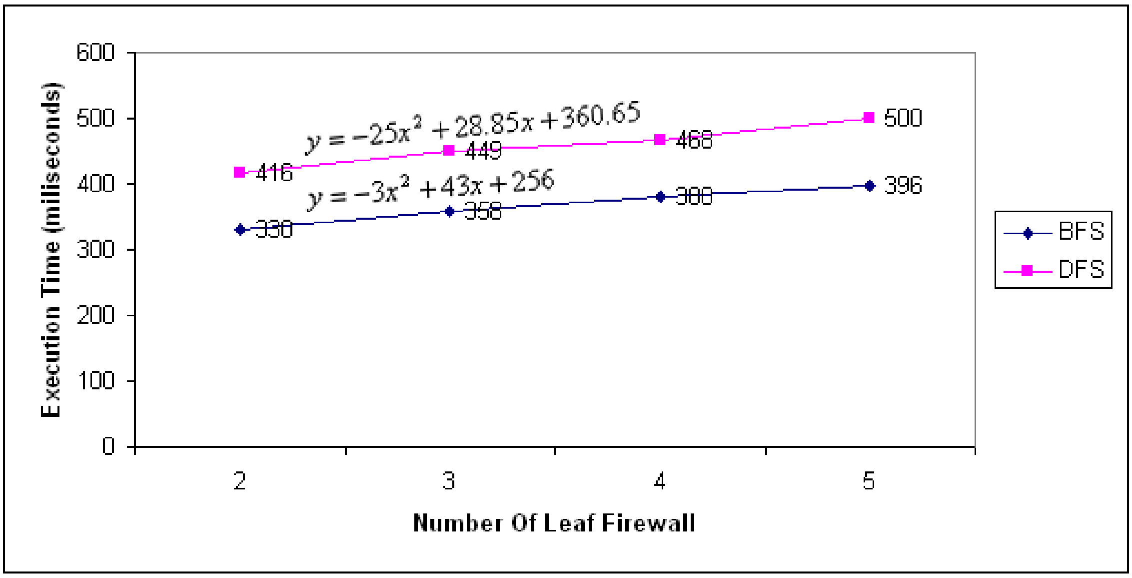

Figure 4.12. Processing Time for 3 Subnets

Equation of the BFS line is $y = -3x^2 + 43x + 256$ however DFS line equation is $y = -25x^2 + 28.85x + 360.65$. This result which is Figure 4.12 indicates that Depth-First Search algorithm processing time higher than Breadth-First Search algorithm. Breadth-First Search algorithm work principle is visits all of the nodes nearest to starting node then begin distributing outer but in Depth-First Search, it is continue to deeper along a given path first, and returns only when no child. In the experiments performed when the number of leaf firewall increases DFS algorithm has much execution time than BFS.

4.2. Experimental Setups for Unbalanced Tree

In this experiment the proposed architecture is constructed like unbalanced tree, each subnet firewall has different number of leaf firewalls.

4.2.1. Experimental Setup IX

This experimental setup (Figure 4.13) is composed of 6 firewalls and 2 levels of hierarchy. Subnet firewall, which stands on the left branch of the tree, has two leaf firewalls in contrast right branch of the subnet has just a single leaf firewall. Each firewall in the tree is responsible for executing Distributed Firewall Administration Tool and maintains a local database to store the policy rules that are mentioned in the previous chapter. Domain firewall, which is at the highest level of the hierarchy, sends distributed architecture information collection request.

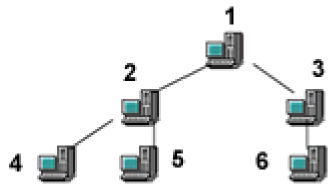

Figure 4.13. Experimental Setup IX

The experiment is performed 100 times for each algorithm. In DFS algorithm the average time to complete the request is 348 milliseconds. However, in BFS algorithm the average time to complete the request is 317 milliseconds.

