ACCESS MONITORING SYSTEM FOR DISTRIBUTED FIREWALL POLICIES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
Izmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Oğuzhan ÇAKI

March 2008
İZMİR

We approve the thesis of **Oğuzhan ÇAKI**

Assist. Prof. Dr. Tuğkan TUĞLULAR
Supervisor

Prof. Dr. Şaban EREN
Committee Member

Dr. Serap ATAY
Committee Member

21.03.2008

Date

Prof. Dr. Sıtkı AYTAÇ
Head of the Computer Engineering 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. Tugkan TUGLULAR, for his guidance, patience and encouragement. His valuable support and confidence have been the driving force of this thesis work.

I would also like to thank Fusun Cetin for her help and support in the experiments and in development of graphical user interface parts of the application.

Finally, I should thank to my parents who always supported me throughout my education in my graduate study.

ABSTRACT

ACCESS MONITORING SYSTEM FOR DISTRIBUTED FIREWALL POLICIES

Internet has provided several benefits in terms of information sharing. However, Internet is an insecure environment that can cause threats to private networks. As a result, network security becomes a critical issue. One of the important tools used in network security is firewall. Firewalls protect a private network from external threats by restricting network traffic according to predefined security rules. Basically, firewalls apply these rules to each packet that passes over them. Distributed firewalls are a new approach to firewall to overcome some drawbacks of traditional firewalls. Distributed firewall design is based on the idea of enforcing the policy rules at the endpoints rather than a single entry point to network. Management of policy rules is a critical issue in both traditional and distributed firewalls. We propose a monitoring application for distributed firewall policies to keep track of actions (create, read. update, delete) performed on policy rule set. The resulting data produced by the monitoring application will be very helpful in policy management process.

ÖZET

DAĞITIK GÜVENLİK POLİTİKALARI İÇİN ERİŞİM TAKİP SİSTEMİ

İnternet bilgi paylaşımı açısından birçok faydalar sağlamaktadır. Fakat İnternet özel ağları tehdit edebilecek, güvenlik açısından eksiklikleri olan bir ortamdır. Ateş duvarları ağ güvenliğini sağlamada kullanılan önemli bir araçtır. Ateş duvarları önceden belirlenmiş güvenlik kurallarına göre ağ trafiğini düzenleyerek özel ağları dışarıdan gelebilecek saldırılara karşı korur. Temel olarak bir ateş duvarı güvenlik kurallarını üzerinden geçen her ağ paketine uygular. Dağıtık ateş duvarları geleneksel ateş duvarlarının birtakım eksikliklerini gidermek için ortaya atılmış bir yaklaşımdır. Dağıtık ateş duvarı tasarımı güvenlik kurallarını ağa giriş noktasında uygulamak yerine ağın içindeki bilgisayar üzerinde uygulama esasına dayanır. Güvenlik kurallarının yönetimi hem geleneksel hem de dağıtık ateş duvarları için çok önemli bir konudur. Bu çalışmada dağıtık ateş duvarı güvenlik politikaları için bir takip sistemi geliştirmek amaçlanmaktadır. Bu uygulama tarafından üretilecek sonuçlar güvenlik kurallarının doğru bir şekilde yönetilmesinde yardımcı olacaktır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

As a result of rapid developments in computer networks, nowadays it is possible to share and collect information for business, personal or academic purposes across different networks. Especially, the introduction of Internet made the information sharing possible in the worldwide. It is a fact that Internet has so many benefits and made the life very easy. On the other hand, it should be kept in mind that Internet also has some threats. Basically, these threats come from the Internet attacks.

The aim of these attacks is basically to gain access to private information and resources, which are inside a personal computer or private network, in an unauthorized way. In the past, where Internet is not so widespread, network security is not so critical issue. However, today network security should be handled very carefully.

Today, there are several techniques and tools that are used to provide a secure network environment. Firewalls are one of the well-known tools that are very helpful in network security. 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 played 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. Basically, a distributed firewall aims to provide the same functionality as the traditional ones in terms of network security. However, their way of

working is a bit different. In a distributed firewall, security policy is enforced at the each individual network endpoints rather than at a single place (which is the case for traditional ones).

This thesis introduces a monitoring application that is running on a distributed environment to keep track of some certain events performed on policy rules. The thesis is composed of six chapters. Chapter 2 gives some background information about firewalls and distributed firewalls, their types, roles in network security, comparison to each other, etc. Chapter 3 defines the details of proposed distributed firewall architecture on which the monitoring application will run. The details of monitoring application are explained in chapter 4. Chapter 5 explains the experimental firewall setup that is used to test and verify the monitoring application is explained. Additionally, the results that are created by the monitoring application on the experimental setup are presented in this chapter.

CHAPTER 2

DISTRIBUTED FIREWALLS

In this chapter, an overview of firewall concept and the role of firewalls in network security will be explained. In addition, the idea of distributed firewalls will be examined.

2.1. Firewall Overview

As a result of rapid developments in Internet technologies, it has become really easy to access, share and publish information in both personal and business level. It is a fact that Internet provides several benefits for information sharing, however it should be taken into account that there are significant risks in the Internet world. Connecting a private PC or network to Internet allows the ones inside that network to access external resources and Internet services. On the other hand, it also allows outsiders to access that private PC or network's internal resources and private information. In order to deal with such kind of problems efficiently, managing network security is a really key point.

Firewalls are important 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 (Linux Network Administrators Guide 2007).

Firewalls are the separation point between the Internet and private network. Basically, they examine the incoming network traffic in order to prevent the harmful traffic and Internet attacks. They can also detect and prevent unauthorized outgoing traffic. This situation emerges when an intruder from the internal network generates non-legitimate traffic. Firewalls can detect such kind of malicious activities of intruders (Kurose and Ross 2003).

A firewall can be implemented on different hardware platforms. A router, a personal computer or a collection of routers / personal computers can be configured to be used as a firewall. Whatever the hardware is, all network traffic should be managed in a way that it will pass over the firewall so that it will be possible to force predefined security policy (Chapman and Zwicky 2000).

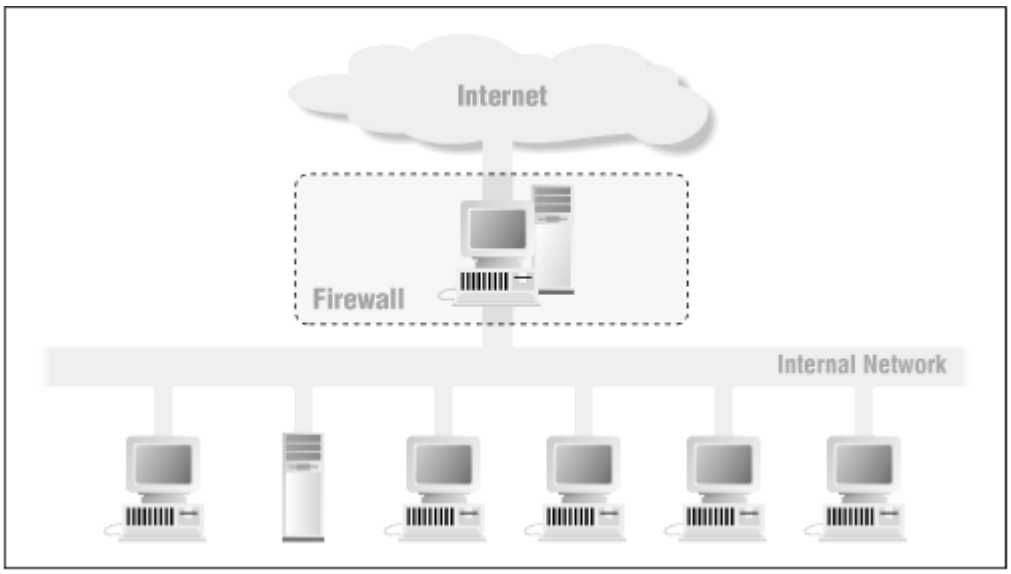


Figure 2.1. A firewall's location in the network
(Source: Chapman and Zwicky 2000)

When defining a network security policy, first the procedures to protect the network and its contents should be defined. From this point of view, a network security policy has an important role in enforcing the whole security policy defined by an organization.

As stated, a firewall's fundamental functionality is to regulate the network traffic between different computer networks having different trust levels to enforce the predefined security policy. A very common example for this situation is Internet (having no trust level) and a private network (has a higher trust level). It is also possible to have another network between Internet and private network having an intermediate trust level. Generally, such kind of networks is called as perimeter network or Demilitarized Zone (DMZ). Perimeter networks or DMZs are useful since they provide an additional layer of security (Wikipedia 2007).

2.1.1. Perimeter Networks

Each network can contain several perimeter networks. In order to have a successful network security perimeter, the firewall server must act as the gateway for all communications between trusted and untrusted / unknown networks. In general, it is

possible to talk about three types of perimeter networks: the outmost perimeter, internal perimeters and the innermost perimeter (Cisco Documentation 2007).

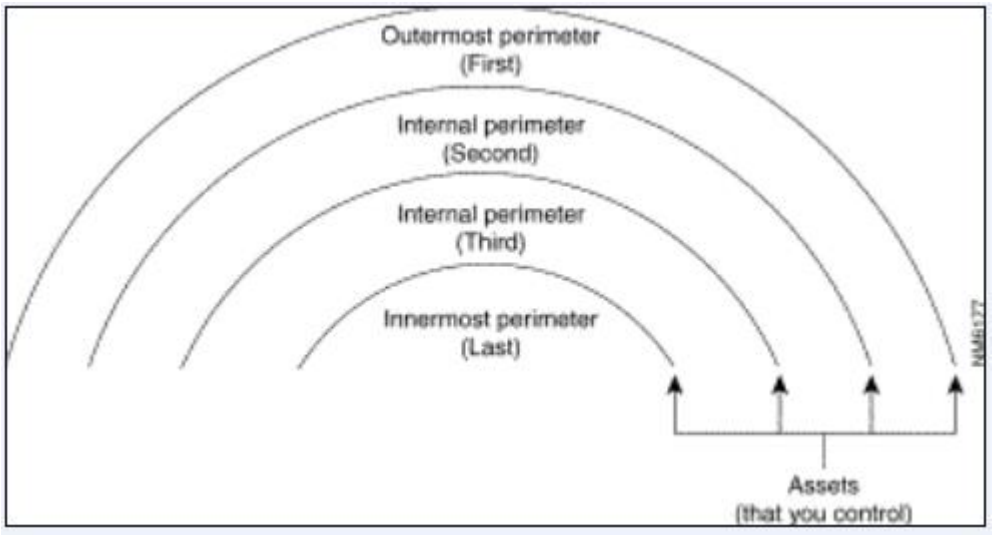


Figure 2.2. Perimeter networks
(Source: Cisco Documentation 2007)

The outmost perimeter network is the border between the controllable zone and the zone out of control. In general, this border is the place where the router, which is to separate the private network from the Internet service providers network, is deployed. Internal perimeter networks are additional boundaries in which any other security mechanisms are placed. Intranet firewalls, filtering routers, etc. are examples of these security mechanisms. The outmost perimeter network is the most insecure part of the network structure. In general, this area is the place where routers, firewall servers and public servers like HTTP, FTP, etc. are located. From an external network's point of view, this area is the key point on which all traffic between two networks should pass. Since this area is the easiest one to access for outsiders, it is the main attack point to gain access to internal networks (Cisco Documentation 2007).

2.1.2. Trusted Networks

Trusted networks are the ones inside the network security perimeter, meaning that these are the ones that need to be protected from external world. After configuring the

firewall server, the trusted networks contain the firewall server and all networks behind the firewall server. An exceptional case to this situation is virtual private networks (VPNs), which allows data transmission over an untrusted network (Cisco Documentation 2007).

2.1.3. Untrusted Networks

The networks that fall outside of the security parameter and external to firewall server are known as untrusted networks. It is possible to control trusted networks but you cannot control untrusted networks since you do not have any control over the administration or security policies for these networks. A private network wants to communicate with untrusted networks but there should be a protection against the threats that can come from the untrusted world (Cisco Documentation 2007).

While configuring the firewall server, it is important to identify untrusted networks from which incoming requests are accepted (Cisco Documentation 2007).

2.1.4. Unknown Networks

The networks that are not trusted or untrusted are defined as unknown networks. Since it is not possible to explicitly tell the firewall server that this networks is trusted or untrusted, they are unknown to firewall. Unknown networks fall outside of the security perimeter. All non-trusted networks are treated as unknown by default and firewall applies the security policy that is applied to Internet node. However, it is possible to identify unknown networks below the Internet node and apply more specific policies to those (Cisco Documentation 2007).

2.2. Firewall Technologies

Firewall technology is a recent but rapidly developing area. The first firewall architecture, which is called packet filters, has appeared in 1985. After that there has been several studies on this field and new architectures / designs have been introduced.

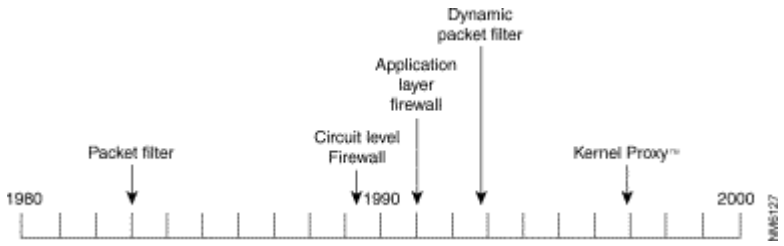


Figure 2.3. Time Line of Firewall Architectures
(Source: Cisco Documentation 2007)

The evolution of firewall technologies is examined in the rest of this section.

2.2.1. Packet Filters

Packet filter firewall technology is known as the first-generation firewall technology. A packet filter firewall basically analyzes network traffic at the transport protocol layer. Packet filtering is based on the some predefined rules that define what kind of data flow is allowed via the firewall. Each incoming network packets are examined whether to find out that they match one of the rules in the filtering rule set or not. If the packet matches one of the rules, than an appropriate action will be performed: The packet may be silently discarded without informing the source or an error message may be sent to the source. Alternatively, the packet filter may allow the packet to pass through. The action that will be performed is specified while defining the filtering rules. This type of packet filtering does not care whether a packet belongs to an existing traffic stream / connection. In other words, it does not store information on connection state. The only criteria to filter packets are the information stored in the packet itself. Basically, this information includes packets source and destination addresses, its protocol, the protocol type (TCP, UDP), or the port number (Cisco Documentation 2007).

Obviously, packet filter firewalls have some certain advantages. First of all, since they perform fewer and less complicated operations, they are generally faster than other firewall technologies. Additionally, with the help of a simple set of filtering rules, it is possible to protect whole network by denying connections between specific outside sources and internal computers. And finally, it is not needed to configure client computers in a specific way for packet filters. All of the work is handled by the packet filter itself (Cisco Documentation 2007).

On the other hand, besides the advantages mentioned above, packet-filtering firewalls have some insufficiencies. First of all, they do not understand application layer protocols. For even most basic services (like PUT and GET commands of FTP), they cannot restrict access to protocol subsets. Because of these reasons, they cannot provide security for higher levels of network protocol stack. In addition, since they are stateless, they do not store any information about a session or application derived information. Another disadvantage of packet filters is that they cannot restrict what type information is passed from internal computers to outside services on the firewall server. They only restrict what information can go to inside of the private network. Therefore, they cannot restrict intruders' accesses to the services on the firewall server. They also do not have audit event generation and alerting mechanisms. And finally, it might be difficult to test and verify "access" and "deny" rules are working properly (Cisco Documentation 2007).

2.2.2. Stateful Filters

Opposing to first generation firewalls, the second-generation firewalls do not simply control the contents of each packet individually by ignoring their placement in the packet series. Instead, they store all connections passing through the firewall, and can determine whether a packet is the start of a new connection, or part of an existing connection. If a packet is not a part of existing connection or does not belong to a start of new connection, then it is refused by the firewall. Since they keep information about the state of connections, this type of firewalls are generally called as stateful firewall filters. Alternatively, they are called as dynamic packet filters since they can adopt their packet handlings dynamically according to network traffic passing though them (Wikipedia 2007).

As mentioned, stateful filters controls that a packet is either a connection request or a data packet belonging to an existing connection. In order to validate a connection session, a stateful firewall examines each connection request to guarantee that it follows a reasonable handshake for the transport layer protocol that is in use (TCP is the most widely used protocol that has an handshake mechanism). Additionally, data packets are not forwarded until the handshake is completed successfully (Cisco Documentation 2007).

Stateful firewalls maintain a table of valid connections including complete session state and sequencing information. This state table is used to validate network traffic and removed when the connection is terminated.

The state table generally includes the following information that identifies a connection:

- A unique session identifier for the connection
- The state of the connection (handshake, established, closing, etc.)
- The sequencing information
- The source IP address
- The destination IP address
- The physical network interface through with the packet arrives
- The physical network interface through which the packet goes out

With the help of this information, the stateful filters control the header information for each network packet in order to decide whether the source machine has permission to send data to the destination machine or whether the destination machine has permission to receive that data (Cisco Documentation 2007).

As an enhancement to packet filters, stateful firewalls share advantages and disadvantages of packet filters. However, the addition of state table to the system makes stateful firewalls more secure that the ordinary packet filters.

2.2.3. Application Layer Firewalls

Basically, an application layer firewall, known as third generation firewall technology, operates at the application layer of a protocol stack. It controls the data carried in network packets at the application layer and manages the whole connection state and sequencing information. Besides, an application layer firewall can also validate other security items, like user passwords, service requests, etc, that are only available in application layer data (Cisco Documentation 2007).

Generally, application layer firewalls make use of proxy services. Proxy services are specialized applications or server programs that take users' request for Internet services and forward them to the actual services. Proxy services are specific to the protocol that they

are designed to forward. They can provide enhanced access control, detailed checks for valid data and generate audit records for the traffic they are transferring (Cisco Documentation 2007).

Proxy services are developed on top of the firewall host machine's network stack and operate in the application space of the operating system. Therefore, each network packet should travel through the low-level protocols in the kernel before reaching to the application space for a inspection of the packet headers and packet data by the proxies. Then, each packet should travel back to kernel and stack for distribution. As a result of this process, proxy services are a bit slower (Cisco Documentation 2007).

Application proxies are generally implemented as a single executable composed of two main components: proxy server and proxy client. A proxy server is the end server for all connection requests coming from a trusted network. In other words, internal users are not allowed to directly communicate to other services on the Internet. Instead, all traffic between internal users and outside world (Internet) goes over the proxy server. Once a connection request comes from an internal user for an external service, proxy server evaluates the request and decides to allow or refuse it. This decision is performed using some predefined set of rules for the individual network service. Proxy servers know about the protocol of the service they are evaluating, so they only allow packets that suit the definitions of that protocol. Additionally, proxy servers can provide some more advantages including detailed audit records of session information, user authentication, and caching (Cisco Documentation 2007).

A proxy client is the part of an application that talks to the real server on the external network on behalf of the real client. When a request comes from a real client to proxy server, the proxy server either accepts or refuses the request. If the request is accepted, then proxy server forwards it to the proxy client. Then it is responsibility of the proxy client to communicate with the real server on behalf of the real client (Cisco Documentation 2007).

2.2.4. Dynamic Packet Filters

Dynamic packet filters, which is known as fourth generation firewall technology, allows dynamic change of security policy while running. This type of firewall technology is most useful for UDP transport protocol, which is generally used for limited information requests and queries in the application layer protocol exchanges. All UDP packets that cross the security perimeter are associated with a virtual connection. If a response packet is sent to the original requester, then the setup of virtual connection is completed and the packet is allowed to pass over the firewall server. If no response packet is created for a certain period of time, the virtual connection becomes invalid (Wikipedia 2007).

Dynamic packet filters shares the advantages and disadvantages of first generation firewalls. The only exceptional case is that they do not allow unsolicited UDP packets onto the internal network. Once a UDP request packet is send from the internal network to an untrusted host, the firewall server only allows packets that are a response to that request to the source of request. The response packet that is coming back to internal network should contain some certain information. This information includes a destination address matching to the source address of the request, a transport layer destination port that matches to source port number and the same transport layer protocol type. With the help of this feature, it is possible to allow application layer protocols like DNS to operate across the networks' security perimeter. An internal DNS server can communicate to other DNS servers on the Internet to gather address information for unknown hosts. DNS servers make use of TCP connection or UDP virtual connection to make such kind of requests (Cisco Documentation 2007).

2.2.5. Network Address Translation

Network address translation (NAT in short) is a technique of computer networking that allows using two different sets of IP addresses for internal and external networks. Basically, it involves a process of re-writing the source and destination IP addresses of IP packets while they are passing through a router. When an external host sends a packet to internal network, the NAT system converts the destination address that refers to a valid

address for internal network. Similarly, the source address of the network packet going from inside to outside is modified so that it contains a valid source address for the external world. This functionality makes NAT an effective technology to hide the network-addressing schema of the internal network that is located behind the firewall. Besides, NAT forces firewall's control over outbound connections. This is because the addresses of hosts will not work on the external network, so they need to pass through the NAT system. Any connection that bypasses the NAT system will not work (Kurose and Ross 2003).

On the other hand, NAT have some drawbacks. One of the major drawbacks of NAT is that it is not possible to make an end-to-end IP tracing. Additionally, embedded IP addresses (used by FTP for example) require special handling. The hidden addresses located in different part of packet header should be processed and converted to valid addresses by NAT system. To perform such an operation, NAT system should have enough knowledge of protocol (Kurose and Ross 2003).

2.2.6. Virtual Private Networks

Virtual private network (VPN) is a technique to use employ and integrity protection that allows using a public network as if it is a private network. The VPN is based on the idea of combining the benefits of a public network (cheap, widely available, etc.) with the benefits of a private network. The network traffic running over a VPN is encrypted, integrity protected and encapsulated into new packets. At the opposing site, the encapsulation is reversed, the integrity is controlled and the packets are decrypted back to original. The encryption can be done in two ways. The first one is to perform encryption as a transport method meaning that the traffic is encrypted when it is generated at host. The second approach is using a tunnel mechanism meaning that the encryption and decryption is performed at a location between source and destination (Chapman and Zwicky 2000).

VPNs are not exactly a firewall technology but they affect the firewalls way of working in some ways. While using a VPN, one should be careful about its interaction with firewall. There are some cases in which the firewall cannot control the traffic coming through the VPN. Additionally, firewalls can be convenient places to add VPN features (Chapman and Zwicky 2000).

2.3. Firewalls Role in Network Security

There is several attack types that a private machine or network can face on the Internet world. Firewalls can play an important role in detecting and preventing such kind of attacks if they are configured properly. Within this section, the type of attacks and firewalls role in detection / prevention of these attacks will be examined.

2.3.1. Types of Attacks

2.3.1.1. Mapping

Attackers would like to know some certain information about that network before attacking to it. The information that an attacker interested in could be IP addresses of machines on the network, the operating system running on that machines, the services that they offer, etc. With the help of such information, the attacks can be designed in a way such that they are more focused and less likely to cause alarm. The process of collecting this information is called as mapping (Kurose and Ross 2003).

A simple way of detecting the IP addresses on the network is to use ping program to find out whether a machine responds to ping message or not.

Port scanning can be used to find out the services (HTTP, FTP, etc.) offered by the machine. Simply, this technique includes sequentially contacting port numbers on a machine and analyzing the responses coming from it. The contact can be performed via TCP connection request or via simple UDP datagram. Nmap that is a well-known tool for network exploration and security auditing can perform port scanning. Firewalls have an important role in detecting mapping and port scanning activities. Many firewalls available in the market can find out such kind of harmful activity (Kurose and Ross 2003).

2.3.1.2. Packet Sniffing

Packet sniffer is a program running on a machine that is attached to a network and monitors all data link-layer frames passing over the machine's network adapter. In an Ethernet LAN, packet sniffer receives all of frames that are transmitted from / to all hosts on the LAN. Any machine with an Ethernet card can behave as a packet sniffer if the Ethernet adapter is configured as promiscuous mode to receive all passing Ethernet frames. Packet sniffing can be both a very beneficial and harmful tool for the network security. It can be beneficial for network administrators in network monitoring and management. On the other hand, hackers can use packet-sniffing tools to extract user accounts and passwords from the network traffic (Kurose and Ross 2003).

2.3.1.3. Intrusion

Another common attack type is intrusion. Basically, with the help of intrusions, external users can use a computer as if they are legitimate users. There are several ways of getting access to a machine: a straightforward guesswork (try to guess account information necessary to login to the target machine), or to intricate ways to get in without knowing the account information, etc. Firewalls are beneficial in preventing intrusions in several ways. Basically, they can block all connection requests to the system without knowing an account name and password. If a firewall is configured properly, it can reduce the number of accounts that can be accessible from the outside so that the risk of social engineering or guesswork is decreased. Alternatively, firewalls can be configured to use one-time passwords that prevent guessing attacks. Even if these passwords are not used, a firewall will create the log of attempts of connection to the system and these logs will be helpful in detection of attacks (Chapman and Zwicky 2000).

2.3.1.4. Spoofing

A machine that is connected to Internet inevitably sends to IP datagrams into the network that contains information about sender's IP address and some upper-layer data. A

user that has a full control of device's software and operating system can change the device's protocols to put an arbitrary IP address into datagrams source address field. This action is known as IP spoofing. In general, IP spoofing is used in denial-of-service attacks in order to hide the source of attack. It becomes really difficult to find the host that sent the datagram if the IP address of that datagram is spoofed. Spoofing can easily be prevented by using routers that can perform ingress filtering. Ingress filtering is to check the IP addresses of incoming datagrams and detect if the source address is in the network address range. It is possible to perform such a check at the edge of network that is a corporate gateway or firewall (Kurose and Ross 2003).

2.3.1.5. Denial-of-Service Attacks

Denial-of-service (DoS) attacks involve a large variety of threats. As its name indicates, the main aim of a DoS attack is to prevent a network, host or another network element being used by legitimate users. Basically, a DoS attack creates so much workload on the network infrastructure that is being attacked so that the legitimate tasks cannot be performed on that network (Kurose and Ross 2003).

A DoS attack can have several versions. One of the well-known attack types is called as flooding or SYN flooding. In SYN flooding, the attacker sends a heavy load of TCP SYN packets to a server. Each of these packets contains a spoofed source IP address. Since the spoofed SYN packets contains a reasonable source address, the server cannot differentiate between a real SYN and spoofed SYN. Therefore, the server continues to second step of the TCP handshake, allocates necessary data structures and updates state accordingly. However, the attacker cannot complete the third step of the three-way handshake, so that the number of partially open connections increase. As a result, the load of SYN packets that need to be processed consumes all of the memory available in the server and server stuck (Kurose and Ross 2003).

The smurf attack is another type of DoS attack. It again creates very heavy network traffic on the target site by using spoofed broadcast of ping messages. In a smurf attack, the attacker floods a large number of ICMP echo (ping) messages to IP broadcast addresses. Each ping message contains a spoofed source IP address that is the IP address of the target

machine. If the routing device delivers the IP broadcast to all hosts, hosts receiving the ping message on that IP network will reply to it and send a reply message to the spoofed IP address. These reply messages will create a heavy load of traffic on the victim machine so that it is not available for the legitimate operations (CERT Advisory 2007).

Another form of DoS attacks is distributed DoS attack. In a distributed DoS attack, the attacker first gains access to several user accounts on the Internet. Then, by using that accounts, the attacker installs and runs a slave program that waits the commands of a master program at each site. At the next step, the master programs triggers the slave programs to start a denial-of-service attack to the same target (Kurose and Ross 2003).

It is not so easy to avoid DoS attacks since it is difficult to differentiate between a valid datagram and a spoofed one. Additionally, it is difficult to find out the attack source because of the IP spoofing. However, a well-designed firewall will not be prone to DoS attacks and will prevent them reaching to the internal sources.

2.4. Drawbacks of Conventional Firewalls

Although they provide several benefits for network security, traditional firewalls have some drawbacks as stated in (Bellovin 1999, Li 2000, Ioannidis, et al. 2000, Gatus, et al. 2004). These drawbacks can be summarized as follows.

2.4.1. Topological Dependence

In order to define an efficient security policy for conventional firewalls, it is necessary to know network topology including location of all network egress points and internal network topology. If network topology is changing so frequently, then this may introduce scalability issues since the system administrator must update firewall policies constantly (Gatus, et al. 2004).

Additionally, knowledge of topological dependence is necessary while dividing a network as trusted and untrusted. To detect whether a machine is trusted or untrusted depends on whether that machine is physically attached to that network or not. For instance, consider a laptop that has carried out of a company's internal network. Since the

machine is not physically attached to internal network, then the company's firewall cannot protect that machine. On the other hand, a visitors laptop that is physically attached to company's network may be considered as insider and may have a chance to access some internal resources and information (Li 2000).

At the moment, it is possible to produce solutions both of the cases mentioned above. In the first case, with the help of a connection through a secure channel like VPN or SSH, it is possible to connect the laptop to company's network. This solution may add extra cost and connection overhead. In the second case, multiple firewalls can be used to divide the network into smaller parts that have different security levels. But this solution increases the network complexity and makes network administration more difficult (Li 2000).

2.4.2. Single Point of Failure

All the incoming and outgoing traffic should pass over a single point in order to enforce security policies. Although the speed of CPUs has increased dramatically in recent years, the speed of networks does not change as rapid as CPU. In addition to this gap between network and CPU speed, protocols like IPSec, which requires some computational power, become quite widely used in recent years. Since such kind of protocols increase the amount of traffic, the filtering of that traffic can cause processing overheads for the firewall. This congestion on a single point can significantly affect network throughput (Gatus, et al. 2004, Li 2000).

2.4.3. Implicit Trust of Insiders

As mentioned, traditional firewalls divide network into two artificial parts as trusted and untrusted. This division is based on the idea that all of the hosts that is inside the trusted network can be trusted. However, this idea has been invalidated quite some time since it may be desired to allow some specific individuals or remote networks to access to the protected network. As a result, the classical notion of security perimeter mentioned in the previous sections cannot stay unmodified (Ioannidis, et al. 2000).

2.4.4. Multiple Entry Points

Nowadays, large networks are designed to have several entry points for some reasons like performance, failover, etc. In addition, many networks contain internal firewalls to construct different security levels. Such kind of cases makes network administration more challenging and threats the policy consistency.

Apart from these, it is possible to construct an unauthorized network entry point without the knowledge of network administrator. With the use of various types of tunnels, wireless, or dial-up connections, one can establish an access point that bypasses all security mechanisms provided by conventional firewalls (Li 2000).

2.4.5. End-to-End Encryption

End-to-end encryption can threat conventional firewalls since it prevents them checking packet fields that are necessary to perform a packet filtering. End-to-end filtering can be allowed in conventional firewalls in case of existence considerable trust on the users on behalf of network administrator (Ioannidis, et al. 2000).

2.5. A Distributed Approach to Firewall Design

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).

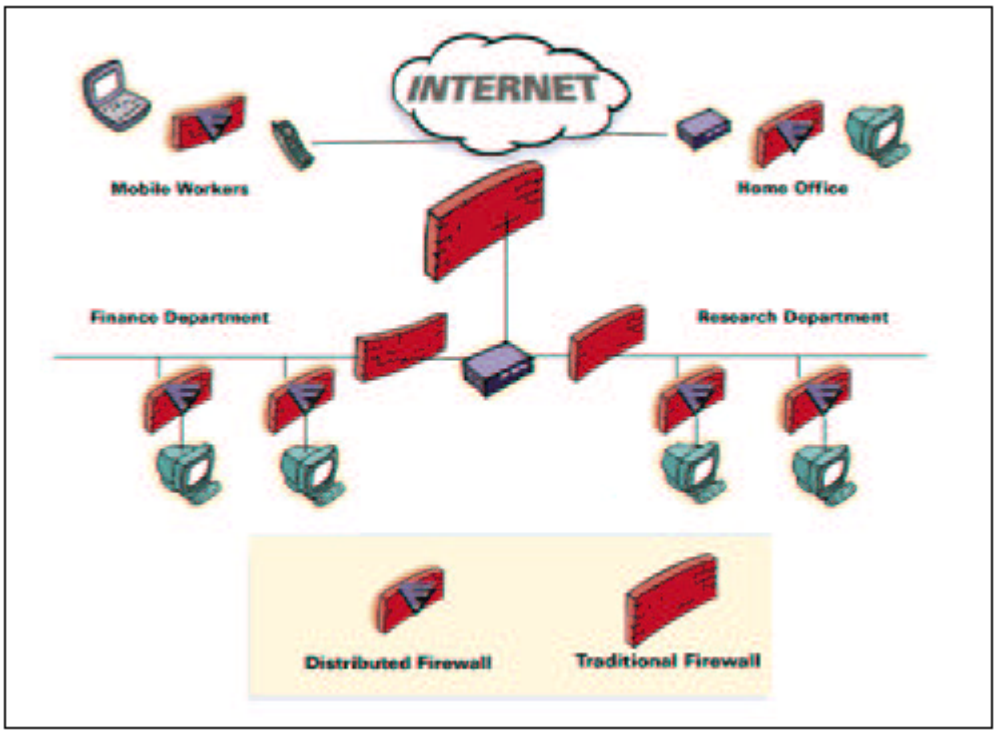


Figure 2.4. A distributed firewall architecture
(Source: Li 2000)

There are three notions on which distributed firewalls based.

- Policy language: This is necessary to define what kind of connections are allowed or rejected.
- System management tools like Microsoft's SMS or ASD.
- Network level encryption mechanisms for TCP/IP such as IPSEC.

Basically, using these notions a distributed firewall system works as follows. The policy language is converted into an internal format using tool or a compiler. The system management tool distributed this policy data to all of the hosts that are protected by the firewall. Then at each host, the incoming packets are accepted or rejected according to both the security policy and cryptographic verification of each sender (Bellovin 1999).

One of the critical points in a distributed firewall design is centralized management of the security policies. For such a large and complex system, centralized management is very important to provide consistency across all firewall devices in the system for enhanced security. Another critical point is to distribute policies in a secure way. To provide this,

there is a need to use security services that guarantee the integrity, confidentially, authenticity of security policies (Gatus, et al. 2004).

Policy Based Network Management (PBNM) is a new network management approach that abstracts the task of network management by providing a system wide approach to management and configuration. Opposing to traditional management systems that focus on device characteristics, PBNM treats the network as a whole and attempt to manage network as a single entity. PBNM performs this abstraction with the help of centralized storage, creation and management of network policies. This is very helpful to provide consistency, efficiency, reliability and a dynamic approach to system wide device configuration. Such kinds of things are very essential when managing security policies in a distributed firewall (Gatus, et al. 2004).

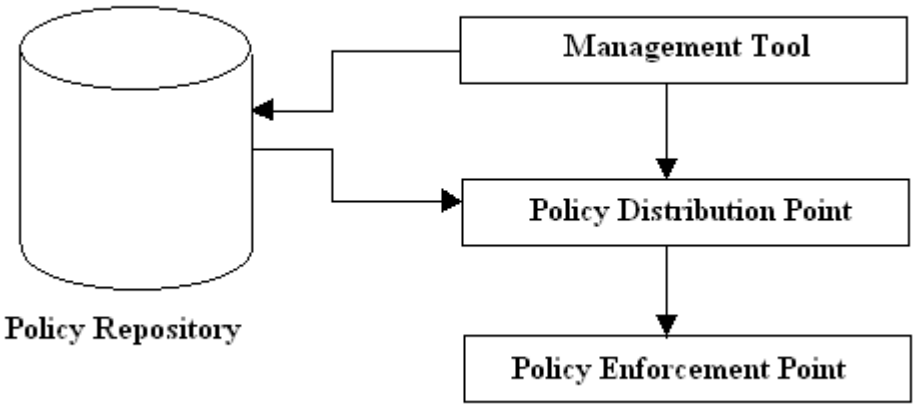


Figure 2.5. Architectural model of a PBNM system
(Source: Gatus, et al. 2004)

2.6. Advantages and Disadvantages of Distributed Firewalls

The introduction of distributed firewalls offered some solutions to the problems cannot be easily handled by conventional firewalls. The advantages of distributed firewalls can be stated as follows (Ioannidis, et al. 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.

- Opposing to conventional firewalls, network security is no more dependent on the single firewall so that problems like performance bottleneck and traffic congestion are resolved. Besides, the load on the traditional firewall is reduced since a large amount of filtering is performed at the end hosts.
- As mentioned earlier, filtering of certain protocols such as FTP are not so easy on a conventional firewall. Such kind of a process is much easier on distributed firewalls since all of the required information is available at the decision point, which is the end host in general.
- The number of outgoing connections does not create so many difficulties in terms of network administration. Adding new links or removing existing links does not affect the network security. Similarly, backdoor connections that are created by insiders intentionally or inadvertently do not create new threats to network security in distributed firewalls.
- As mentioned, in conventional firewalls there is an assumption on that insiders are trustable. However, this assumption is source of several problems. With the distributed firewall architectures, the insiders are no longer treated as *"unconditionally trusted"*. Dividing network into parts having different security levels is much easier with distributed firewalls.
- Security policy rules are distributed and established on an as-needed basis. Only the host that needs to communicate with the external network should determine the relevant policy. This approach dramatically eases the policy updating process and does not require each firewall to maintain the complete policy set. Note that especially for very large networks, the policy rule set may be quite big.
- 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.
- It is not so easy to implement an intrusion detection system in a distributed firewall environment. 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.

2.7. Performance of Distributed Firewalls

As mentioned, distributed firewalls have some advantages and disadvantages compared to traditional firewalls from the architectural point of view. Additionally, it would be better to compare these two in terms of their performance against network attacks.

2.7.1. Service Exposure and Port Scanning

Both the conventional and traditional firewalls are very successful in rejecting requests for inappropriate services. Conventional firewalls reject the request at the network entry point whereas distributed firewalls do it at the host (Bellovin 1999).

2.7.2. Denial of Service Attacks

Policy distributions in distributed firewalls are based on some protocols like IKE or ISAKMP that are used by IPSec. As a result of this, distributed firewalls are prone to DoS attacks. On the other hand, conventional firewalls are also prone to DoS attacks because of their “single point of entry” characteristics.

Both of the firewall types will not be able to offer an effective defense against the smurf attack. If it is possible to change the network topology, both of then can offer some sort of support. It can be possible to locate conventional firewalls at the ISP's POP in which case the attack can be blocked before it reaches the low-bandwidth access line. Distributed firewalls allow a host to have several outgoing connections so that the problem is a bit finessed (Bellovin 1999).

2.7.3. Insider Attacks

Since every host performs their own packet filtering, there may occur serious problems when a host changes its local security settings. However, misbehavior of insiders can also cause problems for conventional firewalls. For example, a host can use SSH to tunnel TCP ports or can use end-to-end encryption to hide the traffic. Overall, it is possible to say that insiders that want to violate security policy can do it for both types of firewalls (Bellovin 1999).

However, in some cases of insider attacks, distributed firewalls can have some advantages over conventional ones. With the distributed firewall architecture it is easier to create smaller groups of users. By this way, it is possible to allow only the authorized user groups to access a resource on the network (Bellovin 1999).

CHAPTER 3

PROPOSED ARCHITECTURE

In this chapter, the distributed firewall architecture on which the proposed monitoring application will run is clarified. Additionally, the database design that each node in the architecture needs to maintain is explained in details.

3.1. Proposed Distributed Firewall Architecture

As proposed by Bellovin and Ioannidis et al, who are the pioneers of the distributed firewall concept, the basic idea on which distributed firewall systems are based is *to decentralize policy enforcement to several nodes in the overall system rather that enforcing the policy at a central location* (Bellovin 1999, Ioannidis, et al 2000). By keeping this fundamental idea in mind, we propose a hierarchically organized distributed firewall system.

The suggested architecture, as indicated by figure 3.2, is hierarchically organized and has a tree-like structure. Each domain has a firewall node at the domain entrance point to control the traffic going inside / outside the domain (domain firewall). Similarly, each subnet in the domain also has a firewall node located at the subnet entrance point for the same purpose (subnet firewall). Each subnet may contain several hosts that are connected to subnet firewall. These hosts (for example PCs) can also manage a firewall to control the network traffic only on that host. It is possible to say that there are three types of firewall and these firewalls are organized in a hierarchical manner. Domain firewalls are the ones that are at the highest level of hierarchy. Subnet firewalls are connected to a domain firewall and located at the lower levels of the hierarchy. Finally, firewalls running on the hosts inside a subnet (leaf firewalls) are the lowest level of the distributed firewall architecture.

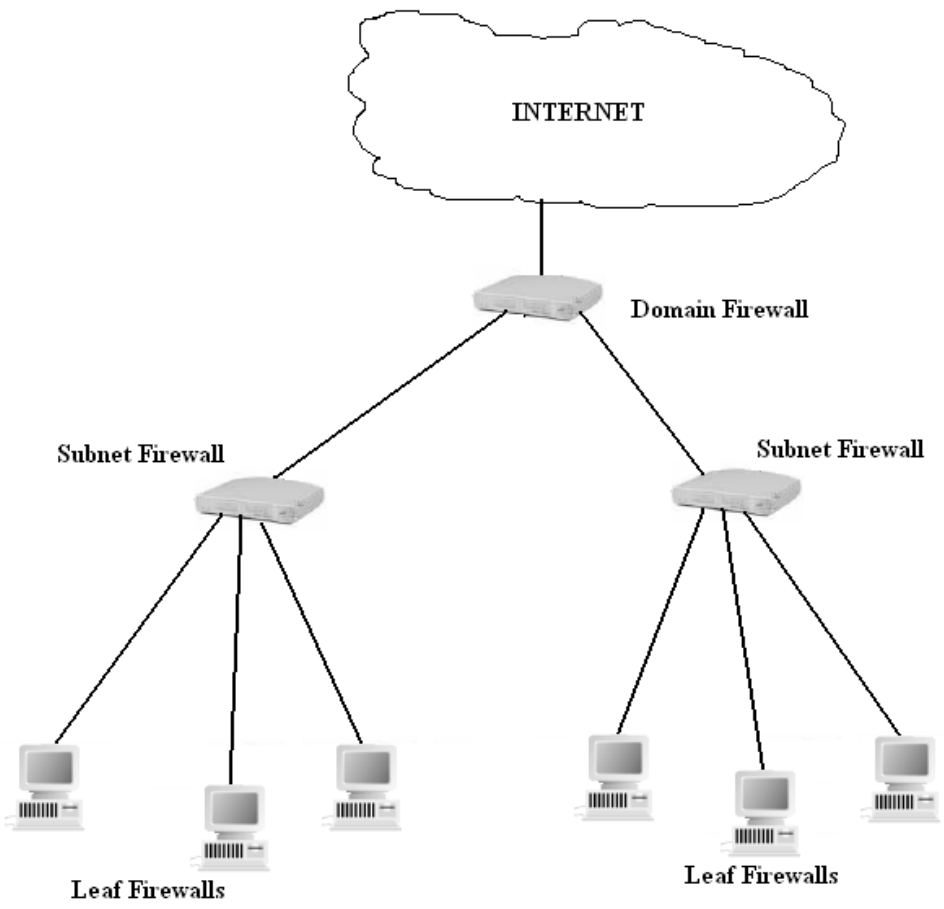


Figure 3.1. Proposed Distributed Firewall architecture

There is a strictly defined communication scheme between these firewall nodes inside the system. Each firewall node in the system can only communicate to another firewall node that sits at a lower level of the hierarchy with one exceptional case, which is explained in the next paragraph. A leaf firewall at the lowest level cannot communicate to any other node in the system. A subnet firewall can communicate to all of the nodes inside that subnet but it 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.