4.2.2. Experimental Setup X

This experimental setup (Figure 4.14) is composed of 7 firewalls and 2 levels of hierarchy. Subnet firewall, which stands on left branch of the tree, has three leaf firewalls in contrast right branch of the subnet has just a single leaf firewall. Each firewall in the tree is responsible for executing Distributed Firewall Administration Tool and maintains a local database to store the policy rules that are mentioned in the previous chapter. Domain firewall, which is at the highest level of the hierarchy, sends distributed architecture information collection request.

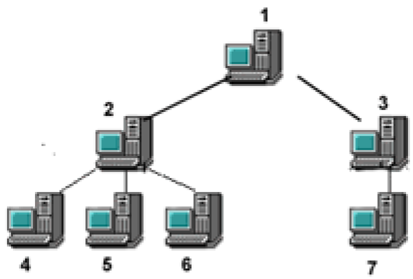

Figure 4.14. Experimental Setup X

The experiment is performed 100 times for each algorithm. In DFS algorithm the average time to complete the request is 364 milliseconds. However, in BFS algorithm the average time to complete the request is 331 milliseconds.

4.2.3. Experimental Setup XI

This experimental setup (Figure 4.15) is composed of 8 firewalls and 2 levels of hierarchy. Subnet firewall, which stands on the left branch of the tree, has four leaf firewalls in contrast right branch of the subnet has just a single leaf firewall. Each firewall in the tree is responsible for executing Distributed Firewall Administration Tool and maintains a local database to store the policy rules that are mentioned in the previous chapter. Domain firewall, which is at the highest level of the hierarchy, sends distributed architecture information collection request.

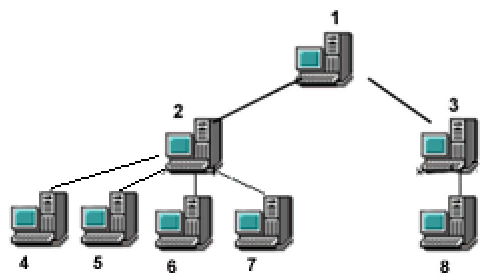

Figure 4.15. Experimental Setup XI

The experiment is performed 100 times for each algorithm. In DFS algorithm the average time to complete the request is 388 milliseconds. However, in BFS algorithm the average time to complete the request is 342 milliseconds.

4.2.4. Experimental Setup XII

This experimental setup (Figure 4.16) is composed of 9 firewalls and 2 levels of hierarchy. Subnet firewall, which stands on the left branch of the tree, has five leaf firewalls in contrast right branch of the subnet has just a single leaf firewall. Each firewall in the tree is responsible for executing Distributed Firewall Administration Tool and maintains a local database to store the policy rules that are mentioned in the previous chapter. Domain firewall, which is at the highest level of the hierarchy, sends distributed architecture information collection request.

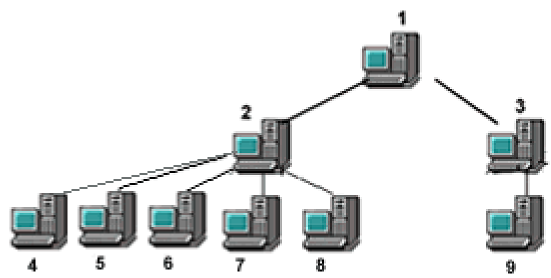

Figure 4.16.Experimental Setup XII

The experiment is performed 100 times for each algorithm. In DFS algorithm the average time to complete the request is 413 milliseconds. However, in BFS algorithm the average time to complete the request is 355 milliseconds.

In each case, experiments are performed 100 times and average processing time is measured for both algorithms. The results are shown in Figure 4.17.