As mentioned, there is an exceptional case where a firewall node can communicate to other nodes at the higher levels of the hierarchy. This is only possible when the higher-

level nodes are on the path that connects a lower level node the Internet as indicated by the figure 3.3. Note that in this case this case a lower level node (for example leaf firewall) sends the communication request to the node that is located at the next level of the hierarchy (for example subnet firewall) until the request reaches to the domain firewall.

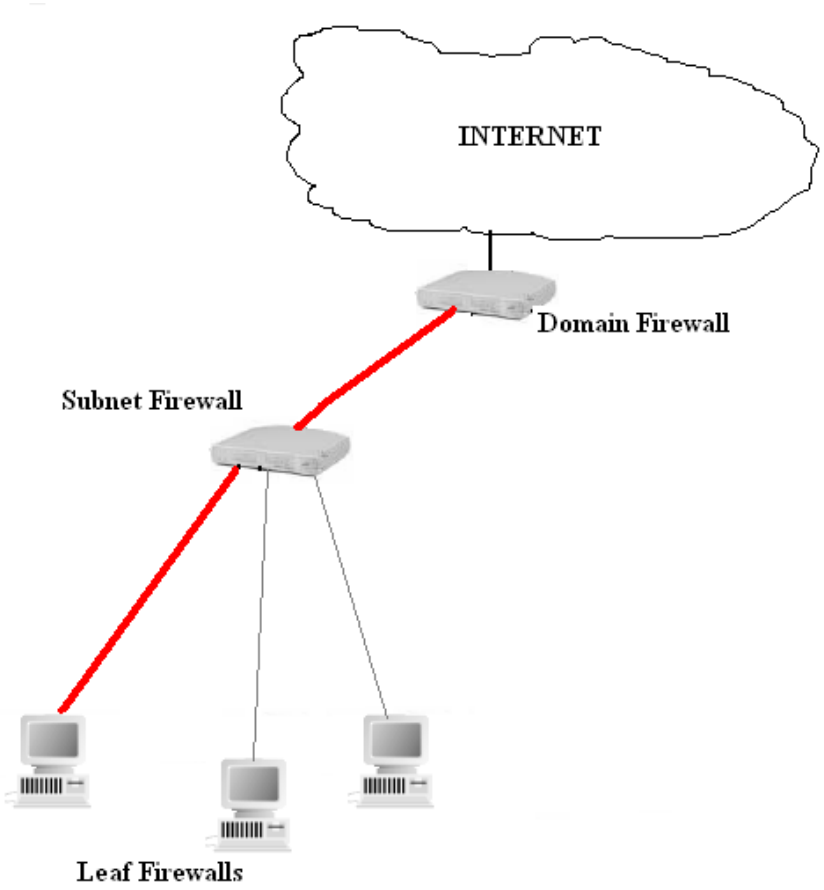


Figure 3.2. Communication on the Internet Path

As expected, each node in the system is responsible for enforcing corresponding security rules that are a subset of overall security policy. In addition to this, each firewall node should also perform some extra jobs to meet our monitoring purposes. First of all, we propose that every node is responsible for storing firewall rules, which are to be applied, in a local database system. Besides, this local database will be responsible for storing the necessary information for the use of monitoring application. And finally, each firewall node has to execute the monitoring application whenever requested.

The details of the monitoring application are explained in Chapter 4. Through the rest of this chapter, the focus is on database design and the data stored in the database.

3.2. The Database Design

As mentioned above, each firewall node has to maintain a local rulebase to store rules that will be used in the monitoring application. Basically, the monitoring application is 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. In order to meet this requirement, the following tables will be stored in each local database of every single node.

- ActiveRules: This table maintains the policy rules that are actively applied by the firewall at a certain time.
- DeletedRules: This table contains rules that are removed from ActiveRules table and is not valid any more.
- UpdatedRules: This table is to keep track of updated rules.
- CRUD: This table maintains the CRUD (Create, Read, Update, Delete) operation and relevant information like who made the action, when, etc.

The detailed explanation of these tables is given below.

3.2.1. Active Rules Table

Basically, this table contains the necessary information to define a firewall policy rule that is actively applied by the firewall node. The database table contains the following fields, which are used to describe a firewall policy rule.

- Rule ID: A unique rule id to identify each 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.
- 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.
- Ingoing/Outgoing: This field specifies whether network packet belongs to an incoming or outgoing traffic.
- Ethernet Card ID: The identity (ID) of the physical interface though which the network traffic passes.

3.2.2. Deleted Rules Table

This table is to store the policy rules that are removed from the security policy rule set. The structure of the table is same as the Active Rules table.

3.2.3. Updated Rules Table

It may become necessary to update policy times in time as a result of changing requirements, etc. Monitoring application needs to keep track of updated rules and their original versions. This table helps to implement this feature by using the information stored in the following fields.

- 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. The former rule is removed from the Active Rules table and inserted into Deleted Rules table.

3.2.4. CRUD Table

This is the main table that the monitoring application is mainly interested in. 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:

- User Name: Every user that has the right to edit the policy rules should have a unique user name. This will allow the monitoring application to query who made the specified operation.
- Domain: The domain ID inside which the specified operation is performed. As mentioned above, each node in the system belongs to a domain.
- Operation: The type of the operation (Create, Read, Update, Delete) performed on the rule set.
- Date: Time of the operation.
- Rule ID: The ID of the rule on which the operation is performed.

3.2.5. Query Table

This table is used to store data that is necessary to keep the history of queries that are performed by the users in the system. The fields in the table are as follows:

- User Name: User that has performed the query.
- Query Type: Type of the query that has been performed.
- Date: Date when the query is performed.

The ER diagram in Figure 3.3 explains the relationships between the database elements. CRUD table contains the information about the active, deleted and updated rules. As a result of an update operation, the existing rule is deleted and a new rule containing the updated information is added to the system. Therefore, Updated Rules table refers to the deleted rule and active rule that are the results of an update operation.

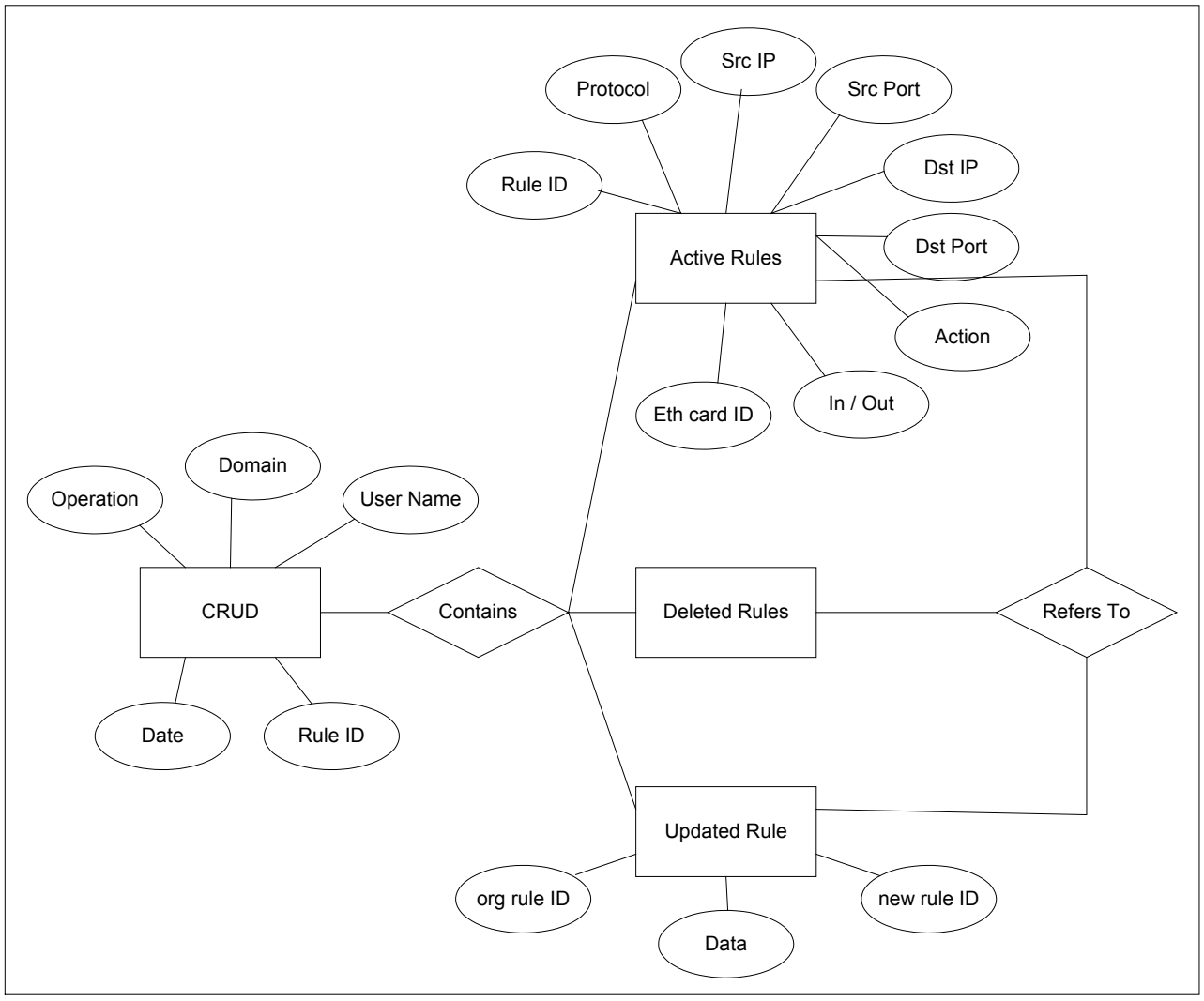


Figure 3.3. ER diagram for the local database managed in each firewall node.

3.3. Scenarios Related to Database Tables

3.3.1. Create a New Rule

Assume that a new rule called R will be created. Once the necessary data to define the rule is entered, R will be inserted into Active Rules table. After insertion to Active Rules table is completed successfully, the CRUD table will be updated with the corresponding data. Figure 3.4 indicates the sequence of operations performed on the corresponding database tables.

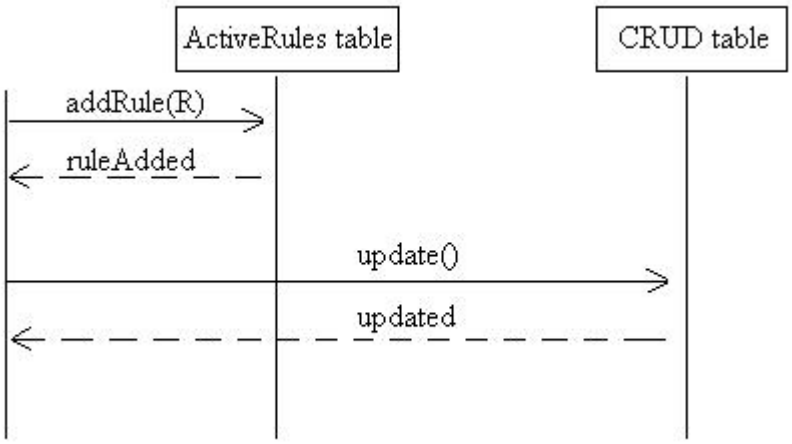


Figure 3.4. Tables that are used when creating a new rule.

3.3.2. Read a Rule

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

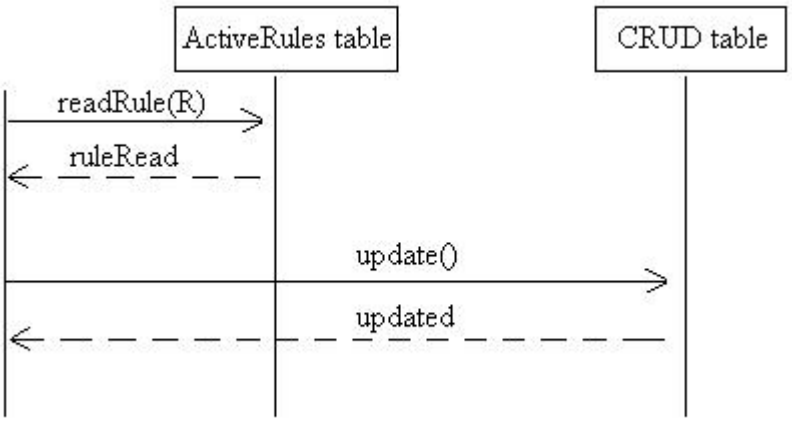


Figure 3.5. Tables that are used when reading a rule.

3.3.3. Delete a Rule

While deleting a rule R from the system, first rule R is removed from the Active Rules table. Then, R is inserted into DeletedRules table so that it is possible to retrieve old

data whenever required in the future. Finally, the CRUD table is updated accordingly. Figure 3.6 summarizes the sequence of operations that are performed when deleting a rule.

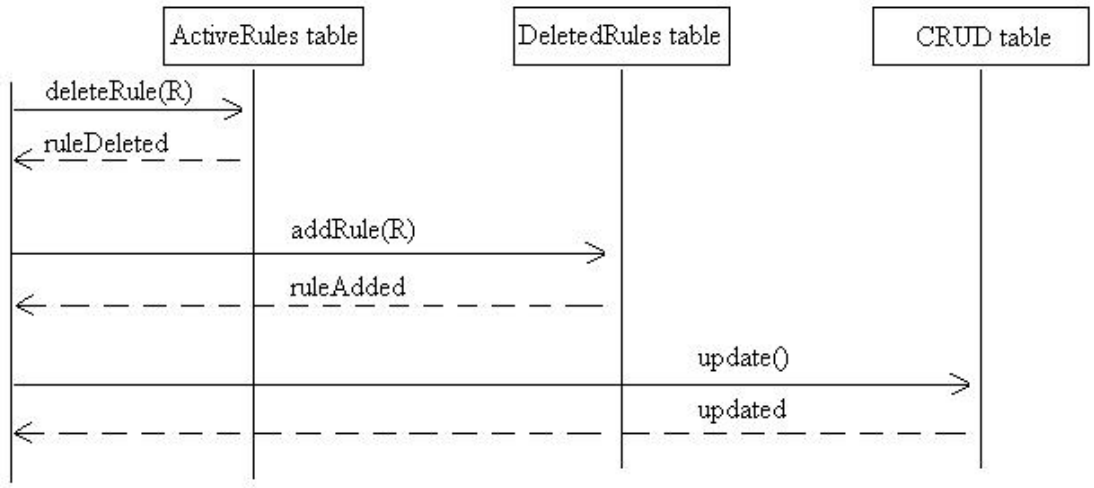


Figure 3.6. Tables that are used when deleting a rule.

3.3.4. Update a Rule

As stated in Figure 3.7, assume that the rule R1 is updated into the new rule R2. First, R1 is deleted from ActiveRules table and the deleted rule data is inserted into DeletedRules table. Then, the new rule R2 is added to ActiveRules table. At the next step, UpdatedRules table is updated with the corresponding data to keep track of the updated rules for later use. Then the final step is to update CRUD table.

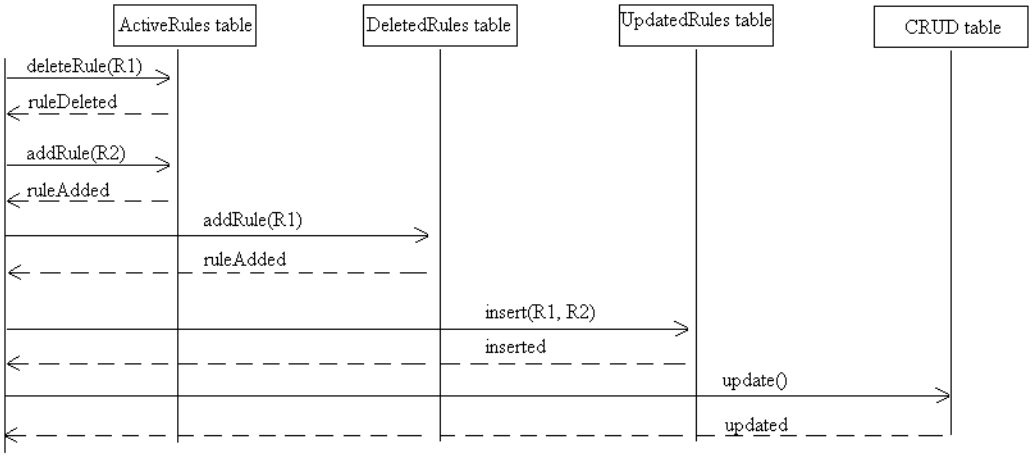


Figure 3.7. Tables that are used when updating a rule

CHAPTER 4

THE MONITORING APPLICATION

As a result of increasing threat of network attacks, network security has become a very hot topic for both industrial and research activities. As stated at the beginning, one of the important elements of network security is firewall. There are several practical and theoretical studies on this area of network security. Firewalls play a critical role for a secure network by controlling / filtering incoming and outgoing traffic according to a predefined set of policy rules. The definition and management of policy rules is very crucial because any inconsistency or insufficiency may dramatically decrease the efficiency and effectiveness of the firewall performance (El-Shaer and Hamed 2003). The conflicts between firewall rules or incorrect ordering of firewall rules may produce incorrect actions performed on network traffic. These conflicts/inconsistencies in filtering rules are named as policy anomalies (El-Shaer and Hamed 2004).

Policy anomalies can be divided into two categories according to the type of firewall architecture. For conventional firewalls it is possible to talk about intra-firewall anomalies, which can basically emerge when a packet may match more than one filtering rule. For distributed firewalls, on which the policy rules are distributed and applied at several firewall nodes instead of a single one, the problem becomes more complicated. Besides the intra-firewall anomalies, it is also possible to have inter-firewall anomalies. Basically, inter-firewall anomalies occur when individual firewalls in the same path perform different filtering actions on the same traffic (El-Shaer and Hamed 2004).

To have a fully and correctly functioning firewall system (either conventional or distributed), the elimination of anomalies from the filtering rule set and resolving any other inconsistencies becomes very important issue. In order to meet this demand, it would be very helpful to have tools/applications that can be used to detect or analyze such kind of problems in a firewall system to have a more secure network. In this chapter, a firewall monitoring application working on a distributed firewall environment is presented.

4.1. Functionality Offered by Monitoring Application

The proposed application is a monitoring system, which works on a distributed firewall environment. The details of the distributed firewall architecture on which the application will run are explained in Chapter 3. 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.
- Reading a policy rule from the existing rule set.
- Updating an existing rule.
- Deleting a rule from the existing rule set.

Basically, the monitoring system is responsible for displaying the detailed information about these major operations that are performed on the filtering rules. For short, these operations will be called as CRUD in this document. The content of displayed data includes enough information that reflects the details of performed operation. For example, it will be possible to answer easily to the following questions by using the presented data: *"When a certain rule is created / read / updated / deleted?"*, *"Who has created / read / updated / deleted a certain rule?"*, *"What are the currently active rules that are applied?"*, "*What is the list of deleted rules from the system?*", and so on.

Note that the application does not intend to make a detailed analysis on the queried data, such as to discover the anomalies or inconsistencies within the overall rule set, etc. So it is important to emphasize that the application is not an anomaly detection or similar kind of tool. It's just a monitoring tool to present data that falls into the area of interest. However, it may play a critical to design and implement more sophisticated tools like mentioned above.

4.2. Working Principles of Monitoring Application

As mentioned while describing the distributed firewall architecture for which the application is designed, each node in the system has to maintain a local database to store the data containing the CRUD actions performed on policy rules. These actions can be performed by different users like network administrator, another application that manages the firewall policy rules automatically. Once a node detects that one of the CRUD actions are performed in its internal rule set, the required data to represent this action is stored in the nodes local database. The stored data contains enough information to represent the corresponding operation and to answer the questions mentioned in the previous section. The detailed information about the stored data and database design can be found in Chapter 3.