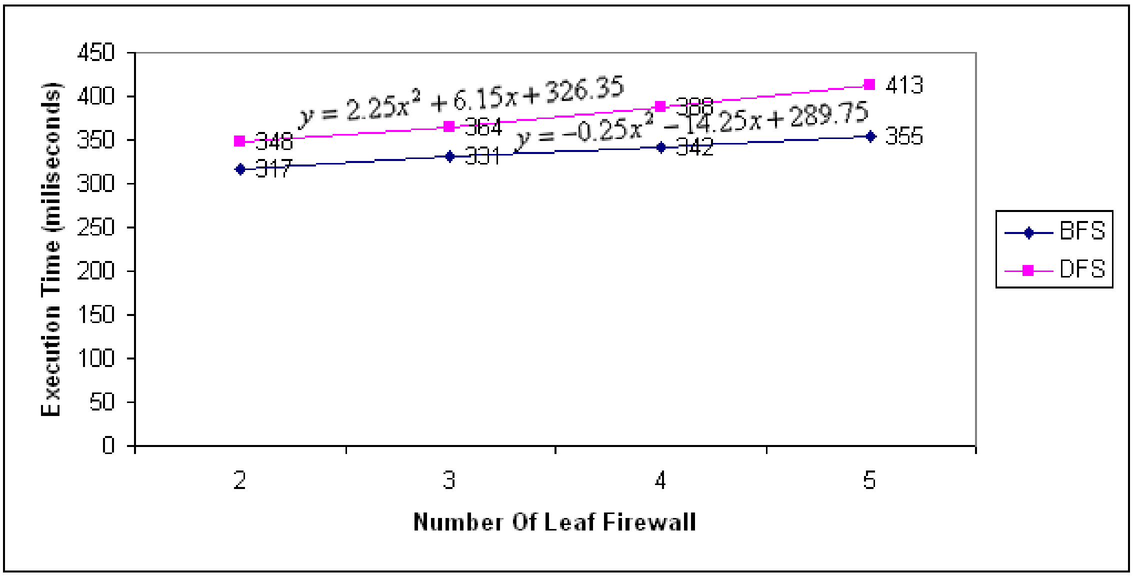

Figure 4.17.Processing Time for 2 Subnets

Equation of the BFS line is $y = -0.25x^2 - 14.25x + 289.75$ however DFS line equation is $y = 2.25x^2 + 6.15x + 326.35$. In the experiments performed when the number of leaf firewall increases in unbalanced topology DFS algorithm execution time higher than BFS.

4.2.5. Experimental Setup XIII

The thirteenth experimental architecture (Figure 4.18) is composed of 8 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. Subnet firewall which stands on the left branch of the tree has two leaf firewall but other subnet firewalls have just one leaf firewall. In Depth-First Search (DFS) algorithm the average time to complete the request is 409 milliseconds. However in Breadth-First Search (BFS) algorithm the average time to complete the request is 364 milliseconds.

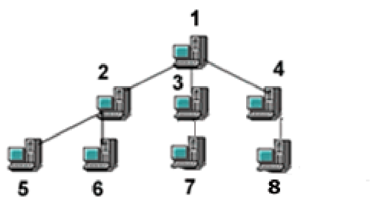

Figure 4.18. Experimental Setup XIII

4.2.6. Experimental Setup XIV

The fourteenth experimental architecture (Figure 4.19) is composed of 9 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. Subnet firewall which stands on the left branch of the tree has three leaf firewall but other subnet firewalls have just one leaf firewall. In Depth-First Search (DFS) algorithm the average time values to complete the request is 423 milliseconds. However in Breadth-First Search (BFS) algorithm the average time values to complete the request is 375 milliseconds.

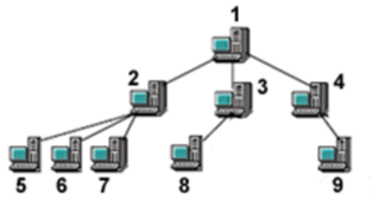

Figure 4.19. Experimental Setup XIV

4.2.7. Experimental Setup XV

The fifteenth experimental architecture (Figure 4.20) is composed of 10 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. Subnet firewall which stands on the left branch of the tree has four leaf firewall but other subnet firewalls have just one leaf firewall. In Depth-First Search (DFS) algorithm the average time values to complete the request is 443 milliseconds. However in Breadth-First Search (BFS) algorithm the average time values to complete the request is 380 milliseconds.

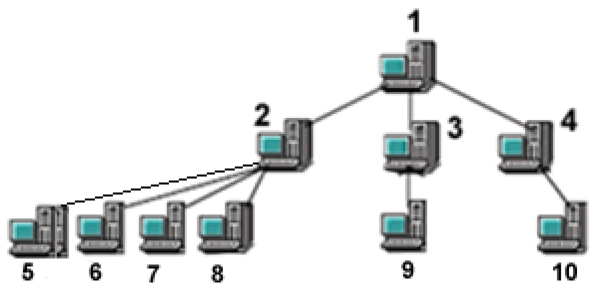

Figure 4.20.Experimental Setup XV

4.2.8. Experimental Setup XVI

The sixteenth experimental architecture (Figure 4.21) is composed of 11 firewalls and 2 levels of hierarchy. Domain firewall has 3 Subnet firewall experimental steps mentioned for the previous cases are also valid for this architecture. Subnet firewall which stands on the left branch of the tree has five leaf firewalls but other subnet firewalls have just one leaf firewall. In Depth-First Search (DFS) algorithm the average time values to complete the request is 473 milliseconds. However in Breadth-First Search (BFS) algorithm the average time values to complete the request is 387 milliseconds.

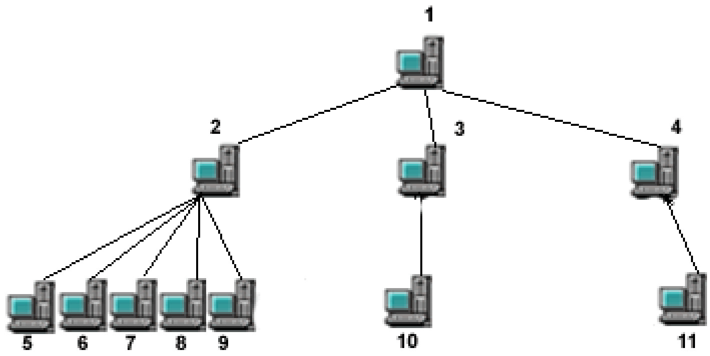

Figure 4.21. Experimental Setup XVI

In each case, experiments are performed 100 times and average processing time is measured for both algorithms. The results are shown in Figure 4.22.