The application is composed of two main parts as indicated by Figure 4.1: Monitoring Agent and Database Agent. It is the responsibility of the monitoring agent to manage the data query request and presentation. To be able to perform this, it is necessary to run a sequence of database queries on the overall database system. The monitoring agent is responsible for retrieving incoming request from external world (i.e. from the user). Once the user request is retrieved, it is converted to a query request by monitoring agents and sent to the database agent of the same node. The database agents are responsible for handling incoming query requests to them. One of the main functionality of the database agent is to perform the incoming database query request on the local database and to send the resulting data to the monitoring agent. Note that the application is working on a distributed environment and the required data is stored in distributed manner, so the data gathered from the local database of a node is not enough to reflect the whole. Thus, it becomes quite important to run the database queries correctly on the overall architecture to cover all of the required data in the system.

Besides, it may also be important to hide the distributed characteristics of the system from the application. The database agents have an important role while hiding the distributed characteristics of the database system from the application. To perform this, database agents have another responsibility, which is to pass the incoming query requests to the other nodes in the system whenever needed. This feature of the database agent lets the

application not to know anything about the distributed database architecture lying in the background. The monitoring agent will only need to deal with its local database agent. It will just send the query request to the local database agent and get resulting data from it. In the background, active database agents will communicate to each other to extract the necessary data, but all of these operations are hidden to monitoring agent. Querying only the local database or the whole database is not different from the each other from the application's point of view.

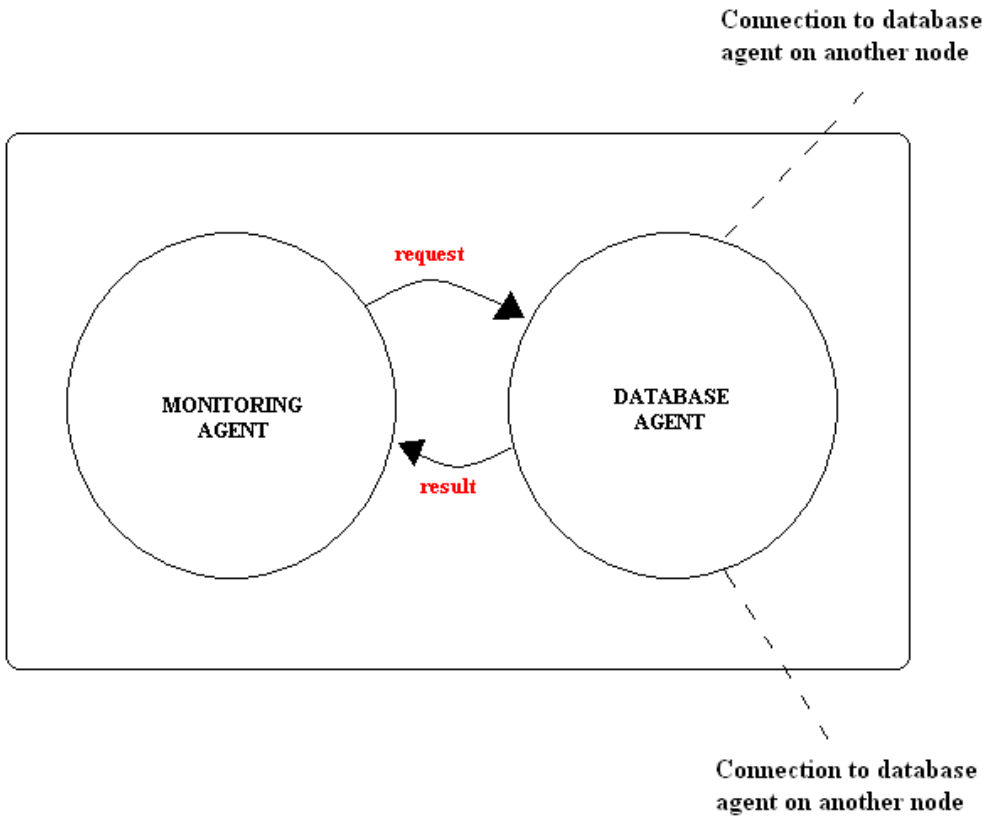


Figure 4.1. The parts of monitoring application running on a single firewall node.

4.2.1. Design and Implementation Details of Monitoring Application

As mentioned, monitoring agent and database agent are two main components of the application. The main responsibility of the monitoring agent is to get request from a

user and to post it the database agent. Additionally, monitoring agent is responsible for the presentation of resulting data to the user. To be able to perform those tasks, two different versions monitoring agents is implemented. The first version is a command line interface that reads user inputs from command line and display the resulting data to the command line. The second version is a graphical user interface that is implemented using the graphical widgets provided by the Java. In either way, the underlying communication between monitoring agent and database agent is the same.

Database agent is responsible for handling the incoming query requests and preparing the resulting data.

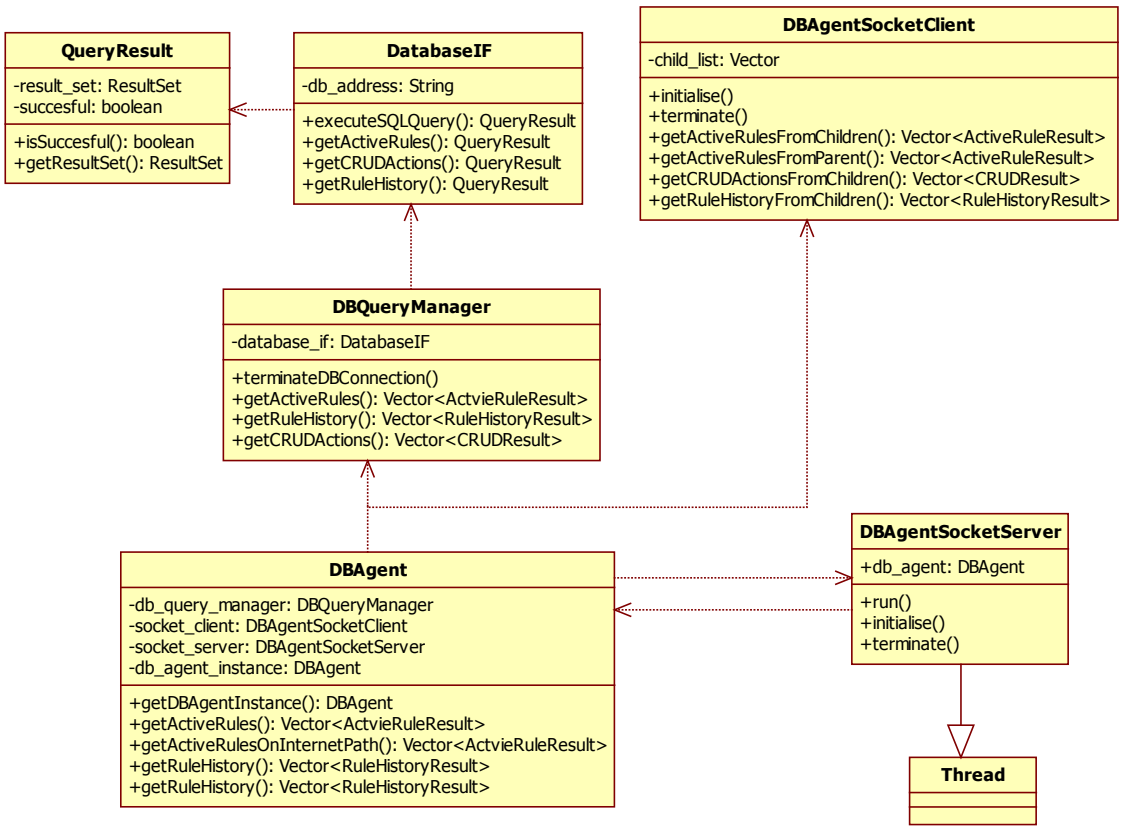


Figure 4.2. Class Diagram for Database Agent.

Figure 4.2 indicates the details of database agent. As it can be seen from the figure, database agent contains a couple of classes. The responsibilities of each class is as follows:

DatabaseIF: DatabaseIF is responsible for the management of primitive database operations. This class basically performs simple database queries and returns the raw data containing the query result.

DBQueryManager: DBQueryManager uses a DatabaseIF instance to perform database queries and prepares resulting data in a more compact format.

DBAgentSocketClient: As stated earlier, a database agent is responsible for forwarding query request to the database agents of other nodes whenever required and collecting the resulting data from them. The DBAgentSocketClient class performs these operations. DBAgentSocketClient maintains an internal child list to which it can communicate. When an incoming request comes, it connects to children in the child list, forwards the request to them and gets the resulting data from them.

DBAgentSocketServer: When request received from the DBAgentSocketClient of another node, it is the responsibility of the DBAgentSocketServer to answer that request.

DBAgent: It is the responsibility of the DBAgent to manage all of the classes mentioned above. Basically, monitoring agent just knows about DBAgent, it does not need to know about the underlying structure. DBAgent provides the necessary functionality to handle necessary requests and to gather required data.

Figure 4.3 explains how these classes interact with each other during the processing of a "retrieve active rules" request for an architecture having two nodes (Node1 and Node2). Monitoring Agent of Node1 makes a request to the database agent of the Node 1. Database agent of Node1 first queries its local database and then forwards the request to Node2. Node 2 gets the resulting data from its local database and sends back to Node1. DBAgent of Node1 collects the whole data and sends it the monitoring agent.

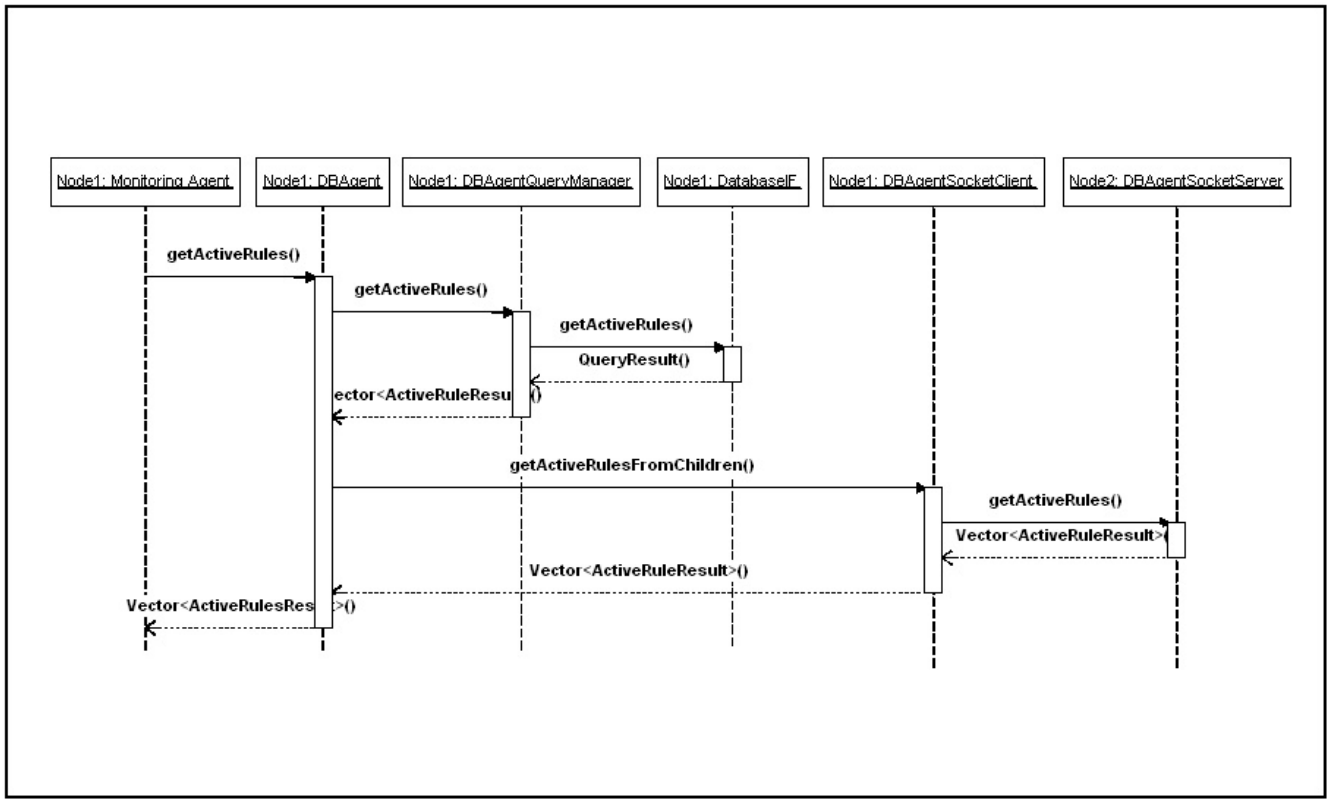


Figure 4.3. Sequence diagram to retrieve active rules.

The implementation of the application is done by using Java programming language and facilities on Windows operating system. The communication between different nodes in the system is based on the Java sockets. Besides, as a database engine, JavaDB is used. Java DB is a SQL based lightweight database engine that is provided by the Sun Microsystems as a free utility. It is possible to configure JavaDB in either embedded mode or client/server mode. In our case, each node needs to maintain its own local database, so we preferred to use JavaDB in embedded mode.

4.2.2. Communication Between Firewall Nodes

The flow of communication between the firewall nodes during a database query request depends on the node that has triggered the query. Besides, it may be requested to query just the local database rather than the whole distributed database in the corresponding domain. The flow of communication in each of these cases is explained below.

Case 1: The monitoring application is running on a leaf firewall

As mentioned the application may want to query the information stored in the local database of the node on which the application is running. This is the case when a leaf firewall has executed the application. The monitoring agent of the leaf firewall will send the query request to the local database agent and database agent will return the data to the monitoring agent after extracting it from local database. Such a query does not require an inter-nodes communication so the database agent does not need to communicate with any other nodes. Although there may be some cases in which this type of monitoring is useful, it provides data just for a subset of the overall rules.

Case 2: The monitoring application is running on a subnet firewall

The application may also want to monitor the data for whole domain or subnet instead of a single node. As expected, this case is more complicated than the previous one since it contains communication and data sharing between database agents on different nodes. Additionally, it is also needed to spend some extra effort to query the data stored in the distributed database system. However, as stated earlier, these extra tasks resulting from the distributed characteristic of the database system will be hidden from the monitoring agent. Let's look at how this will be handled in more details.

If the node that has started the monitoring application is a subnet firewall, then it can query the whole distributed database inside that subnet.

In such a case, the database agent of the node will send the query request to all other nodes in the corresponding subnet as shown by Figure 4.4.

When each node in the subnet receives the query request, database agent on that node will query the local database and will send the resulting data to the node that has triggered query (subnet firewall). The flow of the communication for this case is indicated in Figure 4.4. Once the operation is completed successfully, the overall data will be collected at the subnet firewall that has triggered the request.

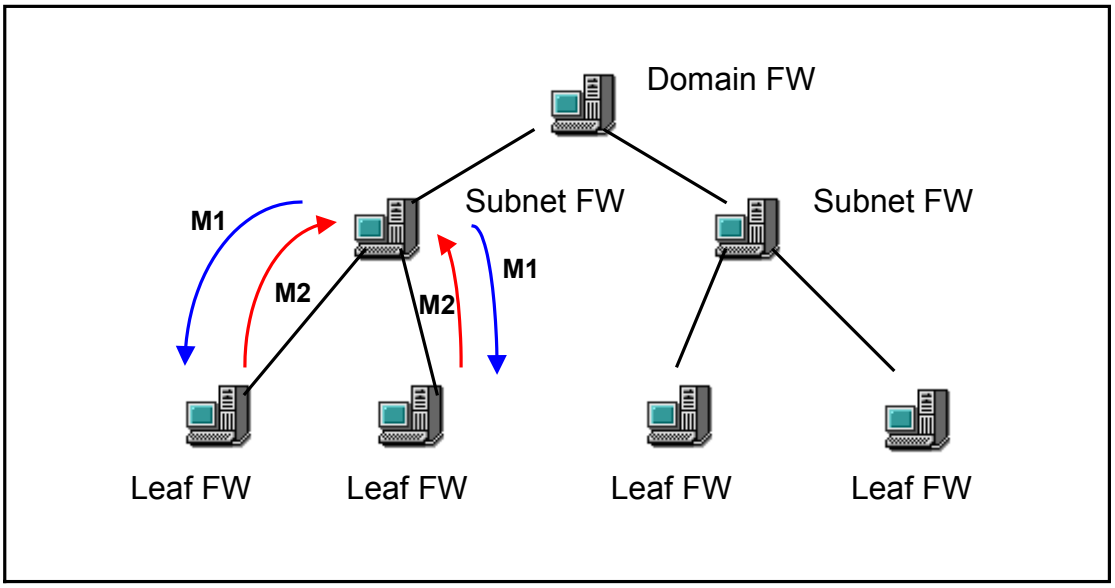


Figure 4.4. Communication between firewall nodes: M1 is request, M2 is result.

Case 3: The monitoring application is running on a domain firewall

If the node that has executed the monitoring application is a domain firewall, then it is possible to query data from the whole distributed database system within that domain. The domain firewall will send the query request to the subnet firewalls inside that domain (Figure 4.5). Then each subnet firewall will forward the request to the leaf firewalls that are connected to them. Once leaf firewalls receive the query request, they will query their local database and gather the required data. Each subnet firewall will wait till the all of the leaf firewalls inside that subnet collects the necessary data.

Once the data is collected at the subnet firewall, it will be forwarded to the domain firewall (Figure 4.5). Domain firewall will present the data to the user when all of the subnet firewalls sends the required data.

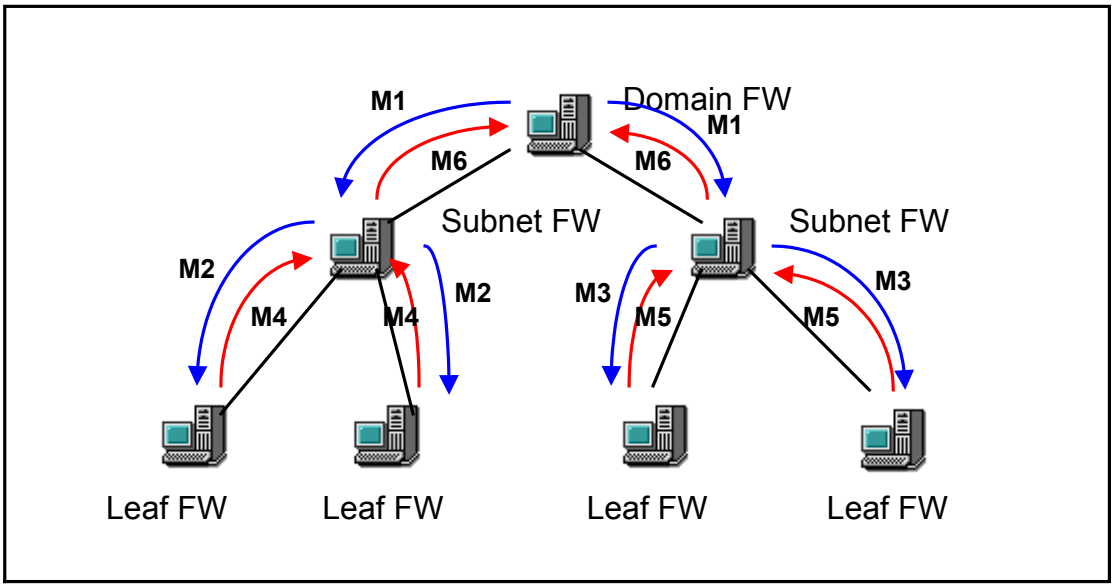


Figure 4.5. Communication between firewall nodes: M1, M2, M3 are request, M4, M5, M6 are result.

Case 4: The monitoring application has requested the policy rules that are active on firewall nodes on the path to the Internet

As stated, the monitoring application cannot make a query request to another node that is in a different domain / subnet or that has located at a higher level of the hierarchy. However, there is an exceptional case for this. In some cases, the monitoring application may need to query data stored in the nodes, which are located in the path that connects a node to the Internet. For example, consider the case when a leaf firewall wants to make a query on the firewall nodes that are on the Internet path. As indicated by Figure 4.6, in this case, the leaf firewall will send the request to the subnet firewall, which is at the next higher level of hierarchy. This process will continue till the request reached to domain firewall. Similarly, the resulting data for the requested query will travel back to the firewall node that has triggered the query.

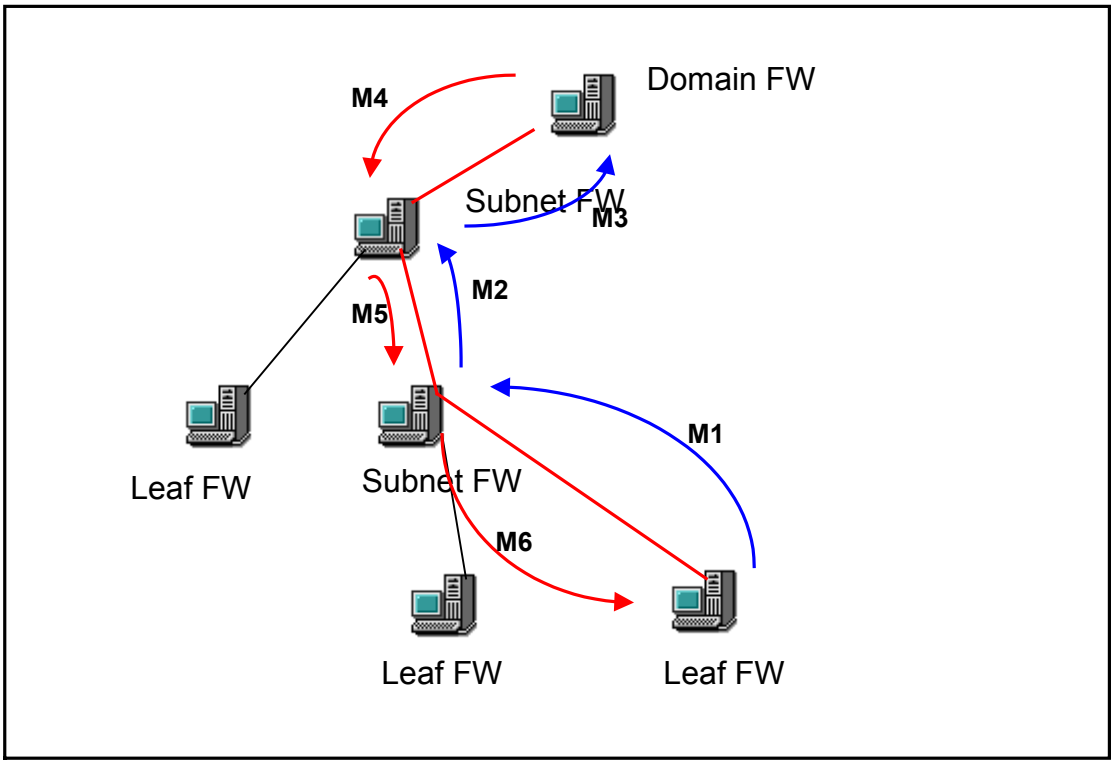


Figure 4.6. Communication on the path to the Internet: M1, M2, M3 are request, M4, M5, M6 are result.

4.3. Possible Scenarios

4.3.1. Listing All of the Active Rules for a Firewall Node / for a Set of Nodes

1) User starts the application.
2) The services that are offered by the application are listed to the user.

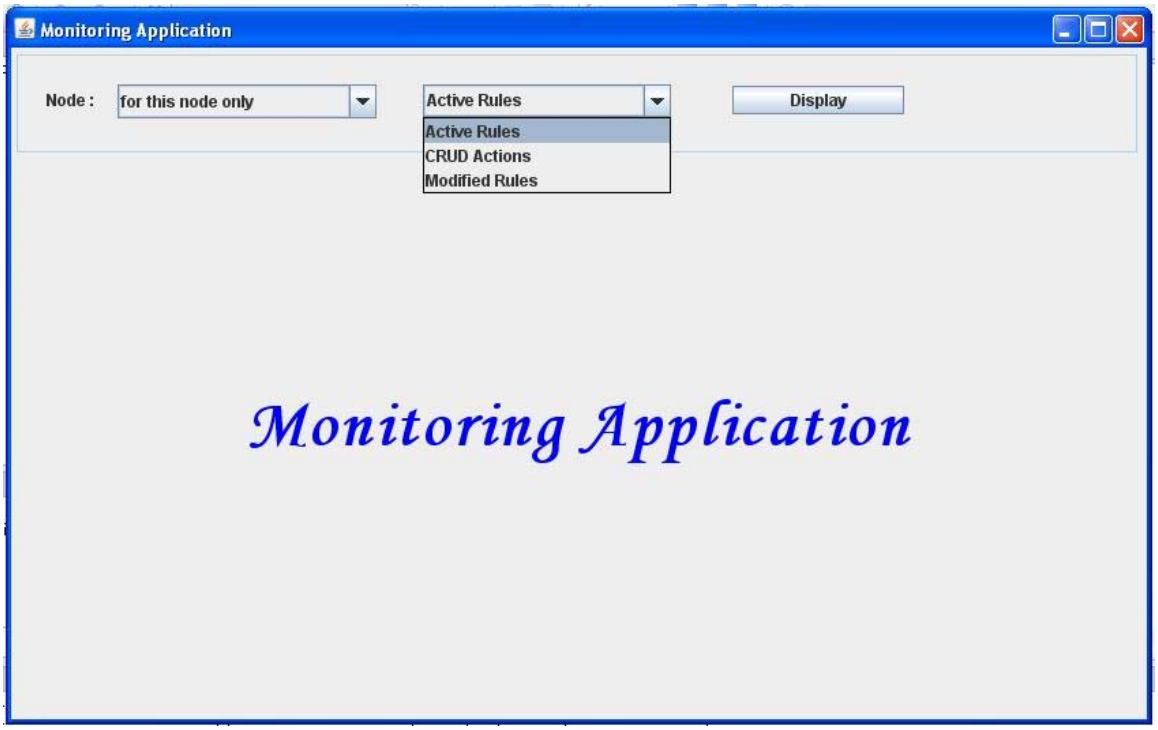


Figure 4.7. Active Rules option.

3) User selects the "active rules" option by using the menu shown in Figure 4.7. Additionally, this menu allows user to specify a single node or a set of nodes (nodes in the lower level of hierarchy), or nodes on the Internet path.
4) Once the user completes the selection of the search parameters, the request should be submitted. Pressing the "Display" button does submission.
5) Once the user submits the request, the Monitoring Agent running on that node sends the request to a Database Agent.
6) If the user selected to view active rules for a single node, then the Database agent executes the query on the local database. The active rules for a node are stored in the Active Rules table, so the query is simply to retrieve the required data from the Active Rules table. Other database tables are not used in this case. If the user requested to see the active rules for a set of nodes, then the Database Agent forwards the query to the Database Agents of the other nodes in that domain. In this case, each Database Agent executes the query and retrieves the data from the Active Rules tables of their local database. Then each Database agent sends the resulting data back to the original node that has started the query.

7) Once the Monitoring Agent gathers the required data either from the local databse or from the specified domain, it forwards the data to the Monitoring Agent. The data is presented to the user. The resulting output window is shown in Figure 4.8.

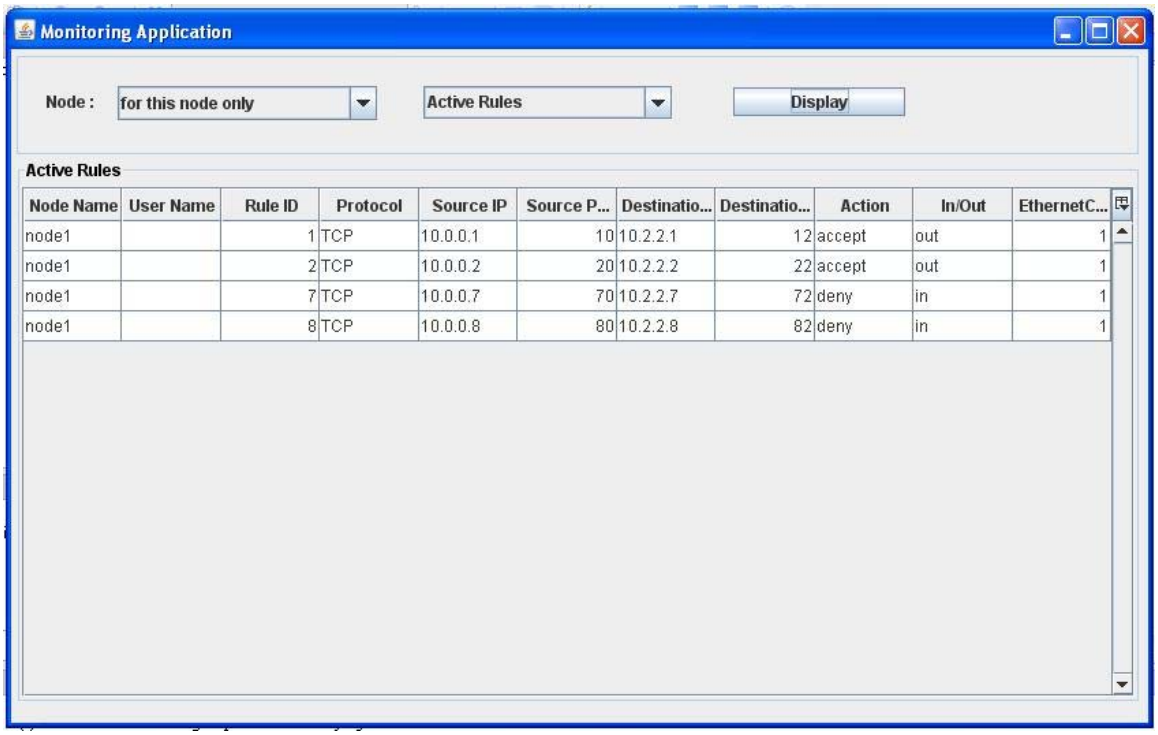


Figure 4.8. Resulting menu for list of active rules.

8) Once the output is listed, it is possible to filter specified data from the output. Once the output window is “right clicked”, the filter window in Figure 4.9 is displayed:

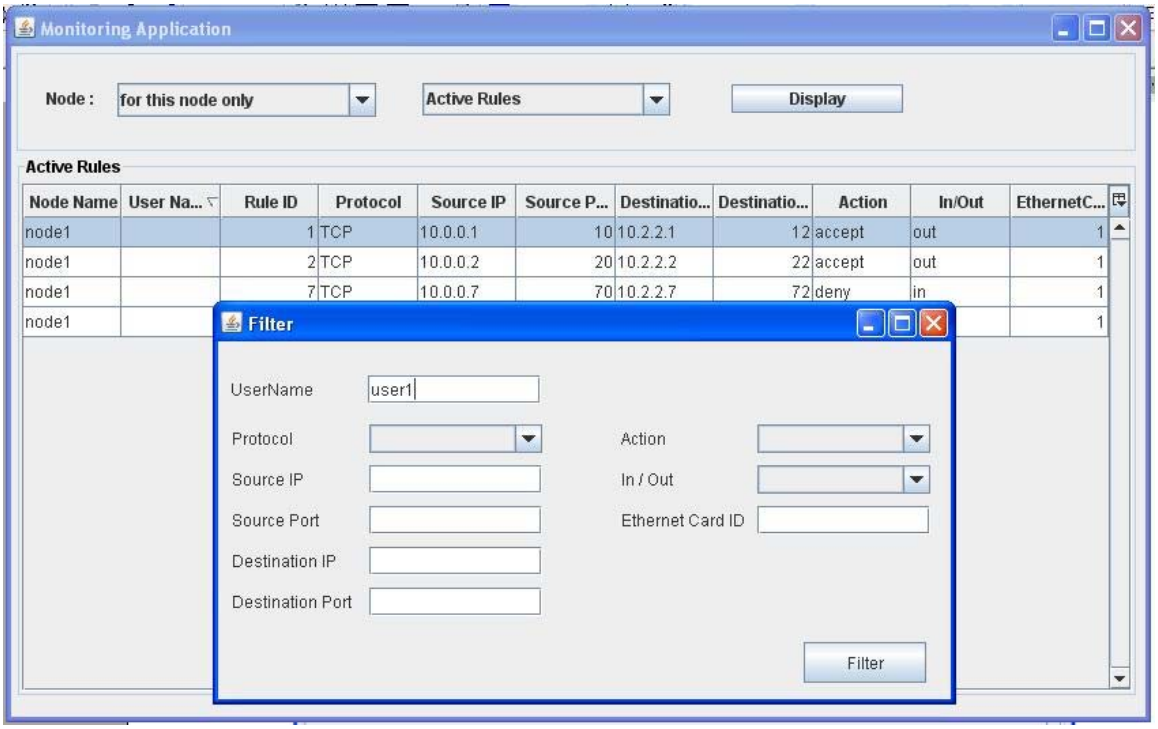


Figure 4.9. Filtering on active rules result.

9) Once the filtering data is filled and submitted, the application will perform the requested filter operation and the output window will be updated.

4.3.2. Listing All of the CRUD Actions [Performed by a Specific User] for a Firewall Node / for a Set of Nodes

1) User starts the application.
2) The services that are offered by the application are listed to the user.

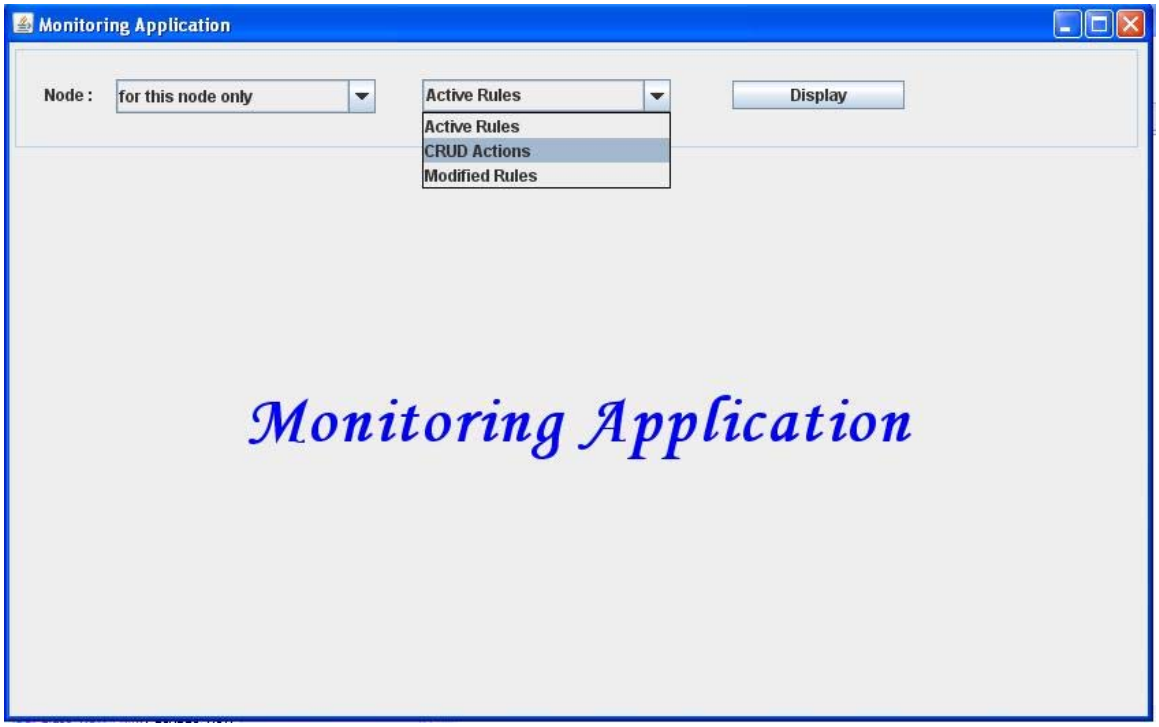


Figure 4.10. CRUD Actions option.

3) User selects "CRUD actions" option as shown in Figure 4.10. Once this option is selected, the search menu is displayed. Additionally this menu allows user to select whether the request is just for a single node or for that node and its children (nodes in the lower level of hierarchy).
4) Once the selections are complete, the request should be submitted to the system.
5) Once the request is submitted, the Monitoring Agent creates the corresponding requests and forwards them to the Database Agent.
6) Database Agent is responsible for gathering data either from the local database or from the databases distributed over the specified domain. After data is ready, the Database Agent will forward it to the Monitoring Agent.
7) Once the Monitoring Agents gets the resulting data for the user request, it will be presented to the user in the format displayed in Figure 4.11.

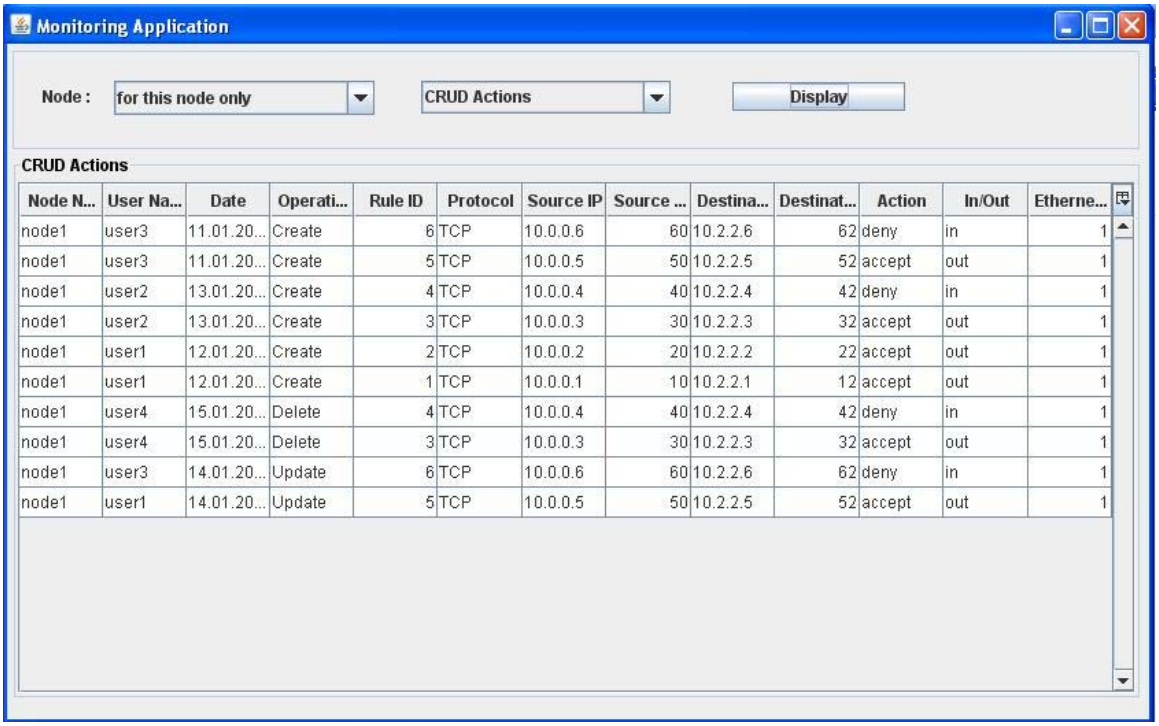


Figure 4.11. Result menu for CRUD actions.

8) The user can perform further filtering on the resulting data as in the previous case.

4.3.3. Listing Modified (Created/Updated/Deleted) Policy Rules [by a Specific User] for a Firewall Node / for a Set of Nodes

1) User starts the application.
2) The services that are offered by the application are listed to the user.

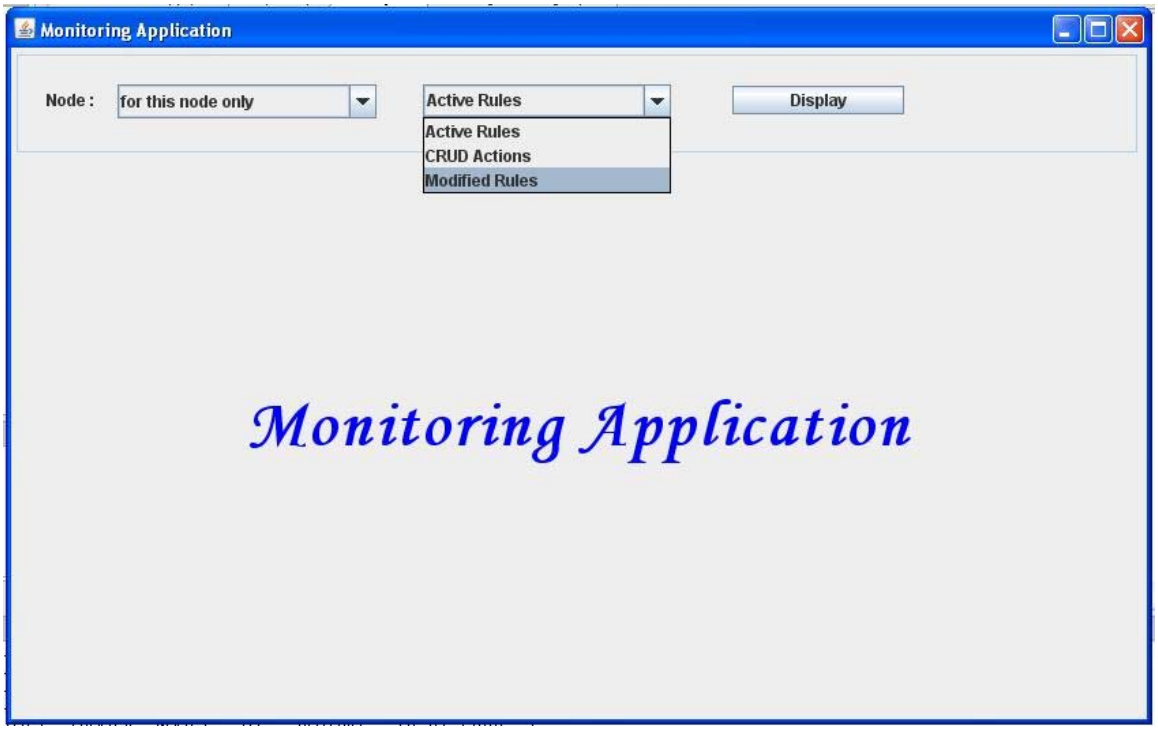
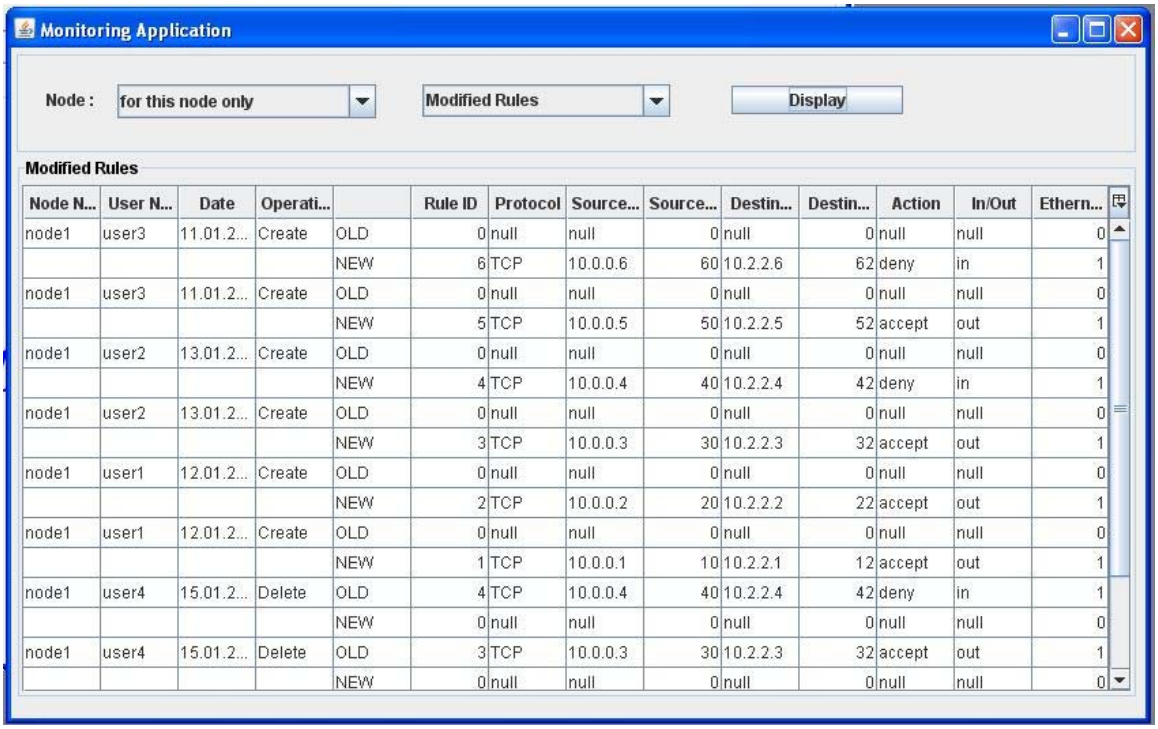


Figure 4.12. Modified Rules option.