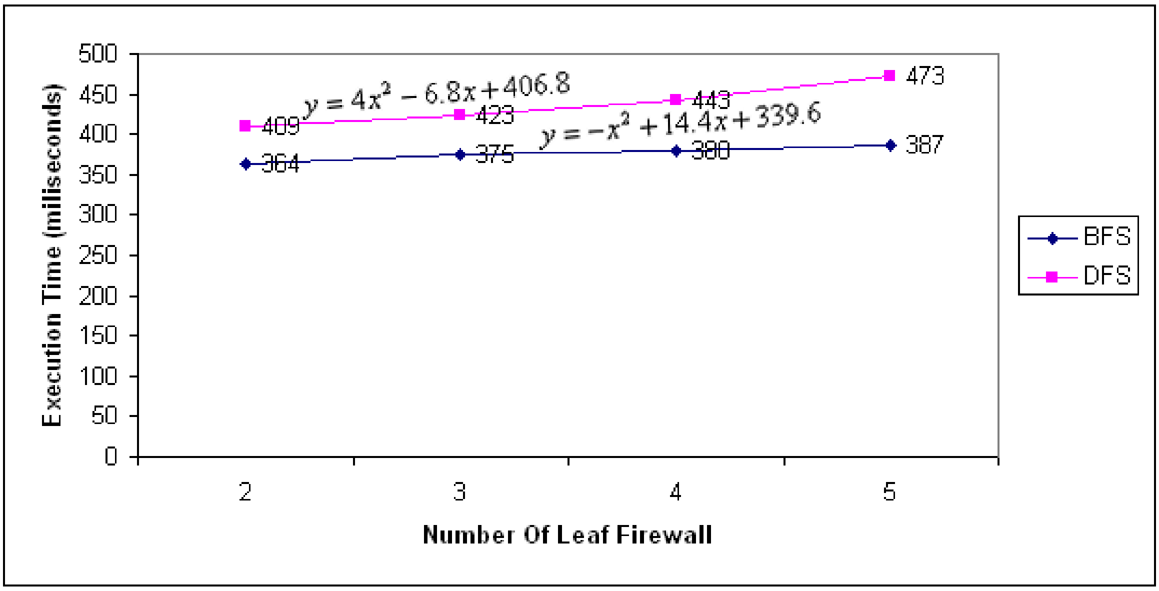

Figure 4.22.Processing Time for 3 Subnets

Equation of the BFS line is $y = -x^2 + 14.4x + 339.6$ however DFS line equation is $y = 4x^2 - 6.8x + 406.8$.

All equations presented in this chapter are generated on the curve fitting tool of Matlab version 7.3.0.267 (R2006b). Curve fitting is a useful tool for representing a data set in a linear or quadratic fashion. Matlab has function, polyfit, which can quickly and easily fit a polynomial to set of data points. Polyfit is computed a least squares polynomial for a given set of data. Polyfit actually generates the coefficients of the polynomial (which can be used to simulate a curve to fit the data) according to the degree specified.

p = polyfit(x,y,n) finds the coefficients of a polynomial p(x) of degree n that fits the data, p(x(i)) to y(i), in a least squares sense. The result p is a row vector of length n+1 containing the polynomial coefficients in descending powers. For a polynomial $p(x) = p_1 x^n + p_2 x^{n-1} + ... + p_n x + p_{n+1}$, the polyfit function provides the solution that minimizes the residuals (The MathWorks, 2008).

CHAPTER 5

CONCLUSION AND FUTURE WORK

The main objective of this thesis is to design a distributed firewall architecture and implement a windows administration tool for this architecture in order to keep track of some certain actions (Create, Read, Update, and Delete) that are performed on the policy rule sets of the distributed firewalls. The most important feature of thesis is that firewall network topology is not given as defined by human. Implementing two different algorithms, which are Depth-First Search and Breadth-First Search, distributed firewall architecture are created and implementing hearth-beat method topology is updated periodically.

Using Distributed Firewall Administration Tool, based on Depth-First Search and Breadth-First Search algorithms for discovering distributed architecture, both balanced or unbalanced processing time for 2 subnets and 3 subnets Depth-First Search algorithm processing time higher than Breadth-First Search algorithm for both subnet type.

The concept of distributed firewall is rather new and as an on-the-shelf-product is not available for windows platform on the market. Distributed firewalls can be categorized as follows domain firewall, subnet firewall, leaf firewall. These firewalls can communicate with each other according to the distributed architecture. The outcome of this work can be transformed into a product for a large network so it can be effective solution for network security.

Distributed firewall administration tool does not have any authentication mechanism between firewalls. According to the proposed architecture parent firewalls send policy rules to their leaf firewalls for providing the consistency overall security policy so during this operation authentication mechanism can be implemented security risks.

REFERENCES

Al-Shaer, E.S., Hamed, H.H., Boutaba, R. and H. Masum. 2005. Conflict Classification and Analysis of Distributed Firewall Policies. *IEEE Journal on Selected Areas in Communications* 23(10): 2069-2084.

Al-Shaer,E.S.,Abdel-Wahab,H.,and. 1999.Hifi:A newMonitoringArchitecture for Distributed Systems Management.In*International Conference onDistributed Computing Systems, Austin, Texas*.

Al-Shaer, E. and Hazem Hamed. 2004. Discovery of Policy Anomalies in Distributed Firewalls, *Proceedings of IEEE INFOCOM'04*.

Al-Shaer, E. and Hazem Hamed. 2003. Firewall Policy Advisor for anomaly Detection and Rule Editing. *IEEE/IFIP Integrated Management IM'2003.*

Awerbuch, B. and R.G., Gallager. 1987. A new distributed algorithm to find breadth first search trees. *IEEE Transactions on Information Theory* 33: 315–322.

Bellovin, S. M. 1999. Distributed Firewalls. *Login*; *special issue on security*.

Chapman, Brent and Elizabeth Zwicky, eds. 1995. *Building Internet Firewalls*. Cambridge: Orielly & Associates Inc.

CodeProject, Developer. 2008. FilterHookDriver. http://www.codeproject.com/filterhook (accessed September 12, 2008).

Dimitrakos, T. and I. Djordjevic. 2002. Policy-Driven Access Control over a Distributed Firewall Architecture. *Proc IEEE Policy Workshop 2002*.

Grant, J., Attfield, P. and K. Armstrong. 2001. Distributed Firewall Technology. *EWA,CANADA.*

Gutin, G. and Jensen J. 2001.*Diagraphstheory*,algorithms and applications.London:College Press.

Ingham, K. and S.Forrest.2002.A history and survey of network firewalls. *Technical Report 2002-37, University of New Mexico Computer Science Department*.

Ioannidis, S., Keromytis, A.D., Bellovin, S.M. and J.M. Smith. 2000. Implementing Distributed Firewall. In *Proceedings of the ACM Conference on Computer and Communication Security, Athens, Greece.*

Kurose, James F. and W.R. Keith . 2003. *Computer Networking A Top-Down Approach Featuring the Internet*. New York: Addison-Wesley.

Li, W. 2000. Distributed Firewall. *GeoInformatica*. 4(3):253.

Linux, Network Administrators Guide. 2008. What Is a Firewall?. http://www.faqs.org/docs/linux_network/x-087-2-firewall.introduction.html (accessed October 19, 2008).

Markham, T. And C. Payne. 2001. Security at the Network Edge: A Distributed Firewall Architecture. *Anaheim, CA, USA.*

Microsoft TechNet. 2008. Windows Firewall. http://technet.microsoft.com/en-us/library/cc755604.aspx#BKMK_architecture (accessed February 4, 2008).

Microsoft Developer Network. 2008. Filter Hook Driver. http://msdn.microsoft.com/ en-us/library/aa504969.aspx (accessed February18, 2008).

Payne, C. and T.Markham. 2001 . Architecture and applications for a distributed embedded firewall. *In 17th Annual Computer Security Applications Conference, IEEE Computer Society.*

Roedig, U., Ackermann, R.,Rensing, C. and R.Steinmetz. 2000 .A Distributed Firewall for Multimedia Applications. *In Proceedings of the Workshop Sicherheit in Mediendaten*,Darmstadt, Germany.

Smith, R. And Y. Chen. 2003 . Cascade of Distributed and Cooperating Firewalls in a Securew Data Network. *IEEE Transactions on Knowledge and Data Engineeting*, Vol.15, NO.5,pp. 1307-1315.

Sahni, S. 1998 . *Data Structures, Algoritms, and Applications in C++. Singapore: McGraw-Hill.*

Smith, R. and S. Bhattacharya. 1999 .A Protocol and Simulation for Distributed Communicating Firewalls.

Stepanek, R. 2001.Distributed Firewalls.Seminar on Network Security.*Helsinki University of Technology*,Finland.

TheMathWorks,Matlab.2008. Polyfit.http://www.mathworks.com/access/helpdesk/help/techdoc/ref/polyfit.html (accessed December 27,2008)

Wack, J., Cutler, K. and J. Pole. 2002. Guidelines on Firewalls and Firewall Policy. *NIST Special Publication* 800-41.

Wikipedia, The Free Encyclopedia. 2008. Breadth-first search. http://en.wikipedia.org/wiki/Breadth-first_search (accessed August 9, 2008).

Wikipedia, The Free Encyclopedia. 2008. Dreadth-first search. http://en.wikipedia.org/wiki/Dreadth-first_search (accessed August 3, 2008).