3) User selects “modified rules” option as shown in Figure 4.12. Modified means that the content of a rule is changed, so this selection includes the created, updated and deleted rules. Additionally, as in the previous cases, the system allows user to perform node selection.
4) Once the search request is submitted, the Monitoring Agent gets the request and forwards it to the Database Agent.
5) Database Agent queries the local database / or sends the request to the specified nodes to retrieve the resulting data. Once the data is ready, Database Agent forwards it to the Monitoring Agent.
6) Once the resulting data is ready at Monitoring Agent, it will be presented to the user in the format shown in Figure 4.13.



Node N...	User N...	Date	Operati...		Rule ID	Protocol	Source...	Source...	Destin...	Destin...	Action	In/Out	Ethern...
node1	user3	11.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	6	TCP	10.0.0.6	60	10.2.2.6	62	deny	in	1
node1	user3	11.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	5	TCP	10.0.0.5	50	10.2.2.5	52	accept	out	1
node1	user2	13.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	4	TCP	10.0.0.4	40	10.2.2.4	42	deny	in	1
node1	user2	13.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	3	TCP	10.0.0.3	30	10.2.2.3	32	accept	out	1
node1	user1	12.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	2	TCP	10.0.0.2	20	10.2.2.2	22	accept	out	1
node1	user1	12.01.2...	Create	OLD	0	null	null	0	null	0	null	null	0
				NEW	1	TCP	10.0.0.1	10	10.2.2.1	12	accept	out	1
node1	user4	15.01.2...	Delete	OLD	4	TCP	10.0.0.4	40	10.2.2.4	42	deny	in	1
				NEW	0	null	null	0	null	0	null	null	0
node1	user4	15.01.2...	Delete	OLD	3	TCP	10.0.0.3	30	10.2.2.3	32	accept	out	1
				NEW	0	null	null	0	null	0	null	null	0

Figure 4.13. Result menu for modified rules.

7) The user can perform further filtering on the resulting data as in the previous case, which is indicated in Figure 4.14.

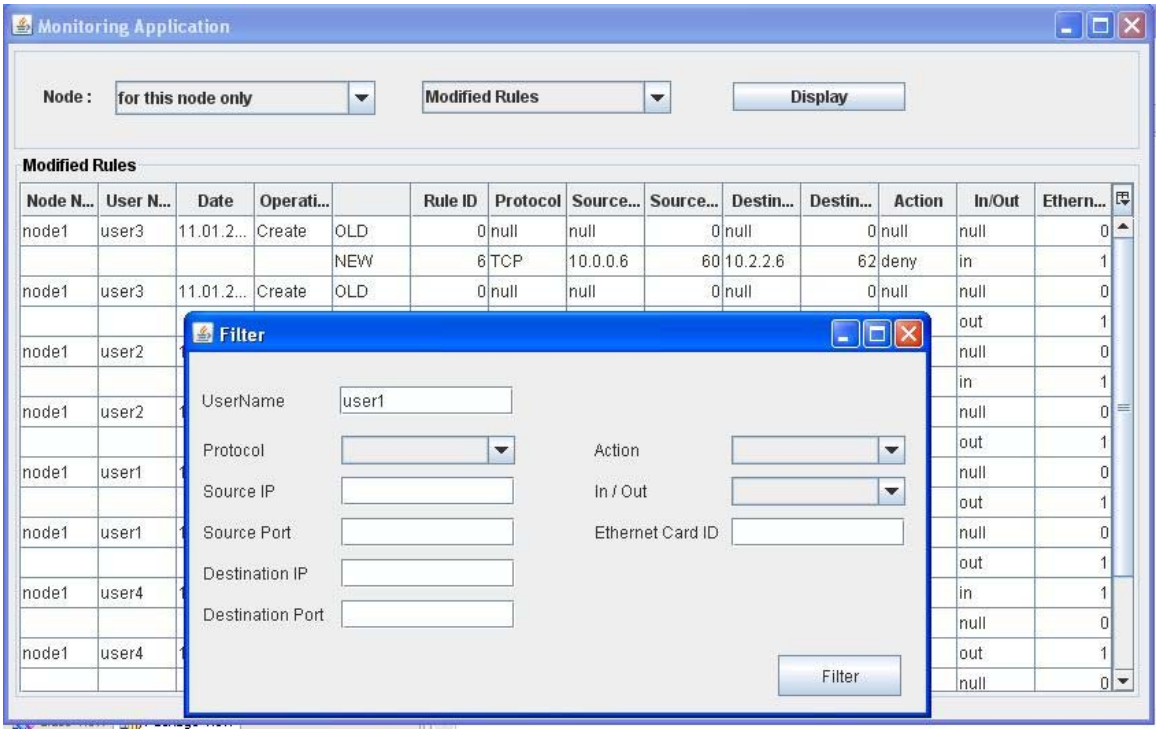


Figure 4.14. Filtering on modified rules output.

4.3.4. Listing History of a Rule [by a Specific User] for a Firewall Node / for a Set of Nodes

1) User starts the application.
2) The services that are offered by the application are listed to the user.
3) User selects "Active Rules" option and active rules are listed to the user.

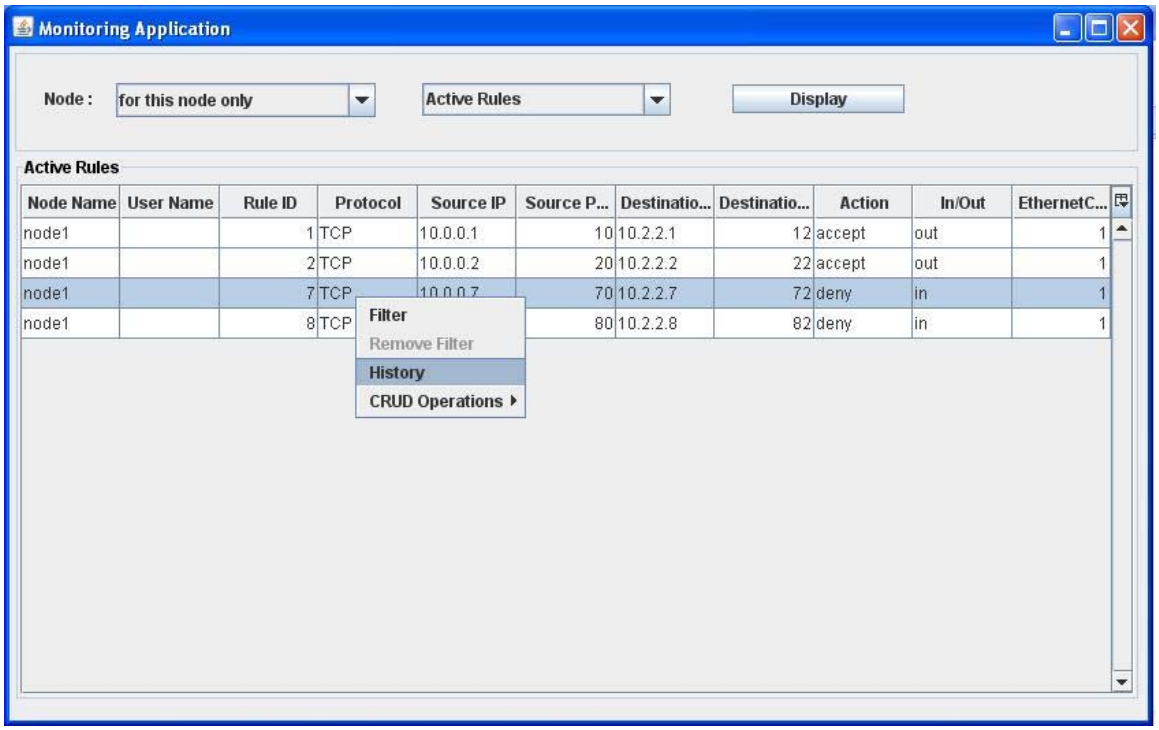


Figure 4.15. Rule history option.

4) When right-clicked on a selected rule as shown by Figure 4.15, "History" option will appear.
5) Once the request is submitted, the Monitoring Agent will get the request and forward it to the Database Agent. Database agent will get the required result from the firewall nodes.
6) Once the Database Agent gathers the data, it will be forwarded to Monitoring Agent and will be displayed to the user as shown by Figure 4.16.

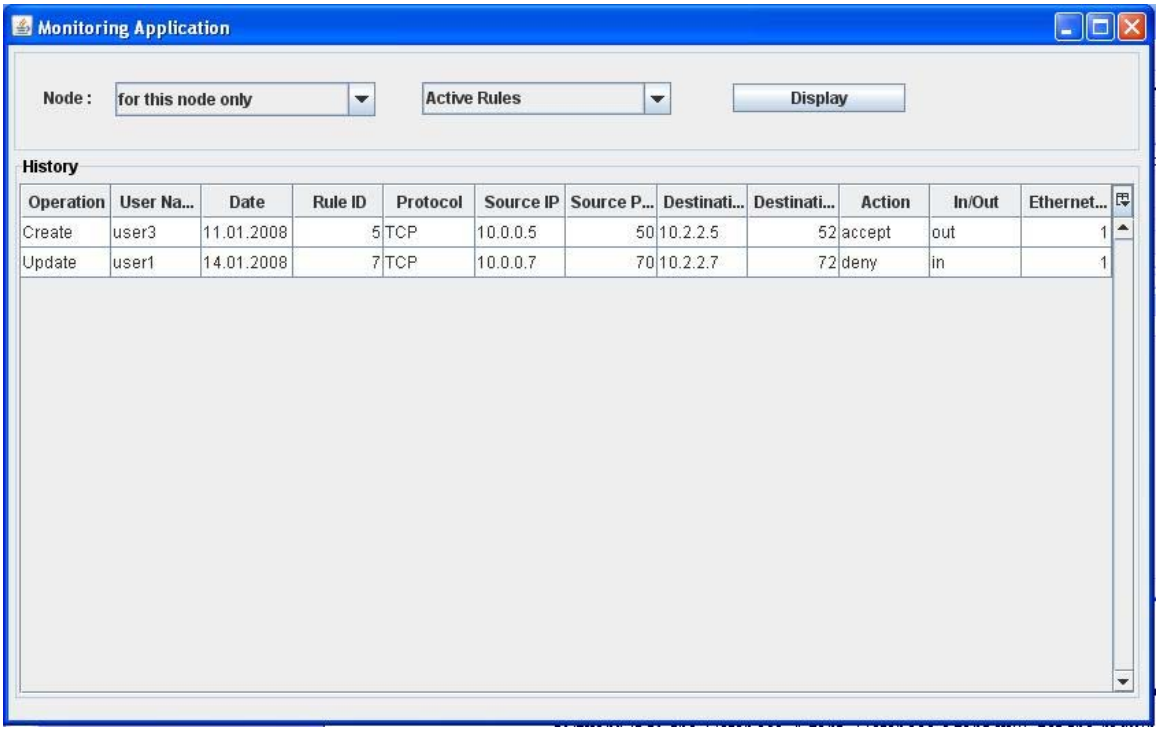


Figure 4.16. Rule History result.

4.3.5. Listing all of the Queries Performed [by a Specific User]

1) User starts the application.
2) The services that are offered by the application are listed to the user.
3) User selects "list of queries" option. Then, the corresponding search menu is displayed. This menu allows specifying a user and a domain as a search option if desired.
4) Once the search is submitted, Monitoring Agent interprets the request and sends a query request to the Database Agent. Database Agent performs the query on the local database of the specified node or on the database that is distributed over the specified domain.
5) Once the Database Agent gathers the required data, it forwards the data to the Monitoring Agent. Then the Monitoring Agent will present the data in an appropriate format.

4.3.6. Adding New Policy Rules to a Firewall Node

1) User enters the application.
2) Open the CRUD operations menu as shown in Figure 4.17.

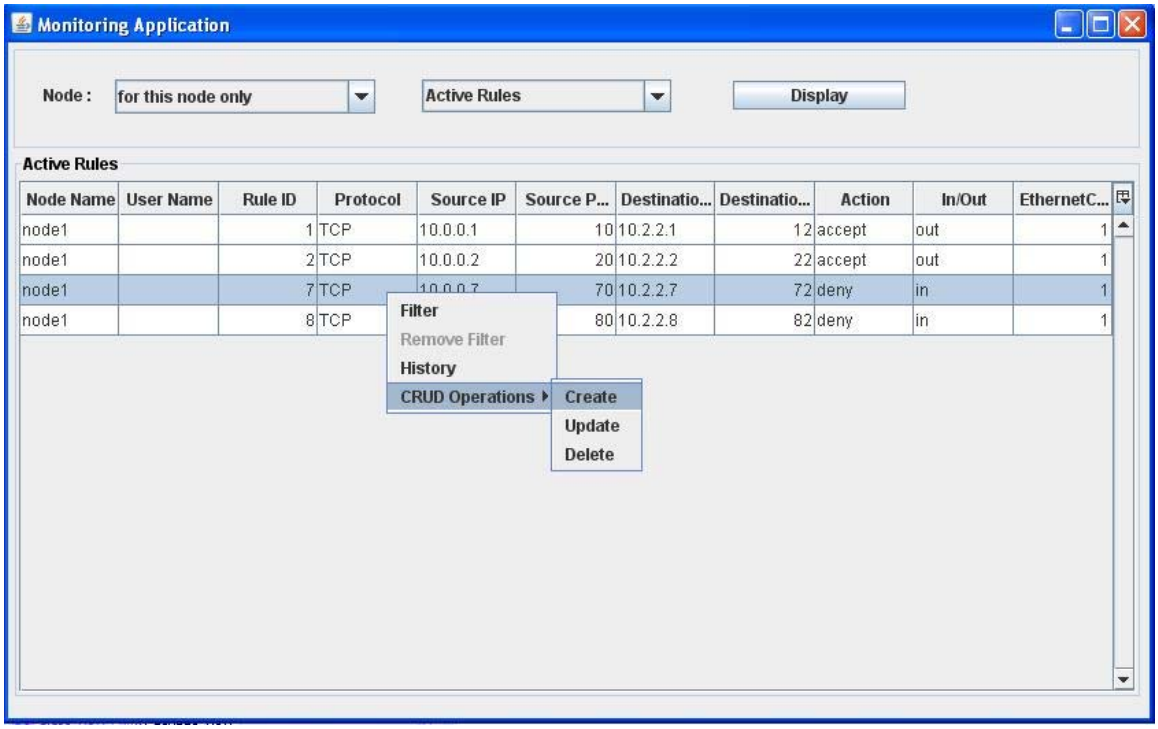


Figure 4.17. Create a new rule option.

3) The window to create a new rule will be displayed (Figure 4.19). Once the required fields to define a policy rule are filled, it should be submitted to the system.

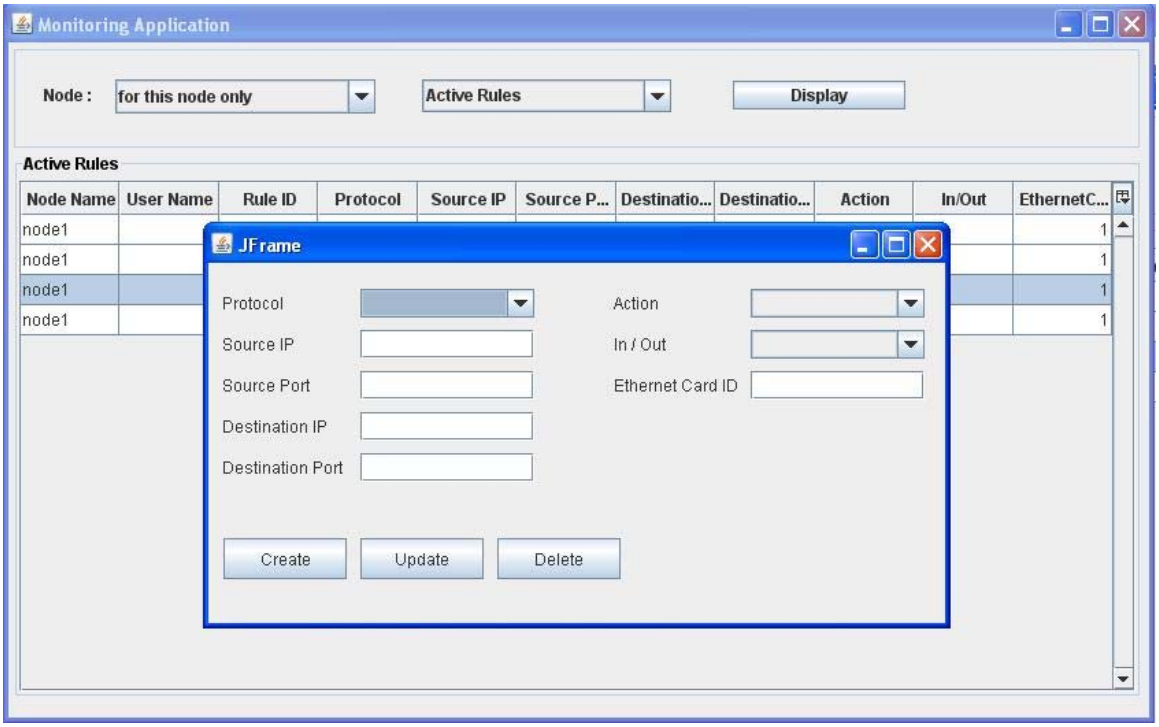


Figure 4.18. New rule definition menu.

4) Once the submission is successful, the rule will be added to the internal rule set.

4.3.7. Updating / Deleting an Existing Policy Rule.

1) User starts the application.
2) Select a rule to update / delete from the active rules list.

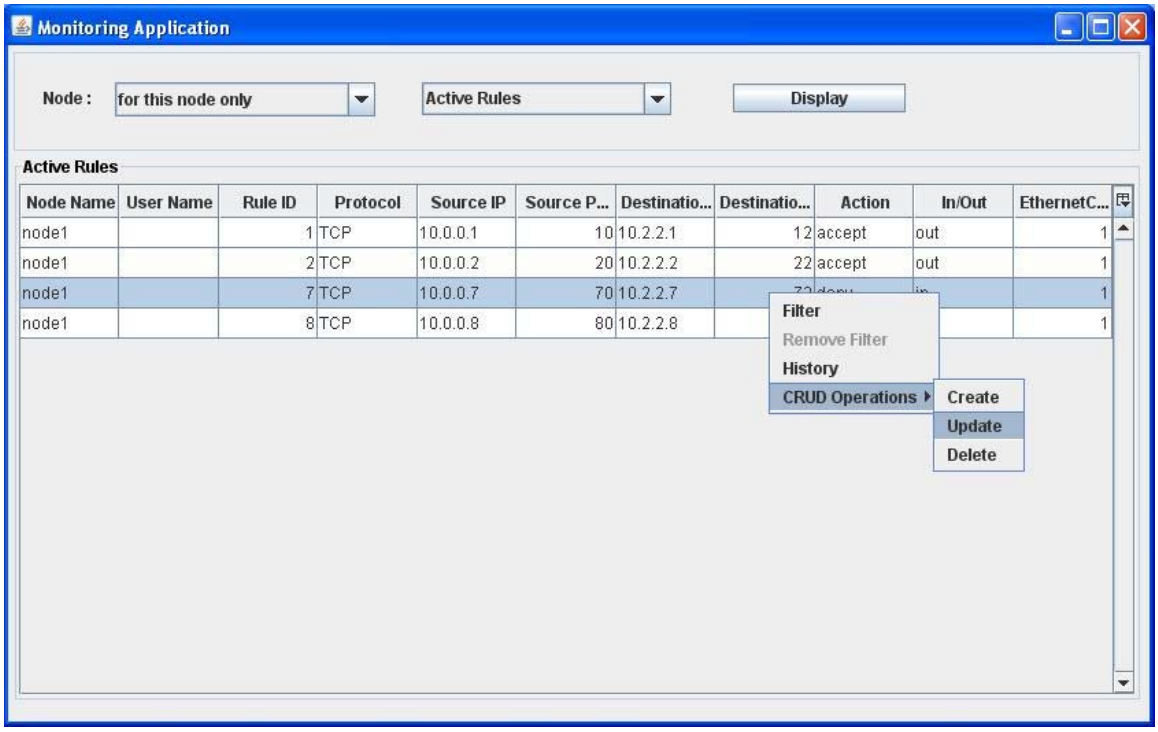


Figure 4.19. Update / Delete option.

3) The content of the selected rule will be displayed to the user. Once the specified fields are updated and submission is completed, the rule will be updated / deleted.

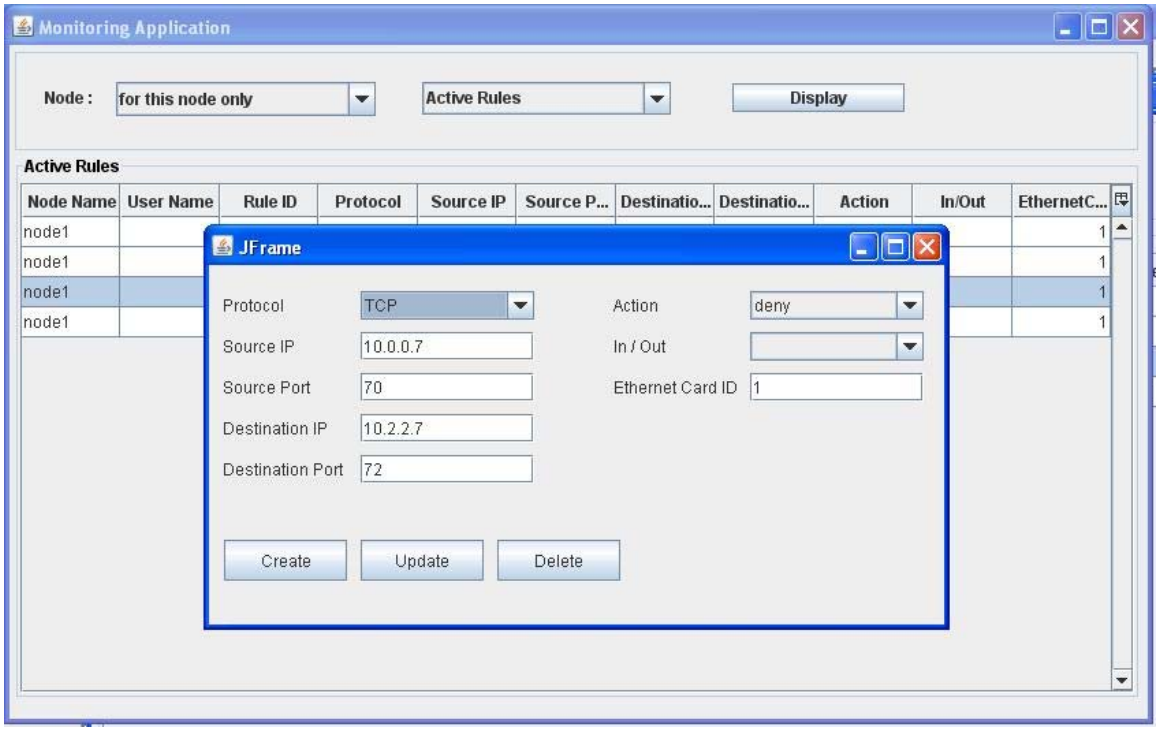


Figure 4.20. Update / Delete menu.

CHAPTER 5

EXPERIMENTS AND EVALUATION

5.1. Experimental Setups and Obtained Results

Different experimental topologies that fulfill the requirements of the proposed architecture are constructed to test and verify the monitoring application. Basically, the experiment is to retrieve active rules running on each node and calculate the amount of time to complete this operation. These experimental topologies and obtained results are as follows:

5.1.1. Single Node

The first experimental setup is just a single node. The monitoring application has been exercised on the rule set of this single node. Five different versions of rule set that contains 20, 20, 60, 80 and 100 rules are used in the experiment.

Since the application is running on a single node using the local database of that node, there is no inter-node communication. The average time values for the corresponding number of rules in the rule set is observed after 100 trials. The extraction of the current active rules varies between 99.76 milliseconds (when there are 20 active rules in the rule set) and 111.24 milliseconds (when there are 100 rules in the rule set). The graph in Figure 5.1 indicates the obtained results in a more compact way.

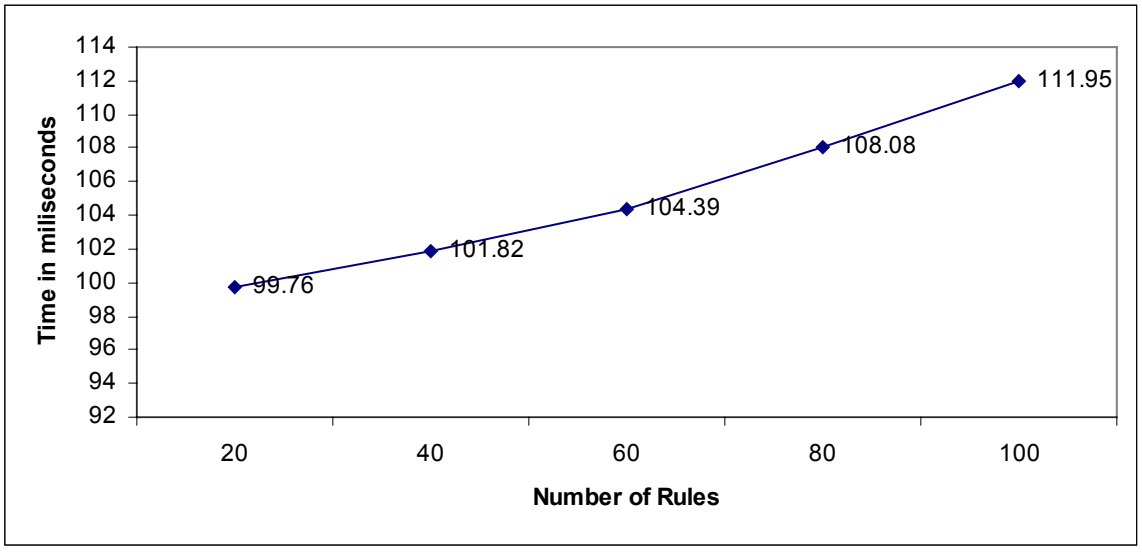


Figure 5.1. Execution times of the retrieving active rules on a single node.

5.1.2. Experimental Setups For Emulation

The experimental topologies are emulated on a PC that has Windows XP operating system running on it. The PC used for emulation has 512 MB of RAM and Intel Pentium Core Duo T2060 processor running at 1.60 GHz.

5.1.2.1. Experimental Setup I

This experimental setup (Figure 5.2) is composed of 7 nodes and 2 levels of hierarchy. Each node in the each level of hierarchy has two children except the ones in the leaf level. The leaf nodes do not have any children. Each node in the tree maintains a local database to store the data that are mentioned in the previous chapters. Additionally, each node is responsible for executing the monitoring application. Node 1, which is at the highest level of the hierarchy, is chosen as the one on which the monitoring request has been performed.

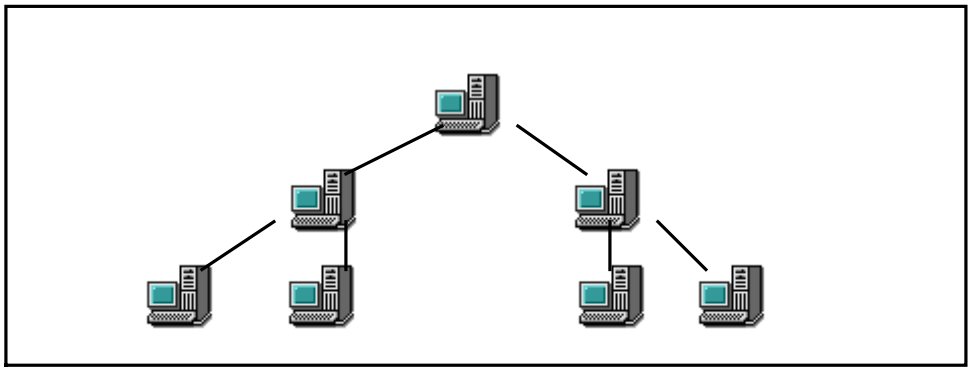


Figure 5.2. Experimental setup I.

Basically, the request is to get all of the active rules from all of the nodes in the topology. The experiment has been performed several times with different number of rules (20, 40, 60, 80, and 100 rules) in the rule set of each node. The average time values to complete the requested query are gathered after 15 trials. The results vary between 963.1 milliseconds (for 20 rules in each node's rule set), and 1268.6 milliseconds (for 100 rules in each node's rule set). As different from the single node case, inter-node communication to share requests and resulting data is performed for this case. The overall results are indicated by the graph in Figure 5.3.

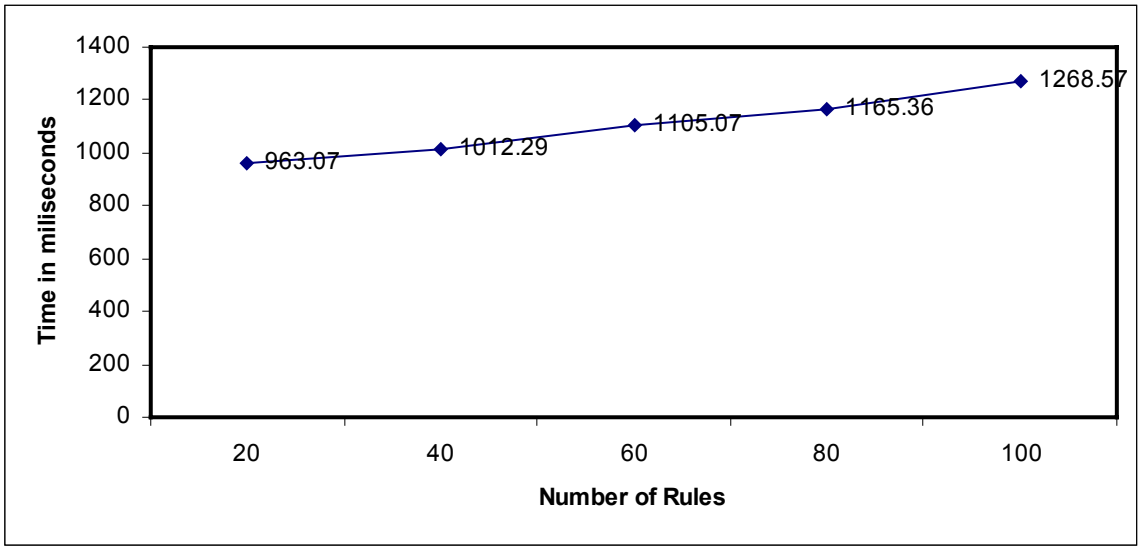


Figure 5.3. Execution times of the retrieving active rules on experimental setup I.

5.1.2.2. Experimental Setup II

The second experimental architecture (Figure 5.4) is composed of 10 nodes and 2 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture.

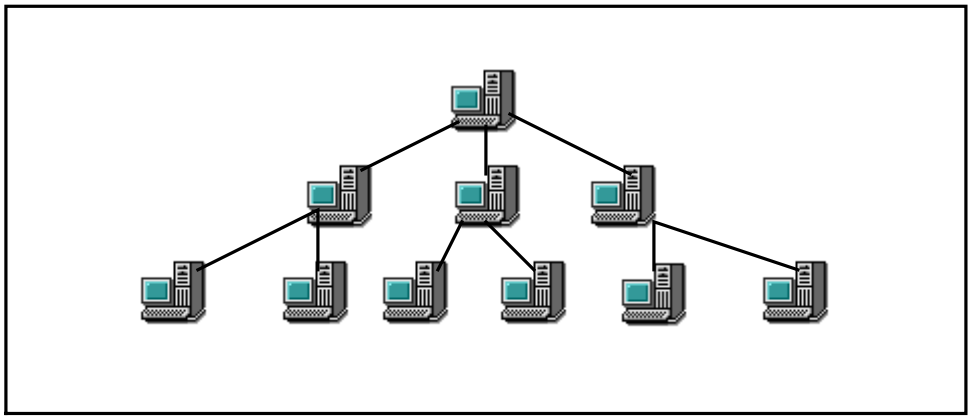


Figure 5.4. Experimental setup II.

The average values are calculated after 15 trials for each case. The results vary between 1897.4 milliseconds (for 20 rules in each node's rule set) and 3004.2 milliseconds (for 20 rules in each node's rule set). The overall results are indicated by the graph in Figure 5.5.

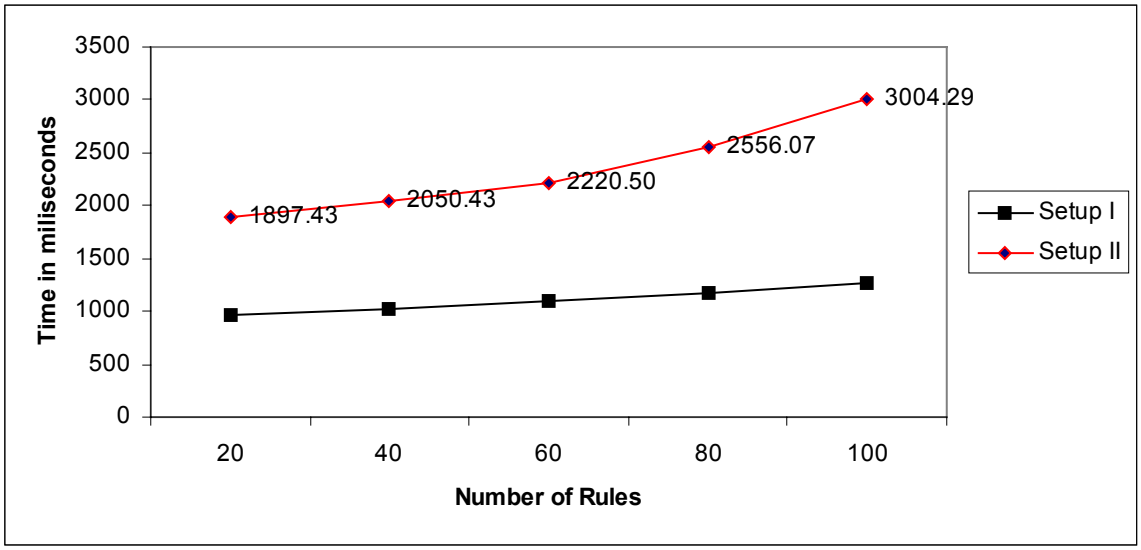


Figure 5.5. Execution times of the retrieving active rules on experimental setup II.

5.1.2.3. Experimental Setup III

The third experimental architecture (Figure 5.6) is composed of 13 nodes and 2 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture.

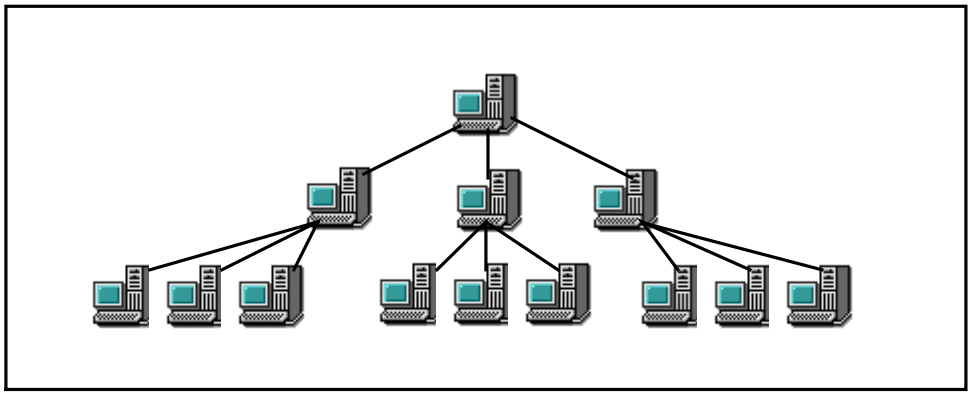


Figure 5.6. Experimental setup III.

The average values are calculated after 15 trials for each case. The results vary between 6414 milliseconds (for 20 rules in each node's rule set) and 12542.9 milliseconds (for 20 rules in each node's rule set). The overall results are indicated by the graph in Figure 5.7.

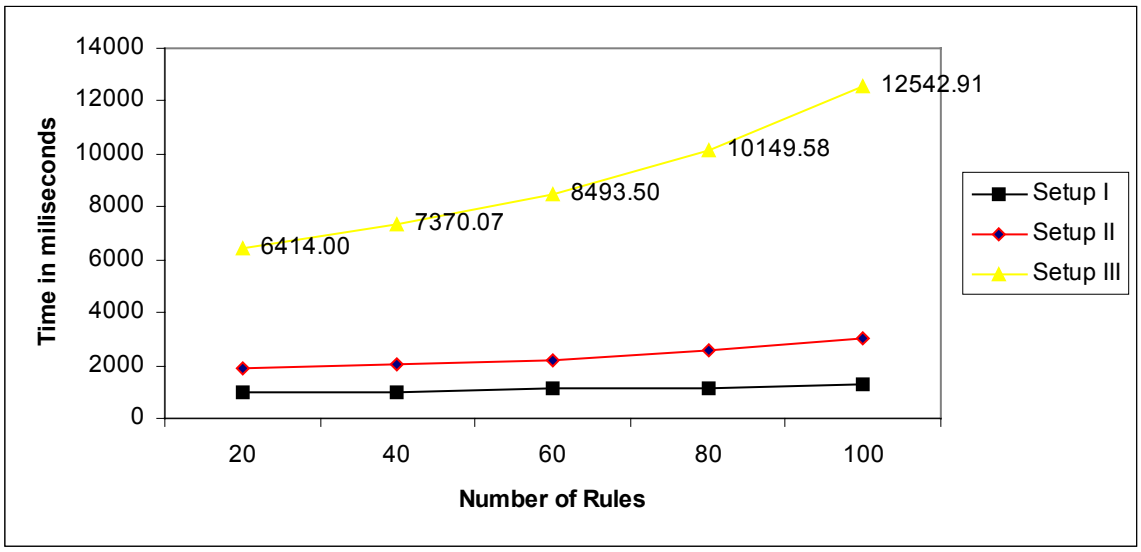


Figure 5.7. Execution times of the retrieving active rules on experimental setup III.

5.1.2.4. Experimental Setup IV

The third experimental architecture (Figure 5.8) is composed of 15 nodes and 3 levels of hierarchy. Experimental steps mentioned for the previous cases are also valid for this architecture.

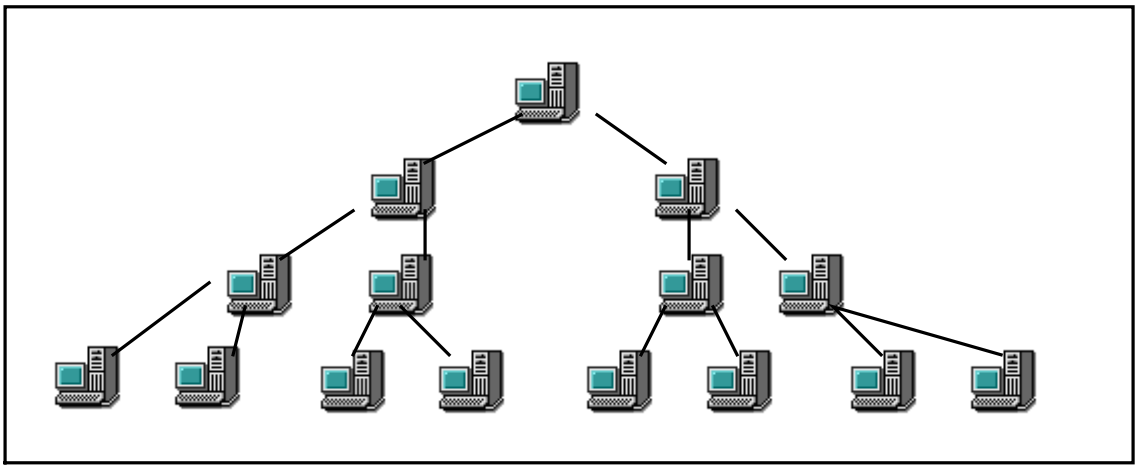


Figure 5.8. Experimental setup IV.

As in the previous cases, the average values are gathered after 15 trials for each case. The results vary between 12209.8 milliseconds (for 20 rules in each node's rule set) and 44917.4 milliseconds (for 20 rules in each node's rule set). The overall results are indicated by the graph in Figure 5.9.

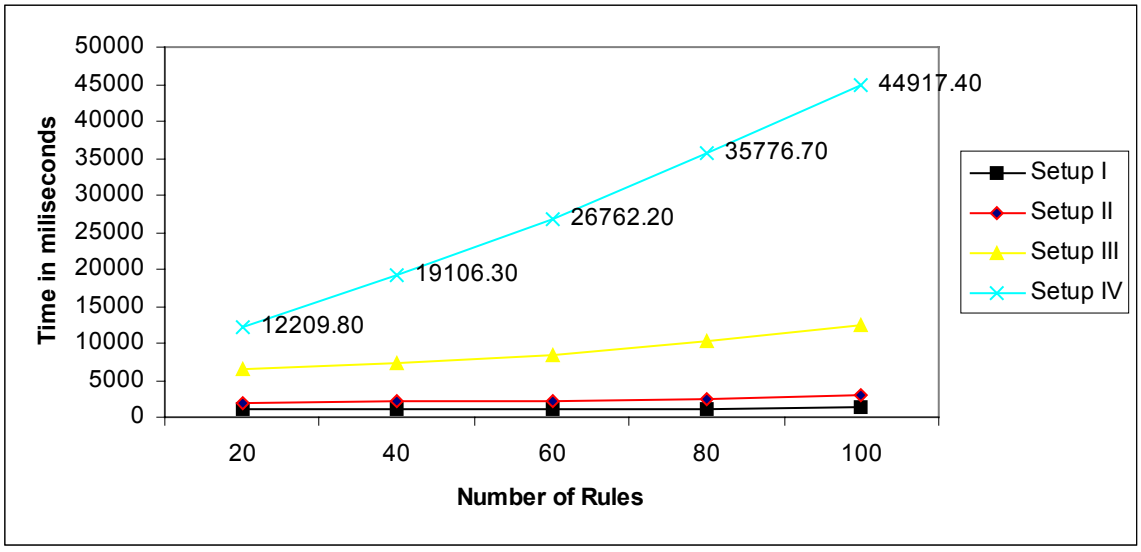


Figure 5.9. Execution times of the retrieving active rules on experimental setup IV.

5.1.2.4. Experimental Setup V

This experimental topology is composed of four nodes. As indicated by Figure 5.10, one of the nodes is configured to be the domain firewall; one is configured as a leaf firewall and the remaining two are configured as subnet firewalls. This topology is also used as a reference to compare the results found on emulation and on laboratory experiment.

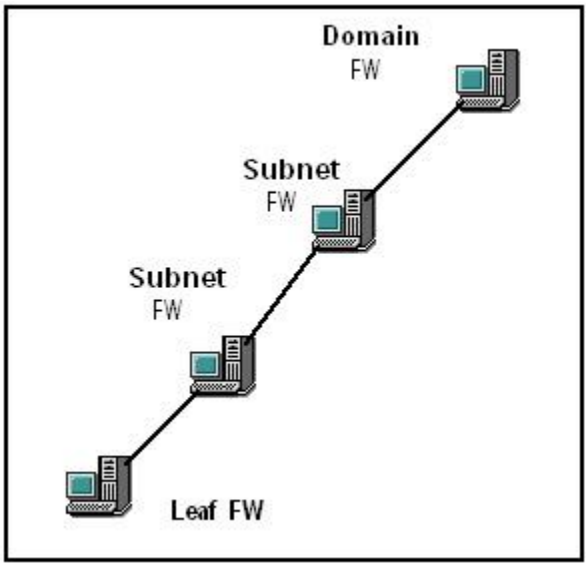


Figure 5.10. Experimental Setup V

As presented by Figure 5.11, the results are between 650.27 milliseconds for 20 rules case and 815.76 milliseconds for 100 rules.

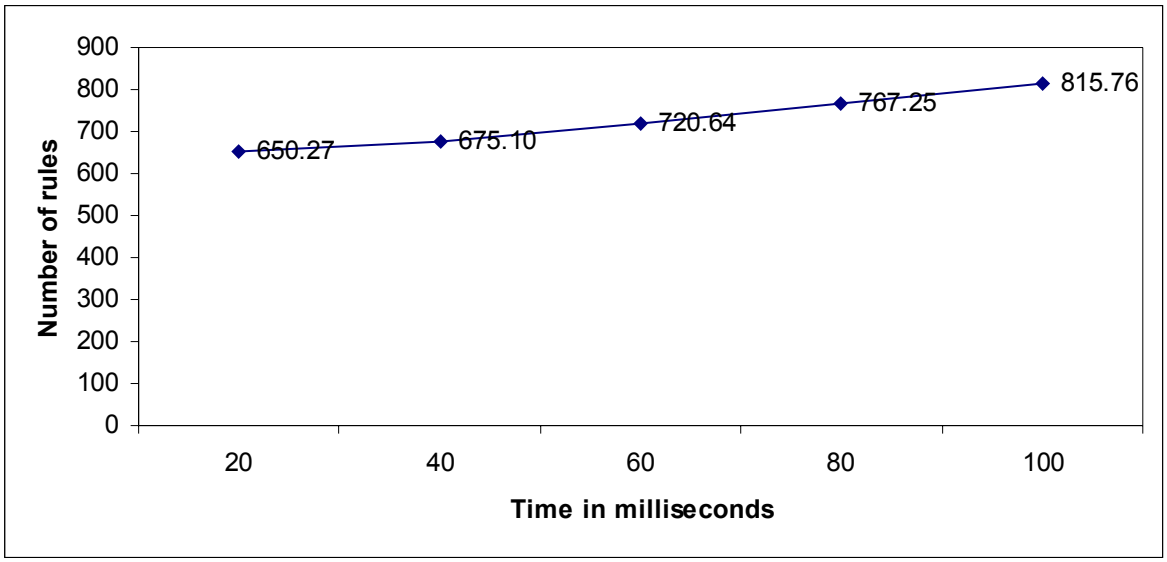


Figure 5.11. Execution times of the retrieving active rules on experimental setup V.

5.1.3. Experimental Setups For Laboratory

In addition to emulations, the same experiment was held on the laboratory on three different architectures. These architectures are described below. Each node in the architecture is a PC that has monitoring application is installed and running on it. As in the other cases, the experiment is retrieval of the current active firewall policy rules from the whole architecture by the domain firewall.

5.1.3.1. Experimental Setup VI

The first experimental setup shown by Figure 5.12 is constructed at the laboratory and composed of two nodes. One of the nodes is configured to be the domain firewall whereas the other one is a leaf firewall.

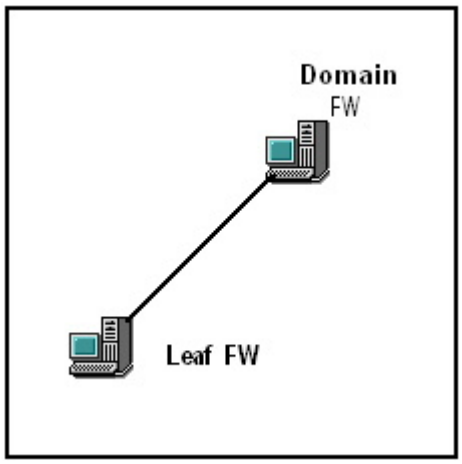


Figure 5.12. Experimental setup VI.

The results of the experiments performed on experimental setup VI with different number of policy rules (20, 40, 60, 80, 100) on each node is presented by the Figure 5.13. The results are between 313.2 milliseconds for 20 rules case and 385.6 milliseconds for 100 rules.

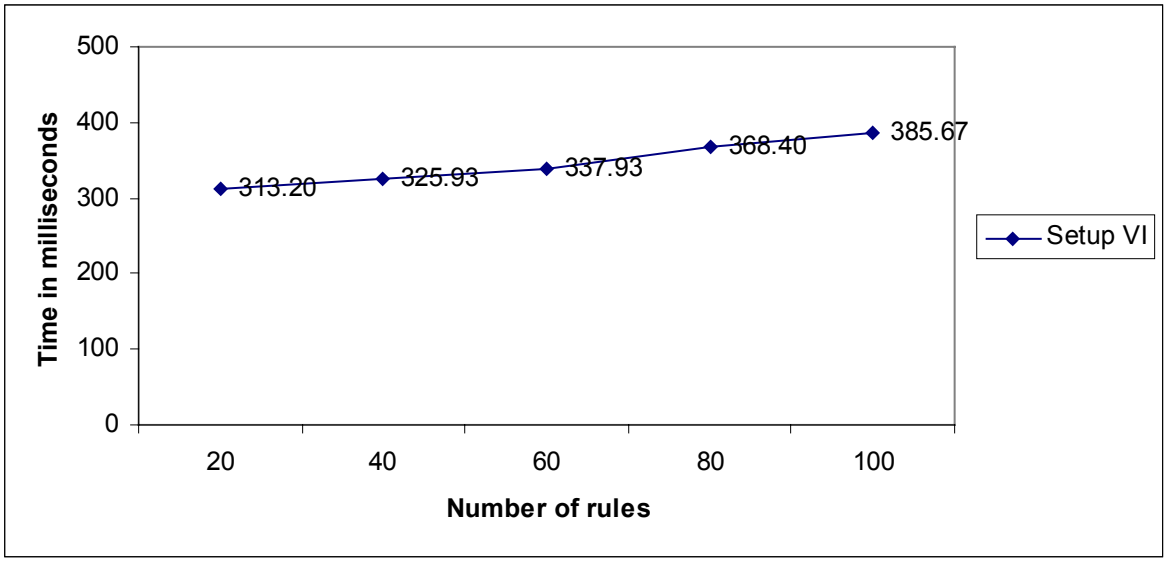


Figure 5.13. Execution times of the retrieving active rules on experimental setup VI.

5.1.3.2. Experimental Setup VII

The second experimental setup shown by Figure 5.14 composed of three nodes. One of the nodes is configured to be the domain firewall, one is configured as a subnet firewall and the last one is a leaf firewall.

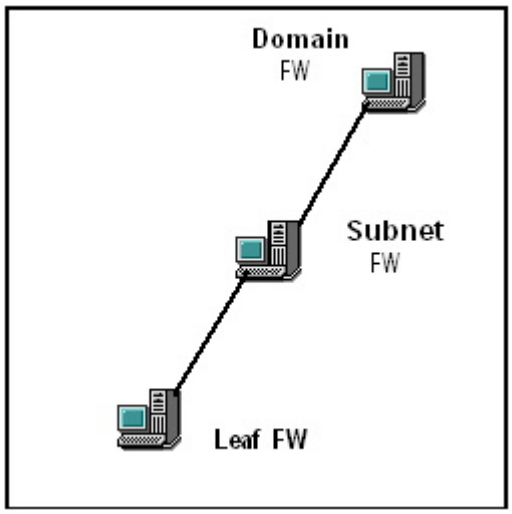


Figure 5.14. Experimental setup VII.

The results of the experiments performed on experimental setup VII with different number of policy rules (20, 40, 60, 80, 100) are between 605.2 milliseconds for 20 rules case and 638.5 milliseconds for 100 rules. Figure 5.15 indicates the overall measurements.

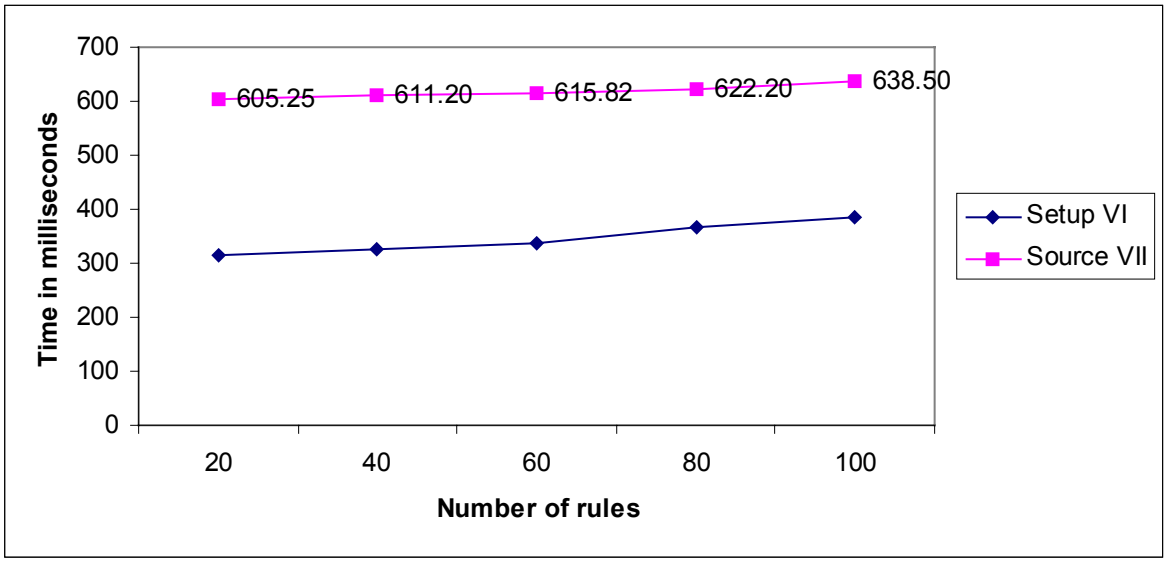


Figure 5.15. Execution times of the retrieving active rules on experimental setup VII.

5.1.3.3. Experimental Setup VIII

As mentioned previously, this topology is same as the one in Figure 5.10 used in emulation.

As presented by Figure 5.16, the results are between 703.5 milliseconds for 20 rules case and 838.2 milliseconds for 100 rules.

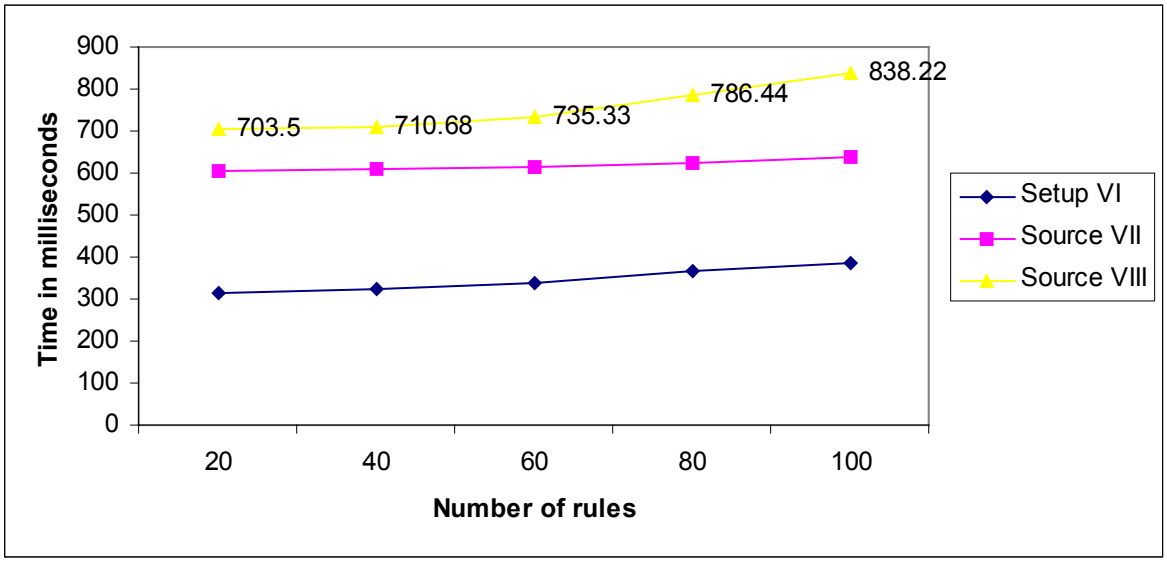


Figure 5.16. Execution times of the retrieving active rules on experimental setup VIII.

5.1.4. Comparison of Emulation and Laboratory Results

Topology on Figure 5.10 is used both for emulation and laboratory experiments. As indicated by Figure 5.17, the obtained execution times on both environments are very close to each other. The results vary between 650.27 – 815.76 milliseconds for emulation and between 703.5 – 838.22 milliseconds on real topology.

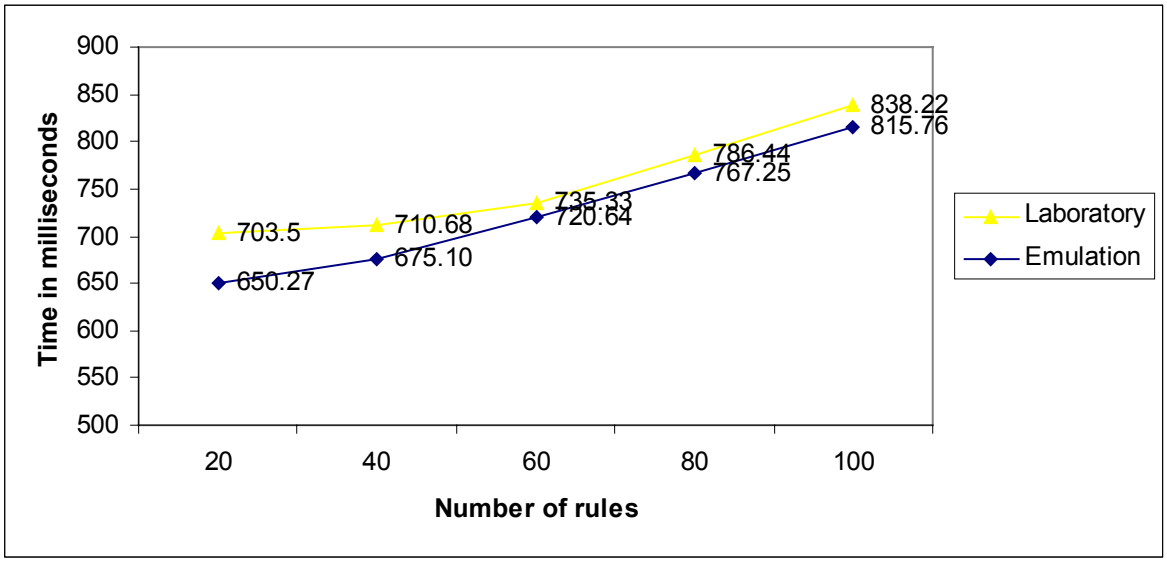


Figure 5.17. Comparison of emulation and laboratory results.

CHAPTER 6

CONCLUSION

The main objective of this thesis is to implement a monitoring application for a distributed firewall environment in order to keep track of some certain actions (Create, Read, Update, Delete) that are performed on the policy rule set. In order to accomplish this aim, first traditional firewalls are examined. Type types of firewalls, their role in network security, their advantages and disadvantages are presented. Then distributed firewall concept is explained and the comparison of two firewall designs is presented in terms of their performance in network security. The next stage is to give the details of distributed firewall environment for which the proposed monitoring application is designed. Once the architecture of the distributed firewall system is clarified, the details of monitoring application are explained. The last stage is to test and verify the monitoring application on an experimental setup that is constructed at the laboratory. The monitoring application is responsible for monitoring the CRUD actions that are performed on the policy rule sets of each individual firewall node on the experimental setup. Such an application will be very helpful in network security management in protecting the consistency among the overall security policy. The data provided by the application can be used to implement more advanced tools like policy anomaly detections tools.

REFERENCES

Al-Shaer, E.S. and H.H. Hamed. 2003. Firewall Policy Advisor for Anomaly Discovery and Rule Editing. In *IEEE / IFIP Integrated Management Conference*.

Al-Shaer, E.S. and H.H. Hamed. 2004. Discovery of Policy Anomalies in Distributed Firewalls. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China.*

Al-Shaer, E.S., Abdel-Wahab, H., and Kurt Maly. 1999. Hifi: A new Monitoring Architecture for Distributed Systems Management. In *International Conference on Distributed Computing Systems, Austin, Texas*.

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

CERT Advisory. 2007. Smurf IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-1998-01.html (accessed September 25, 2007)

Chapman, Brent and Elizabeth Zwicky, eds. 2000. *Building Internet Firewalls*. Cambridge: Orielly & Associates Inc.

Cisco Documentation. 2007. Evolution of the Firewall Industry. http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ch3.htm#xtocid0 (accessed September 19, 2007).

Gatus, G. E. P., Safavi-Naini, R. and Willy Susilo. 2004. Policy Distribution Using COPS-PR in a Distributed Firewall. In *Australian Telecommunication Networks and Applications Conference*.

Ioannidis, S., Keromytis, A. D., Bellovin, S.M. and J. M. Smith. 2000. Implementing a Distributed Firewall. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, *Athens, Greece.*

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

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

Linux Network Administrators Guide. 2007. What Is a Firewall?. http://www.faqs.org/docs/linux_network/x-087-2-firewall.introduction.html (accessed September 19, 2007).

Rubin, Rachel. 2002. Smokey: A User-Based Distributed Firewall System. *Department of Computer Science, University of California, Berkeley.*

Wikipedia, The Free Encyclopedia. 2007. Firewall. http://en.wikipedia.org/wiki/Firewall (accessed September 19, 2007).